https://doi.org/10.31871/IJNTR.7.6.31 International Journal of New Technology and Research (IJNTR)
ISSN: 2454-4116, Volume-7, Issue-6, June 2021 Pages 34-40

Enhanced Data Security Framework by Using a Combination between Cryptography and Steganography Technique

Maysara Mazin Badr Alsaad

Abstract—**Problem statement: Most of hiding information operations are performed within many of the traditional covers such as text, image, audio and video files. These covers become known amongst hackers where they can detect the hidden data or extract through their experience and proficiency in this field. The framework which was designed in this research deals with another kind of covers, which is the portable executable file that has (.exe) extension. Approach: The hidden method which was used in this framework rely on the manner of embedding bit by bit sequentially beyond the end of these files. In general, the hidden information within these covers is not considered a guaranteed protection. Therefore, before the hiding process, we reinforce this information through one of symmetric encryption algorithms, namely; Twofish. The combination occurred between hiding and encryption techniques are due to the lack of a typical technique that can be performing both of these functions in a single operation. Results: This framework is implemented by using Visual Basic.net where the findings show that the framework is able to embed and extract all types of files, regardless their size, even if the secret file is greater than the cover. Conclusion: The stego file can perform normally after the embedding process with the secret data inside where no antivirus software can detect the hidden data.**

Index Terms — **Steganography, Cryptography, Information Hiding, Twofish Algorithm, Portable Executable File.**

I. INTRODUCTION

With the emergence and development of computer science and informatics emerged the need to find ways of securing information so as to avoid computer hackers from stealing data and/or sensitive information (Zaidan *et al.*, 2009a). Cryptography; a science that have evolve steadily over the years, was an early technique used to overcome this and was efficient in its role of preventing hacking into sensitive messages. However, with the advent of global information and communication networks, a more complex problem arose where information of any kind are readily accessible to anyone online. Information attackers have developed many encryption techniques to try to decipher encrypted information and these, even though unsuccessful, might at times, distort information enroute their set destination (Saleh *et al.*, 2016). According to (Avedissian, 2008 and Zaidan *et al.*, 2008), this resulted to governments of some countries preventing the usage of the communication networks for personal use as they were losing control of encrypted messages exchanged between various governmental institutions and companies. More so, there was the possibility of these texts containing encrypted information that may be against public security and interest (Saleh *et al.*, 2015). This created an urgent need to find alternate technique to hide information that would overcome the shortcoming of the encryption model, giving rise to information technology concealment (Steganography), which is based on a different principle from the encryption technology, where information are buried (Information Embedding) within other media carrier and rendered imperceptible to hackers and attackers and from the public domain of information users, while the content monopoly remains “on the relevant agencies, which alone knows how to extract the content” (Smith and Comiskey, 1996; Clelland *et al.*, 2007). Early sources of the ideas of steganography began with of the Greek Herodotus back in the 5th century BC when it was proposed that secret messages be written on the shaved head of a messenger and then allowing the hair to grow back before he is sent to the destination where the information is retrieved (Daren and Scott, 2007; Dorothy, 2006). In the olden days in countries like Rome and Greece, messages were usually written on wax that was poured on top of some stone tablets. If the sender of the information/message wanted to hide the message - for purposes of military intelligence or civil security, for instance - they often used steganography. The wax would be shaved off and the intended message would be written directly on the tablet. Wax would then be poured on top of the message, thereby hiding not just its meaning but its very existence (Neil, 1995). In recent times, steganography has gradually metamorphosed into the art of hiding a smaller message within a larger one in such a way that others cannot discern the presence or contents of the hidden message (Johnson *et al.*, 2006; Saleh *et al.*, 2016). Other means that were in common use in the 1st century AD was the use the use of invisible ink to write messages, usually in between lines of visible message of non-confidential nature. For example, rabbis in those days used fruit juice, milk, urine, vinegar, which becomes dark but visible when the written document is exposed to heat (Zaidan *et al.*, 2009b). In the military secret correspondence during the 1st and 2nd world wars, a more advanced chemical which served as ink was used to pass secret messages. Finally, it should be noted that the senior researcher in the area of concealment and science-based organization is German; Johannes Trithemius (1462-1526) and the oldest books in the area of coverage was posted by Gaspari schott in 1665 (Zaidan *et al.*, 2009b).

Maysara Mazin Badr Alsaad, Faculty of Information Science and Technology (FTSM), Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia.

II. THEORETICAL REVIEW

Steganography vs. Cryptography:

According to (Saleh *et al*., 2016), the information protection systems are grouped under their areas of function as either steganography or cryptography or a combination of the two. Steganography involves hiding the information in such way that its existence will be undetectable by unauthorized persons. Cryptography on the other hand entails encrypting the information so that it cannot be comprehended when discovered. Steganography and Cryptography are in fact complementary techniques and are not mutually exclusive (Othman *et al*., 2009; Naji *et al*., 2009). No matter how well concealed a message is, it is always possible that it will be discovered (Hamid *et al*., 2009). Equally, no matter the strength of the algorithm used, an encrypted message, when discovered, can be deciphered through cryptanalysis. But the combination of steganography and cryptography provides two important added protections:

a) In doing this, we make it far less likely that an encrypted message will be found (Othman *et al*., 2009; Naji *et al*., 2009).
b) On the less likely chance that the hidden message is discovered, the hacker still faces the daunting challenge of deciphering it.

The table below shows comparison between steganography and cryptography art (Usman *et al*., 2018):

Table 1: Comparison Between Steganography and Cryptography Technique

Steganography	Cryptography
1. Steganography is hidden writing. The message is there, but nobody notices it. However, once noticed, it can be read.	1. Cryptography is secret writing. Anybody can see the message, but nobody else can read it. Usually, this is because its letters have been re-arranged, or replaced by different letters, according to some scheme that only the sender and receiver know.
2. Involves embedding the message in a larger one where it cannot be detected.	2. Involve making the message undecipherable; unreadable and not understandable.
3. Message unchanged after steganography.	3. Structure of messages changed after encryption.
4. Final result of steganography is stego-media.	4. End message is cipher text.
5. Information hidden cannot be seen.	5. Encrypted message can both be seen and modified by someone else.
6. Steganography can hide encrypted message.	6. Steganography cannot affect encrypted message.
7. Steganography is relatively unknown.	7. Cryptography is an old science and well in use.

The Challenges of Hiding Messages Multimedia Files:

The concept of hiding information in multimedia files is fast gaining popularity and no stranger to hackers. A determined hacker can detect a hidden file and access the information. The user as well faces the challenge of finding the right size of the file to be used as a cover for the information to be hidden. Overall, we tried to find a way to overcome this problem using executable files as a cover for information to be hidden which solved the problem of the size (Avedissian, 2008; Saleh *et al*., 2015). An executable file has varying sizes depending on application. Some files are 2 megabytes such as images, and other files more than 650 megabytes like operating system (windows, UNIX, etc) (Clelland *et al*., 2007).

The hidden method which will be used in this framework, rely on the manner of embedding bit by bit sequentially beyond the end of executable files. In general, the hidden information within these covers is not considered a guaranteed protection. Therefore, before the hiding process, we reinforce this information through one of symmetric encryption algorthims, namely; Twofish (Rachmat and Samsuryadi, 2019).

Twofish Algorithm:

In 1997, the National Institute of Standards and Technology (NIST) proposed the Advanced Encryption Standard (AES) to replace DES. Twofish is one of the several projects that satisfy NIST design criterions, and was one of the five candidates for AES finals. There are many specifications helped to make it an excellent candidate to be considered as a standard encryption algorithm through several aspects, which are (Reyes *et al*., 2018; Neha and Kaur, 2016):

- A128-bit symmetric block cipher
- Key lengths of 128-bit, 192-bits and 256-bits
- No weak keys
- Efficiency, both on the Intel Pentium pro and other software and hardware platforms.
- Flexible design e.g. accept additional key lengths, be implementable on a wide variety of a stream cipher, hash function and MAC.
- Simple design, both of facilitates case of analysis and ease of implementation.

Additionally, I imposed the following performance criteria on my design:

- Accept any key length up to 256 bits.
- Encrypt data in less than 500 clock cycles per block on an Intel Pentium, Pentium pro, and Pentium II, for a fully Optimized version of the algorithm.
- Not capable of setting up a128-bits key (for optimal encryption speed) in less than the time required to encrypt 32 blocks on a Pentium, Pentium pro, and Pentium II.
- Not contain any operations that make it inefficient on 8-bits, 16-bits and 32-bit microprocessors.
- Have a variety of performance tradeoffs with respect to the key schedule.

Brief Description of the Twofish Cipher:

Twofish uses a 16-round foisted network for computation (Scheier *et al*., 1998). The diagram below show the general structure of the Twofish block cipher (Rachmat and

Samsuryadi, 2019).

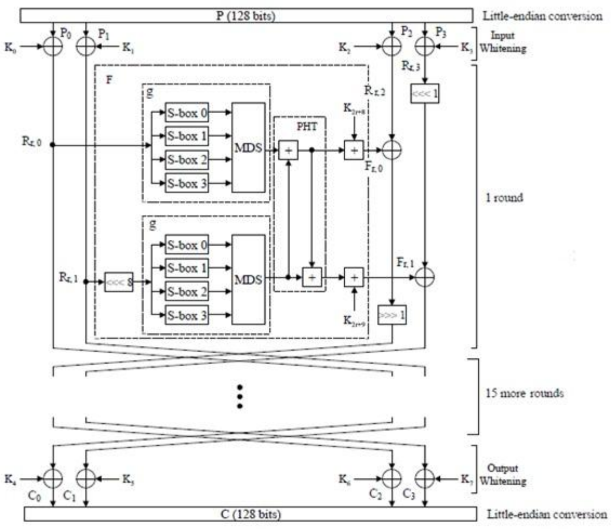


Fig. 1: General Structure of Twofish Block Cipher

The Twofish structure has encryption and decryption circuits that are almost similar which permits the cipher to be done in an area cut nearly in half (Rachmat and Samsuryadi, 2019). To decrypt, one has to reverse the order of the internal keys used for encrypting the message. The Fistel network used for the computation is made up of a basic block known as F-function, a key reliant function which has objective property that allows it to map the input into a specific output (Rane, 2016). The F-function works by using it interactively in each round to build a strong encryption algorithm that consists of several transformation and computations as follows (Neha and Kaur, 2016):

- g-function
- Pseudo- Hadamard Transformation
- Addition modulo 2^{32}

The F-function, combined with some keys, uses half of the block to create the above operations to generate an output that is EX-ORED with the other half of the block. One half of the block then changes places with the other half before the next round of operations is done, but after the final stage of the operation, each half remains in place and do not change places (Reyes *et al*., 2018; Rane, 2016).

Within the structure, each of the g-function consists of S-boxes that are key reliant which raises its resistance to cipher attacks. Each of these S-boxes contains permutations q_0 and q_1 with an 8 bit fixed values. The input, x_i, to each S-box is of length 8 bits. Depending on the S-box and the value of x_i, the value of x_i is inputted with the value of y_i as follows:

$$y_0 = s_0(x_0) = q_1\,[q_0[q_0[x] \text{ XOR } s_{0,0}] \text{ XOR } s_{1,0}]$$
$$y_1 = s_1(x_1) = q_0\,[q_0[q_1[x] \text{ XOR } s_{0,1}] \text{ XOR } s_{1,1}]$$
$$y_2 = s_2(x_2) = q_1\,[q_1[q_0[x] \text{ XOR } s_{0,2}] \text{ XOR } s_{1,2}]$$
$$y_3 = s_3(x_3) = q_0\,[q_1[q_1[x] \text{ XOR } s_{0,3}] \text{ XOR } s_{1,3}]$$

Where $s_{i,\ j}$ (i=0, 1 and j=0...3) are bytes which are obtained by splitting S_0 or S_1 into four bytes (Reyes *et al*., 2018; Scheier *et al*., 1998). These results are multiplied by 4x4 MDS (Maximum Distance Separable) using Galois Field, GF (2^8). The MDS matrix used in the Twofish is a constant 4x4 matrix. $x^8+ x^6+ x^5+ x^3+1$ is the primitive polynomial. The end result is the final output from the g-function. The Pseudo-Hadamard Transform is a simple mixing operation that can be implemented using least resources in hardware (Scheier *et al*., 1998). Just by using two inputs *a* and *b*, each of length 32-bit, PHT can be performed as follows:

$a' = a + b \bmod 2^{32}$
$b' = a + 2b \bmod 2^{32}$

Using the output from PHT and round keys, the addition modulo 2^{32} can be carried out. The Twofish cipher block also have a special design unit called the key schedule which generate the keys needed for input and output whitening (K_0-K_7) while performing each of the 16 rounds (K_8-K_{39}) operation and also generate the key for the S-boxes (S_0-S_1). The same primitives used in the round function are used in the key schedule for keys K_0-K_{39} alongside 128-bit user key. To obtain the key material for the S-boxes, the Reed Solomon (RS) mapping is used. The keys S_0, S_1 are derived by using 128-bit user key to do a matrix multiplication of Galois Field-GF (2^8). The primitive polynomial is $x^8+ x^6+ x^3+ x^2 +1$. The RS matrix is a 4x8 matrix derived from the RS code (Rane, 2016; Scheier *et al*., 1998).

Portable Executable File (PE):

The system being proposed here is the use of a portable executable file (PE-file) as a cover file for burying an execution program. This subject shall be presented in two layouts which are; executable file types and PE file layout (Shetty and Ranjan, 2018; Namanya *et al*., 2019).

Executable File Types:

Executable file types have many varieties and each is unique to the various available operating systems. These executable file types are (Jalab *et al*., 2010 - (Shetty and Ranjan, 2018; Tian & Yang 2021).

1) DOS-MZ Executable: This format is not popular or useful for DOS, and it cannot be run by any other version of DOS, but can usually be run by 32-bit Windows and OS/2 versions.

2) Linear Executable (LE): This format is not used for OS/2 applications anymore, but instead for VxD drivers under Windows 3.x and Windows 9x, and by some DOS extenders.

3) Portable Executable (PE): This format can be run by all versions of Windows NT, and also Windows 95 and higher, also partially in DOS. In addition, it is supported on all CPUs and used for device driver such as (.exe, .dll, .obj, .sys, etc). (Namanya *et al*., 2019).

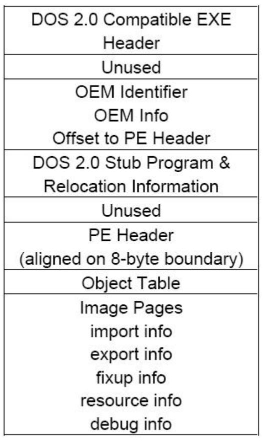
So, the proposed framework in this paper uses a portable executable file that has (.exe) extension as a cover-object to embed the secret information inside it, due to his good specifications which are shown above (Shetty and Ranjan, 2018).

PE File Layout:

In the PE-file layout, one finds two unused space. The size of the second unused space varies from one to another (Othman *et al*., 2009; Namanya *et al*., 2019). It has been proposed that this unused space be taken to hide watermark.

The significance of this proposal is to essentially help leave room for programmer to create back doors for all of their developed application as a solution to their myriad of problem like forgetting of password. The implication of this is that customer will feel unsafe as they would believe that the programmers can hack into their system at will. Customer would then want to hire only programmers they trust to develop their application. The challenge for programmers therefore is their need to show that their applications are safe anywhere no matter the level of relationship that has been establish with the customers. In this proposed framework, a solution is suggested for this problem (Shetty and Ranjan, 2018; Abdulrazzaq *et al*., 2013; Tian & Yang 2021).

The solution is to enclose the password in the executable file of similar system and for the customer to then retrieve the application himself. The system presents another challenge because the steganography needs adequate knowledge of all files format to determine how to hide information in the files. This technique is difficult because there are always large numbers of the file format and some have no way to hide information in them (Abdulrazzaq *et al*., 2013; Othman *et al*., 2009; Tian & Yang 2021). The typical executable file layout is as shown in Figure 2.



DOS 2.0 Compatible EXE Header
Unused
OEM Identifier OEM Info Offset to PE Header
DOS 2.0 Stub Program & Relocation Information
Unused
PE Header (aligned on 8-byte boundary)
Object Table
Image Pages import info export info fixup info resource info debug info

Fig. 2: Typical 32-bit Portable (.exe) File Layout

III. Materials and Methods

The concept of the proposed framework:

The concept of this proposed framework is to hide the password or message beyond the executable file in such a way that there is no task like open file, read or edit file, and close file in the operating system that can retrieve it. The operation, which relies on extant file handling routines, is carried out using unique processes to developing the file handling tasks. The process can be done remotely and it's well suited for networks and internet application. The concept of the proposed framework is shown in Figure 3.

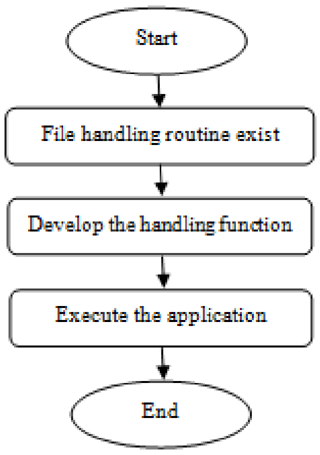


Fig. 3: Method of the Proposed Framework Concept

The feature of the proposed framework:

This framework has the following features:

1. The cover file is the executable file and can be operated even routinely without fear of extraction of the hidden information. This is because the embedded message is already hidden at the end of the file and is no longer subject to manipulation like the executable file. The executable file of this nature can thus be installed on the window.

2. The hiding file is not constrained by size as information of any size can be hidden due to the executable file possessing property of its size being unidentifiable during execution. So also the executable file comes in various types like JDK which contains many sizes of 65MB, 72MB or 77MB that enable any size of the hidden file to be embedded in it and so an attacker cannot guess the actual size of the information hidden. Also, the files is hidden at the end of the executable file which has unlimited space which creates room for any hidden file size.

3. It is very hard to retrieve the hidden information and to know that information is hidden and is this because of the following reasons:
   i. Before hiding the information it is first encrypted using the Twofish system which is a risk-proof technique with a high level of flexibility. Twofish borrows some specifications from other designs like Pseudo-Hadamard Transform (PHT) and from the secure and fast encryption routine (SAFER) family of ciphers related to the earlier block cipher, Blowfish (Reyes *et al*., 2018; Scheier *et al*., 1998).
   ii. The attacker will find it very difficult to know that information is hidden in the executable file because he cannot guess the size of the executable file.
   iii. The hidden information would have to be decrypted after extraction before it can be deciphered.
4. The hidden message can be any one of the following multimedia files types (i.e. text message, audio, video or image) unconstrained by their size. The multimedia can all be hidden at once within the same cover by putting them in one file, compressing them and in turn hiding the compressed file.

Functions of the Proposed Framework:

The detailed functionality of the proposed framework is illustrated in this section. The procedure of the encrypting and hiding operation is shown in Figure 4. This is followed by the flowchart for this operation as shown in Figure 5. However, the procedure of the decrypting and extracting operation is presented in Figure 6. This is followed by the flowchart for this operation as shown in Figure 7.

Input: Any type of secret files, cover file must have (.exe) extension.
Output: Stego file.

- Begin.
- Open the cover file.
- Assign a pointer 1 to the end of cover file.
- Write the secret file name after the pointer 1.
- Assign a pointer 2 behind the hidden file name.
- Encrypt the secret file content by using Twofish technique depending to use the name of the file as a secret key (password).
- Write the encrypted file content beyond the pointer 2.
- End.

Fig. 4: Procedure of Encrypting and Hiding Operation

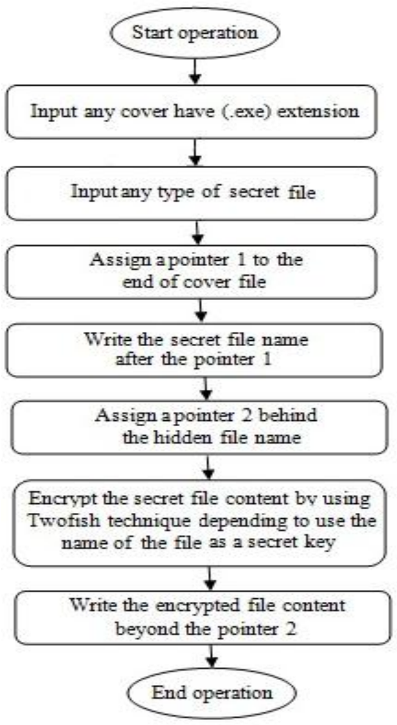

Start operation
Input any cover have (.exe) extension
Input any type of secret file
Assign a pointer 1 to the end of cover file
Write the secret file name after the pointer 1
Assign a pointer 2 behind the hidden file name
Encrypt the secret file content by using Twofish technique depending to use the name of the file as a secret key
Write the encrypted file content beyond the pointer 2
End operation

Fig. 5: Block Flow of Encrypting and Hiding Operation

Input: Stego file that have (.exe) extension.
Output: Secret data.

- Begin (1)
- Select the stego file (.exe)
- Go to the end of (.exe) file.
- If the pointer exist.
- Begin (2) read the secret data name.
- Go to the next pointer
- Read the encrypted hidden data.
- Decrypt the secret data content by using Twofish technique depending on the data name as a secret key (password).
- Create a file using hiding data name.
- Write in to the created file the decrypted data.
- End (2)

Else

- Display a message (no hidden data).
- End (1).

Fig. 6: Procedure of Decrypting and Extracting Operation

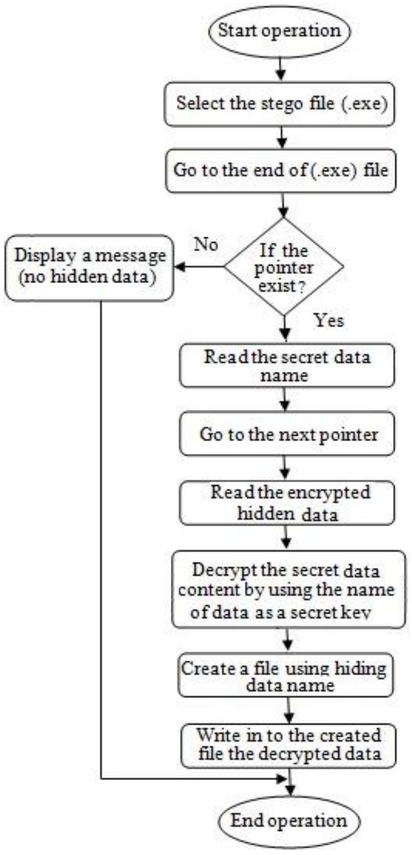

Start operation
Select the stego file (.exe)
Go to the end of (.exe) file
If the pointer exist?
No
Display a message (no hidden data)
Yes
Read the secret data name
Go to the next pointer
Read the encrypted hidden data
Decrypt the secret data content by using the name of data as a secret key
Create a file using hiding data name
Write in to the created file the decrypted data
End operation

Fig. 7: Block Flow of Decrypting and Extracting Operation

The model of information hiding that is proposed in this paper embeds each of the secret file name and the secret data (message) as can be described in details in Figure 8.

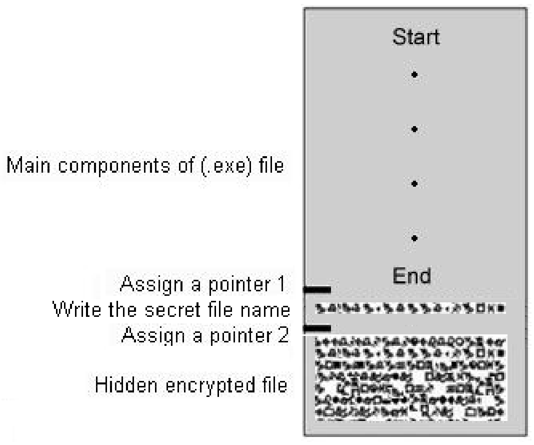


Fig. 8: Model of Hiding Information Beyond the end of (.exe) File

IV. FINDINGS AND DISCUSSION

Test Case Details:

Before we do the test, should be defined the inputs that will be used in this experiment. In fact, there are 12 types of the cover files of different sizes. All of them are in executable format (.exe) as shown in Table 2. This is followed by Table 3 which includes 12 files as a secret data of different sizes.

Table 2: Different Sizes and Types of Cover Files

#	Name of Cover File	Size of Cover File (byte)
1	RealPlayer	818,200
2	Winamp	10,277,728
3	Yahoo! Messenger	14,816,024
4	SmartDraw	59,530,560
5	Adobe Photoshop CS	214,167,816
6	Microsoft Office 2007	305,186,800

Table 3: Different Sizes and Types of Secret Files

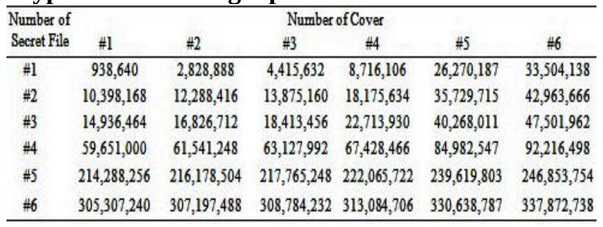
#	Name of Secret File	Type of Secret File	Size of Secret File (byte)
1	Text	Text Document	120,440
2	Image	Bitmap Image	2,010,688
3	Audio	MP3 Audio	3,597,432
4	Video	MPEG Video	7,897,906
5	Compressed	ZIP	25,451,987
6	Executable	Application	32,685,938

Procedure of the Test:

The procedure of the test for the proposed framework is aims to applying 36 test cases, to embedding different types and sizes of secret information within different sizes of the cover files.

The different sizes of stego files that are produced after the encrypting and hiding process as shown in Table 4 as well.

Table 4: Different Sizes of Stego Files After the Encryption and Hiding Operation



Number of Secret File	Number of Cover					
	#1	#2	#3	#4	#5	#6
#1	938,640	2,828,888	4,415,632	8,716,106	26,270,187	33,504,138
#2	10,398,168	12,288,416	13,875,160	18,175,634	35,729,715	42,963,666
#3	14,936,464	16,826,712	18,413,456	22,713,930	40,268,011	47,501,962
#4	59,651,000	61,541,248	63,127,992	67,428,466	84,982,547	92,216,498
#5	214,288,256	216,178,504	217,765,248	222,065,722	239,619,803	246,853,754
#6	305,307,240	307,197,488	308,784,232	313,084,706	330,638,787	337,872,738

* All the sizes of stego files by (byte) unite.

Through the above table, we note the proposed framework is able to accommodate with any size of secret data without any limitation in the hiding process. This feature comes based on the manner of embedding bit by bit sequentially beyond the end of (.exe) files.

V. 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. The challenge of this proposed framework is to conveniently hide information in these files such that it cannot be detected by hackers and others who steal information for detrimental uses. A few points were drawn as conclusion from this work:

- One challenge encountered is the suitability of the cover file size in hiding given information whose size may be larger. However, the hiding technique in this framework ensure an independent relation between the size of the cover file and that of the hidden information.
- The information to be hidden is also encrypted in this framework to further secure the information. There is no constraint to the number or type of files to be hidden by this framework. It can be text message, image, audio, and video.
- The most important beauty of this proposed framework is that the information is buried beyond the end of the executable files where it is virtually impossible for the Antivirus program to detect. Note that Antivirus programs do not read beyond the file.
- The proposed framework overcomes the problem of forgetting and losing the secret key (password) by the users.

REFERENCES

[1] Abdulrazzaq. M. M., H. M. Y. Al-Bayatti and M. A. Fadhil. 2013. A Proposed Technique for Information Hiding in a PE-File. Journal of Advanced Computer Science and Technology Research, Vol.3 No.4, 153-162.

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

[3] Clelland, C. T. R., V. P. and Bancroft. 2007. Hiding Messages in DNA MicroDots. International symposium on Industrial Electronics (ISIE), University of Indonesia, Indonesia, Vol. 1, PP. 315-327.

[4] Daren, P. and M. Scott. 2007. 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.

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

[6] Hamid, A. A. L. M. Kiah, H. T. Madhloom, B. B. Zaidan, A. A. Zaidan. 2009. Novel Approach for High Secure and High Rate Data Hidden in the Image Using Image Texture Analysis. International Journal of Engineering and Technology, Vol. 1, pp. 69-75.

[7] Jalab, H. A., A. A. Zaidan, B. B. Zaidan. 2010. New Design for Information Hiding within Steganography Using Distortion Techniques. International Journal of Engineering and Technology, Vol. 2, No.1, ISSN 1793-8236, pp. 72-77.

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

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

[10] Naji, A. W., A. A. Zaidan and B. B. Zaidan. 2009. Challenges of Hidden Data in the Unused Area Two within Executable Files. Journal of Computer Sciences, ISSN 1549-3636, pp. 890-897.

[11] Namanya, A.P. Awan, J.P. Disso, M. Younas. 2019. Similarity hash based scoring of portable executable files for efficient malware detection in IoT. Future Generation Computer Systems.

[12] Neha and M. Kaur. 2016. Enhanced Security using Hybrid Encryption Algorithm. International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE), Vol. 4, Issue 7.

[14] Neil. F. Johnson. 1995. Steganography. Technical Report, George Mason University. http://www.jjtc.com/stegdoc/sec202.html
[15] Othman, F., A. A. Zaidan, B. B. Zaidan. 2009. New Technique of Hidden Data In PE-File within Unused Area One. International Journal of Computer and Electrical Engineering (IJCEE), Vol. 1, No.5, ISSN 1793-8163, pp. 642-649.
[16] Rachmat. N and Samsuryadi. 2019. Performance Analysis of 256-bit AES Encryption Algorithm on Android Smartphone, Journal of Physics: Conference Series, Conf. Series 1196 (2019) 012049.
[17] Rane. D. D. 2016. Superiority of Twofish over Blowfish. International Journal of scientific research and management (IJSRM), Volume 4, Issue11, Pages 4744 - 4746.
[18] Reyes A. R. L., D. Enrique, Festijo & R. P. Medina. 2018. Blowfish-128: A Modified Blowfish Algorithm That Supports 128-bit Block Size. 8th International Workshop on Computer Science and Engineering (WCSE), Bangkok, pp. 57 8-584.
[19] Saleh. M. E., A. A. Aly, and F. A. Omara. 2016. Data Security Using Cryptography and Steganography Techniques. International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 7, No. 6.
[20] Saleh. M. E., A. A. Aly and F. A. Omara. 2015. Enhancing Pixel Value Difference (PVD) Image Steganography by Using Mobile Phone Keypad (MPK) Coding. International Journal of Computer Science and Security (IJCSS), Volume (9), Issue (2), pp. 397-397.
[21] Scheier, B., J. Kelsey and D. Wagner, C. Hall, N. Ferguson. 1998. Twofish: A. 128-bit Block Cipher.
[22] Shetty. N. P. and N. Ranjan. 2018. Using Steganography & Cryptography to Hide Data in EXE Files. International Journal of Engineering & Technology, 7 (4.41) 9-14.
[23] Smith, J. and B. Comiskey.1996. Modulation and Information Hiding in Images. Proceedings of First Information Hiding Workshop, R. Anderson, Vol. 1174, pp. 207-226, Springer-Verlag, Cambridge.
[24] Tian, Z., Yang, H. 2021. Code fusion information-hiding algorithm based on PE file function migration. J Image Video Proc. 2021, 2. https://doi.org/10.1186/s13640-020-00541-3
[25] Usman. M., I. A. Shoukat, M. S. A. Malik, M. Abid, M. M. Hasan and Z. Khalid. 2018. A Comprehensive Comparison of Symmetric Cryptographic Algorithms by Using Multiple Types of Parameters. International Journal of Computer Science and Network Security (IJCSNS), VOL.18 No.12.
[26] Zaidan, A. A., F. Othman, B. B. Zaidan, R. Z. RAji, A. K. Hassan, A. W Naji. 2009a. Securing Cover-File without Limitation of Hidden Data Size Using Compution between Cryptography and Stenography. Proceedings of the congress on Engineering, Vol. 1, UKE 2009, London, U.K
[27] Zaidan, A. A., A. Majeed and B. B. Zaidan. 2009b. High Security Cover-File of Hidden Data Using Statistical Technique and AES Encryption Algorithm. World Academy of Science, Engineering and Technology 54.
[28] Zaidan, B. B., A. A. Zaidan and F. Othman. 2008. Enhancement of the Amount of Hidden Data and the Quantity of Image. Faculty of Computer Quantity of Image. Quantity of Image. Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia.

Mr. Maysara Mazin AL-Saad, received his B.Sc degree in computer sciences from Baghdad college of economic sciences University, and M.Sc in computer sciences from University Kebangsaan Malaysia. He is currently working as an IT Coordinator in the department of computing and information technology, ministry of education and higher education in the state of Qatar.