Securing Cover-File Without Limitation of Hidden Data Size Using Computation Between Cryptography and Steganography

A.A.Zaidan, Fazidah.Othman, B.B.Zaidan , R.Z.Raji, Ahmed.K.Hasan, and A.W.Naji

ABSTRACT-- The rapid development of multimedia and internet allows for 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 and encryption information or a combination between them. 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. In this paper, a new information hiding system is presented. The aim of the proposed system is to hide information (data file) in an execution file (EXE).The new proposed system is able to embed an information in an execution file and also able to retract the hidden file from the execution file. Meanwhile, since the cover file might be used to identify hiding information, the proposed system considers overcoming this dilemma by using the execution file as a cover file.

(keyword): Cryptography, Steganography, Information Hiding, Advance Encryption Standard, Portable Executable File.

Aos Alaa Zaidan - Masters Student, Department of Computer Science & Information Technology, University Malaya, Kuala Lumpur, Malaysia, phone: +60172452457, Postcode: 50603 and Email: awsalaa@perdana.um.edu.my.

Mrs. Fazidah Othman - Lecturer, Department of Computer Science & Information Technology, University Malaya, Kuala Lumpur, Malaysia, fazidah@um.edu.my.

Bilal Bahaa Zaidan - Masters Student, Department of Computer Science & Information Technology, University Malaya, Kuala Lumpur, Malaysia, bilal@perdana.um.edu.my.

Raji Zuhair Raji - Masters Student, Department of Computer Science & Information Technology, University Malaya, Kuala Lumpur, Malaysia, rhaddawi82@perdana.um.edu.my

Ahmed.K.Hasan - Masters Student, Department of Computer Science & Information Technology, University Malaya, Kuala Lumpur, Malaysia, Email: Faahkh@perdana.um.edu.my.

Dr Ahmed Wathik - Lecturer, Department of Electrical & Computer Engineering, International Islamic University Malaya, Kuala Lumpur, Malaysia, ahmed@iiu.edu.my.

I. INTRODUCTION

With the emergence and development of computer science and informatics emerged the urgent need to find ways to avoid Muggers and computer hackers from stealing or disclosure of Data and sensitive task information. It was learned that arose organization (Cryptography) optimal way to achieve this end, this science has evolved steadily emerged the systems and very efficient techniques But with the advent of information and communications networks global information network (Internet) has become the issue of complexity of the privacy and unattainable due to achieve this process and to ensure access to the data required was necessary to be accessible to everyone online common, and here is the problem of inefficient organization, vision statements as loose enough to push the spam or the attacker to believe that important or sensitive data lies in these random or encrypt text, some techniques comes using anti-encrypted to attempt to dismember the symbols and creating content, even if unable to do so, it might tampering or distorts or used some means available to prevent access to its goal[1].Elsewhere Governments began losing control of the encrypted messages exchanged between the institutions, companies and the possibility of these texts contain encrypted information may be against the security and the public interest, and therefore resorted to some governments to prevent the use of the organization for users of communications networks for personal purposes[1],[2].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 [3]. One of the latest techniques that have been used in this area by researchers at the Mount Sinai School MOUNT SINAI Medical in New York New York in 2007, as they managed to hide the secret texts in sentences Strand human DNA (Human DNA) by using a technique called genetic system coverage (Genomic Steganography), and by placing signs resolution to be agreed upon in the nuclei chromosomes and then integrate these with millions sentences and sent to the other end. To extract the secret message is soaking get special distinction sentences used on the other and then placed under the microscope to extract the

required text [3]. The oldest Authentications on Steganography taken from the legendary stories Greeks Herodotus and then back to the fifth century BC, these sources indicate that they felt they fly head of the Messenger and then write the letter secret in the head, leaving hair to grow then be sent to the required which is a re-extraction letter [3],[4]. Authentications and other writing secret messages on the wood panels and then covered wax and will be hid those writing panels appear free of anything and they were killing their animals as rabbit example corner confidential letter inside it. [4].Other means that the common use since the first century AD, invisible inks Invisible Inks, which was able to write a confidential letter with any other non-value-confidential and usually write between lines, for example those rabbis some fruit juices Fruit Juices, milk, urine, vinegar, and all these species become dark and visible when exposed to heat the written document [4]. Then these kinds of inks evolved with the evolution of science chemical was used vehicles carrying chemical characteristics of the same old species with a more accurate and efficient have been used during the First and Second World Wars in the military secrecy of correspondence[3][4] .Other technical been used during World War II is sending a message hidden within another message is not relevant, and based on the idea of a nomination letters every word of the letter counterfeit representation of characters from the characters letter requested confidentiality. Moreover, there are numerous ideas for the same method is used to be more than characters, or take certain words or phrases within the text fake and leaving the rest. Finally, it should be noted that the senior researcher in the area of concealment and science-based organization itself is German Johannes Trithemius ((between (1462-1526), and the oldest books in the area of coverage Posted by Gaspari Schotti)) in1665 in the name of (Steganographyica) and (400) contains a page where all the ideas included (Trithemius) [2][4].

II. INFORMATION HIDING

With the emergence and development of computer science and informatics emerged the urgent need to find ways to avoid Muggers and computer hackers from stealing or disclosure of data and sensitive task Information. Hidden information in the cover data is known as the "embedded" data. The "stego" data is the data containing both the cover signal and the "embedded" information. Logically, the processing of putting the hidden or embedded data, into the cover data, is sometimes known as embedding, while split the embedded information from the cover is called "extracting"[4].

In spite of the important characteristics achieved by hiding information system, there are some weaknesses that could be exploited by the attacker and then steal these hidden statements or break it. Usually, people would hide their data in a multimedia files such as, an image or video file. This can be done using many tools in the market. The concept of hiding data into multimedia files becomes very popular and no longer secret to the attacker. Through the knowledge of the attacker, they can simply find out the file and extract the data out from it. Besides that, as for the user, there are some limitations in terms of the size of file that can be hidden into a file. The size of file or data to be hidden must be about the same size as the cover file. For example, user need to find an image which size is large enough to cover a 700Mb size of file, usually a multimedia files. The most common methods of concealment and simplest is Switches (Binary Digit) known briefly as (bit), least significant known as (LSB), where it is altered binary digit characters to the message characters to be hidden, after conversion of such characters to byte as well as the American standard code for information interchange (ASCII)[5].

We cannot use this method here because switching binary digit might lead to an increase or decrease the value of letter by (1); this leads to the advance of this letter with the neighbor letter, for example the letter (C) in English represent in binary (100 0011) with replacement the Least Significant bit the binary value become (100 0010) which is represent (B) in English, that will make the carrier text become a meaningless, which denies the goal of hiding technique[5].Therefore researchers come out with other technique which exploits spaces between strings and words in the carrier text**.** Methods of concealment in the text is weak, inefficient, not suitable for the application and the main disadvantage include; need large text to hide small message that leads to an increase in the size of the cover (hiding data in wave module)(8-bit)[4],[5]. It was pointed out that this pattern is represented by each audio sample size (1 byte/sample) and common here that the process of concealment in last significant bit each model where the switch imperceptibly by authorizing human. In the 8-bit audio file representation means that each model will be represent eight size, meaning that there is (256) audio level can be represented at the highest in this type is between (0-255).In this kind of representation researcher hide data in the first bit that less important to detect this concealment, that is to conceal ratio (12.5% from the size of the file).The size of the output of the hidden data file is larger comparing to the encoded data. In its most efficient possible case, it may reach double the size of encoded data or a bet less . In some situations output file may reach eight times larger than the encoded data, as well as certain files of media images and text, files may reach fifty times larger when they are encoded. Its known that executable file; size varies depending on application, some files size 2 Mega bytes such as images, and other files more than 650 mega bytes like operating system (Windows, UNIX, etc.)[5]. this disparity in the size of the files gives flexibility to user to use it as a cover file to hide data regardless of the data size. Over all we tried to find a way to overcome this problem and using Executable file (EXE File) as a cover for information to be hidden which solved the problem of the size.

III. ADVANCE ENCRYPTION STANDARD

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 [5].

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 [5].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 [5], [3].

B. The Round Transformations [5][6].

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.

- 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).

- 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).

C. The Rounds

A round transformation is composed of four different transformations.

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

Figure 1. Four Different Transformations.

The final round is like a regular round, but without the mix columns transformation:

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

Figure 2. Final Round.

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.

IV. PORTABLE EXECUTABLE FILE

The proposed system uses a portable executable file (PE-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.
- 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.

- 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.

- 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.

- 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.

- ELF:

The ELF, Executable Linkable Format 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. 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 [18].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 [6].

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.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 [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 [6][7].

C. Techniques Related with PE

Before looking inside the PE file, we should know special techniques some of which are [5],[6],[7]:

- 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 [8].

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[8]. 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

There are two unused spaces in PE file layout [8], and these unused spaces are suggested to hide a watermark. The size of the second unused space is different from one file to another .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 [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.

- First region (MS-DOS 2.0 Compatible.EXE Header).
- Second region (Unused).
- Third region (OEM Identifier, OEM Information, Offset to PE Header).
- Fourth region (MS-DOS 2.0 Stub Program & Relocation).
- Fifth region (Unused).
- Sixth region (PE Header).
- Seventh region (Section Headers).
- Eighth region (Image Pages):
  - Import info
  - Export info
  - Fix-up info
  - Recourse info
  - Debug info
- Ninth region (Base of Image Header).
- Tenth region MS-DOS 2.0 Section (For MS-DOS Compatibility Only).

V. 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**.**

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. This way needs the customer to install the system application manually as shown in Figure 3.
- Developing the file handling functions depending on the existing file handling routines. This way can be performed remotely as shown in Figure 4. 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 paper.

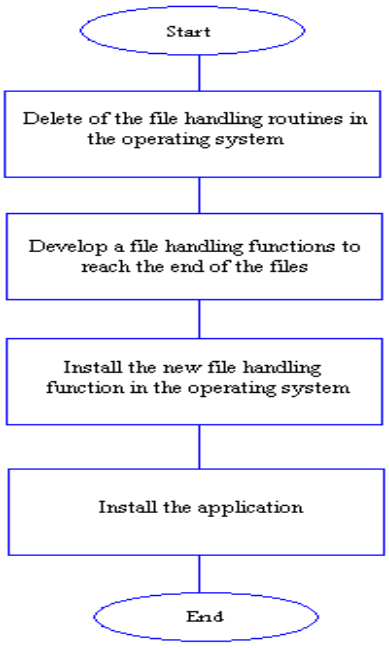

Figure 3. First Method of the System Concept

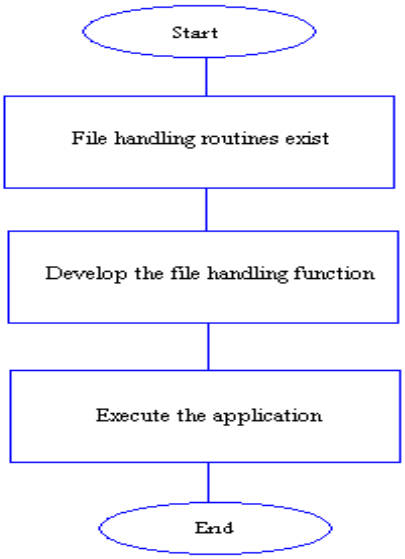

Figure 4. Second Method of the System Concept

B. System Features

This system has the following features:

- The cover file can be executed normally after hiding operation Because the hidden information already hide after the end of 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.

- There is no limitation on the hidden file size where you can hide any file of any size regardless of the size of hidden information by structure on the property of the EXE file, so that the EXE cannot identify the size of the EXE file, so can using type of EXE file such as JDK whose contain number of different size (72MB, 77MB or 65MB), other world disparity in the size of the executable files, so can hide any size inside it without guessing the real size of the information hidden by the attacker. Furthermore, when hide after the end of EXE file, there is no limitation of the size files which must be hiding after the end of EXE file, open space of any size.

- 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.

- The hidden information can be of any type of multimedia files (Text, Audio, Video or Image) of any size without limitation and also can hidden all type of multimedia files in the same time inside the same cover, so can put (Text, Image, Video and Audio) in one folder and compressed them and then choose the compressed folder as a information hiding, in that way can hidden all in the same time.

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 VB.net. The block flow of hiding operation can be performed as shown in Figure 5. The block flow of retraction operation can be performed as shown in Figure 6. The following algorithm is the hiding operation procedure:

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 of file.
- Write the file name after the end of file (EXE 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.

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 file.
- If end of file pointer exist.
- 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).

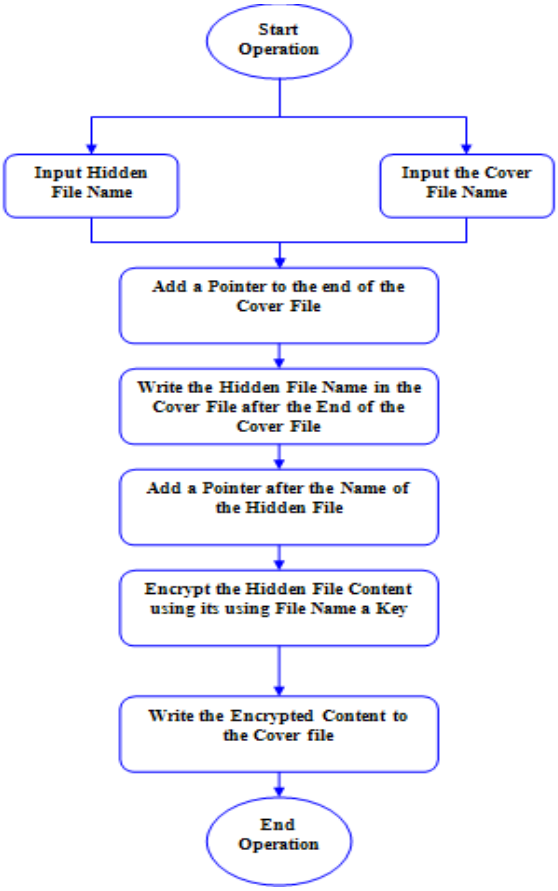

Figure 5. Block Flow of Hiding Operation.

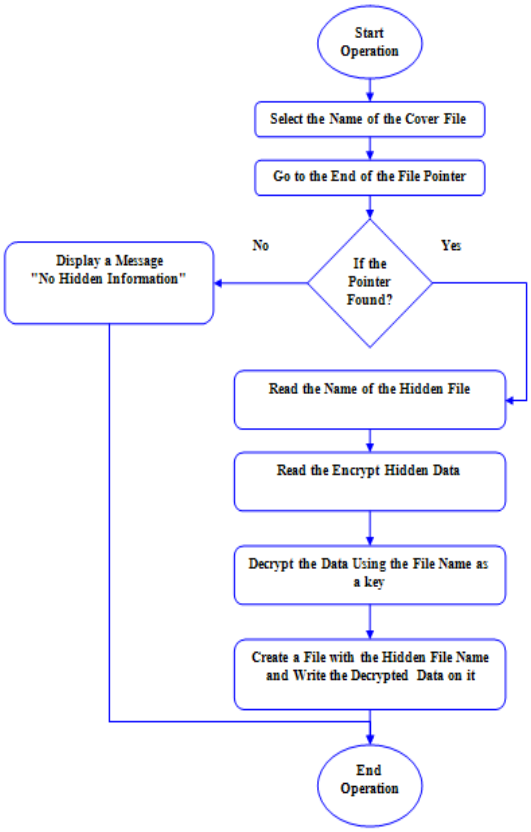

Figure 6. Block Flow of Retraction Operation

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.
- One of the important conclusions 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 independent.
- 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) because there are no limitations or restrictions on the type of the message (image, sound, text and video).

ACKNOWLEDGEMENT

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

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[5] 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,

[6] Johnson, N. F. S. D, Z., "Information Hiding: Steganography and Watermarking-Attacks and Countermeasures", Center for Secure Information Systems (CSIS), Boston/Dordrecht/London, George Mason University, 2006.

[7] Katzenbeisser, S. P., A. P, "Information Hiding Techniques for Steganography and Digital watermarking", available from: Artech house pub, 2005.

[8] Katzenbeisser S. & Petitcolas, F. A., "Information Hiding Techniques for Steganography and Digital Watermarking", Artech House, USA, 2001.