Hiding Encrypted Data into Audio File

Hazem Kathem Qattous

Department of Computer Information Systems
Applied Science Private University
Amman, Jordan.

Summary
Communication is one of the most important issues that affect different aspects of our lives. To communicate, it is important to ensure that the channel of such communication is secure. Secure channels are very limited because of their high cost. Encrypting data before transferring it could provide some security. Steganography is also introduced as an alternative way to secure communication by hiding the required data to be transferred into a media file and transferring the cover file itself. This ensures that data will be transmitted under a cover of an innocent media file. This paper introduces an idea of combining both cryptography and steganography to increase the level of security that is required to transfer data. To achieve this, two encryption algorithms, Pohlig-Hellman and One-Time-Pad, are used to encrypt data before hiding it. To hide encrypted data, Least Significant Bit (LSB) technique is used whereby audio wave file (.wav) is used as a cover file to hide data into it. This paper described the development of a software that uses above algorithms and technique to encrypt and hide data into audio wave files. The developed software allows the user to hide data into two different forms, file format and text format. In file format, the software allows the user to hide a file of any type into an audio wave file while, to hide the text format, it allows the user to write a text message before asking the software to encrypt and hide it. To be able to hide a larger data size, a compression algorithm, LZ77, is used to compress the text before encrypting and hiding it.

Keywords
Cryptography . Steganography . Pohlig-Hellman . One-Time-Pad . Least Significant Bit . Text Compression

1. Introduction

The dependency on communication is increasing as new technologies and techniques are developed. Simultaneously, techniques to provide the privacy and security for communication are also being developed. Different authors, such as [17], justify the development of the security of communication by the development of communication techniques themselves. The same notes are introduced by [3] who adds that security in communication is one of the most important issues as different people communicate with each other and the communication technologies are developed tremendously.

When talking about security, especially in communications, two terms come to mind, the first is cryptography and the second is steganography. Cryptography is the science of securing messages [15]. It uses the encryption of data using a method which will not allow anyone who catches the message to decrypt it unless s/he knows the correct method to do that [18]. So, encrypting a message and sending it through a communication channel could provide a good level of security. The second method of securing digital communication that comes to mind is steganography. [18] defines steganography as the process of hiding one communication medium into another. Steganography usages extends to include different applications such as watermarking and digital rights and annotation in addition to other applications. One of the most important applications for steganography is covert communications [4]. This paper concentrates on the latest application of steganography.

This paper deals mainly with the problem of the ability to provide a suitable method to prepare data to be transferred securely using the two security techniques, cryptography and steganography. It investigates suitable manners to take the benefits of both of them to allow two parts to communicate with each other securely through insecure communication media like Internet. Internet is a medium rife with many listeners and it is difficult to ensure that communication through it is secure. This research tires to find a way using the two security techniques to facilitate communication through the Internet. The paper introduces a software that combines both methods to increase security of communication between two parties using the Internet. The software allows the user to encrypt and hide data into an audio (.wav) file. It also provides the user with the ability to write a text message and prepare it to be transferred securely after compressing the text to reduce its size. The software also allows the user to retrieve the hidden data.

Combining of cryptography and steganography is suggested by many authors such as in [18] and [3]. They say that cryptography and steganography could be combined by encrypting the message then hiding it into a cover file. Although the idea has been discussed and suggested previously, research still active in this field as different encryption algorithms used with different steganography methods leads to different results regarding security, memory requirements, time, and memory and covert file space required for applying the techniques.

Manuscript received June 5, 2017
Manuscript revised June 20, 2017

2. Literature Review and Research Justifications

2.1 Steganography

Steganography, as a term, comes from two Greek words, steganos, which means the covered or secret and graphy, which means writing or drawing. So, steganography could be defined as the covered, hidden or secret writing. [18] specifies the definition of steganography as the process of hiding a medium of communication, which could be text, sound or image into another medium. Steganography was anciently used by first recorded in the histories of Herodotus were. Greek text messages were written on wax tablets which were covered with another layer of wax to cover the text. During World War II, steganography was used by hiding a message text within another message. This was achieved by distributing the letters of the secret message between the words of the cover message. Invisible ink that was used to communicate agents in occupied Europe during the war is considered also as a type of steganography into a specific order that allowed its extraction later on [18].
There are several applications of steganography that include securing a message transfer between two authorized entities imperceptibly. The two entities may require to communicate secretly for different reasons such as businesspeople for protecting secrets of companies, governments or even people may communicate secretly for fun [7] and [3]. Another application of steganography is digital watermarking for the purpose of copyrighting. Digital media files required to be protected, such as images and sound files, are fingerprinted by hiding digital watermarks are hidden into them for copyright protection. Secrete data can be protected from viewing, robbing or damaging by maintain it hidden into other innocent looking files [7]. Through this steganography application, some companies can protect their data by hiding it into cover files [3]. What can be noticed here is that many authors including [7] and [3] introduced the application of communication between different parties for the purpose of transferring secure messages as the first and the main application of steganography. [9] introduced that the main goal of steganography is to transfer a message and avoid drawing suspicion to its transmission. This justifies the investigation of this application of steganography in this research and its attached software discussed in this paper. So, the software uses steganography to hide data, whatever its format, within a cover or host file to prepare this file to be sent over a communication media such as the Internet. This application of steganography, as stated above, will provide the user with the ability to communicate with others securely and transfer secret messages within an innocent unsuspicious-looking file.

2.1.1 Media Used in Steganography

Through reviewing different steganographic science research, it is clear that digital media files such as images, audio and video files are used to hide data. [3] encourages hiding data into image and audio files as these types of files could be modified without noticeable perceptual effect on their quality. Data is hidden within audio files using the bits that represent sounds which are not audible to human ears. For images, data could be hidden using redundant bits in uncompressed files.
[4] introduce the advantages of using the steganography technique called “Least Significant Bit (LSB)”. One of its main advantages is the huge amount of data that could be embedded within audio files using this methodology. Through their paper, they concentrate mainly on the amount of data and the ability to increase the bit depth within multimedia files. They depend in their choice on that combining LSB methodology with audio files of amplitude resolution of 16 bits per sample achieves the required large amount of data size hiding. They say “Data hiding in the least significant bits of audio samples in the time domain is one of the simplest algorithms with very high data rate of additional information”.
Consequently, this research adopts the use of audio files and the use of LSB technique for steganography. Wave (.wav) audio file type was selected to be used as a cover file for the purpose of hiding data. Advantages of this type of file include high data redundancy which allows higher data capacity to be hidden in the file which makes it suitable to apply LSB that depends on redundancy for hiding data. The high data redundancy came from the fact that wave file format is not subjected to any type of compression. In addition, wave files can exist in a 16-bit sample audio file so it achieves the requirement that [4] introduces for providing high data hiding capacity. One more advantage is that wave files can be generated easily using the sound recorder of Windows OS and can be transferred through the Internet either directly by email or by publishing or sharing it on a website.

2.1.2 Audio Wave File Format

Since the audio wave file is decided to be used as the cover file in this research and its attached software, its structure should be studied so that the internal construction of this file can be understood. Wave file structure can be summarized in .

Table 1: Audio wave file structure

Chunks and Layout	***Description***
“RIFF”	char[4], indicates the starts of the RIFF header.
Size of rest of file	32 bits, describes FileSize - 8

“WAVE”	char[4], describes data format.
“fmt “	char[4], start of describing the format specifically.
WAVEFORMATEX Size	Wave format data size.
WaveFormatTag	16 bits, always PCM.
Number of Channels	16 bits, 1 = mono, 2 = stereo.
Number of Samples Per Second	32 bits, samples rate in samples/sec.
Number of average Bytes per Second	32 bits, Bytes rate in bytes/sec.
Number of Block Align	16 bits, specifies number of blocks aligned in byte.
Bits per Sample	16 bits, specifies number of bits/sample.
“data”	char[4], indicates the start of data chunk.
Data Size	32 bits, specifies the data size
Data	The actual sound data

Adapted from http://soundfile.sapp.org/doc/WaveFormat/.

Wave file type adopts little-endian byte ordering, which the default byte is ordering in wave files [8]. The following example explains this ordering. If a number in hexadecimal is (OX 10 00). If this number is transferred to bits for binary representation, it will be (0001 0000 0000 0000) and this number, in decimal representation is (4096) which is wrong because the number is in decimal (16). This is because of the byte ordering, also called “Intel 80x86 format”. In this ordering the final byte of the number is placed at first and the first is placed at the end. Figure bellow () clarifies the least significant bytes and bits into a wave file.

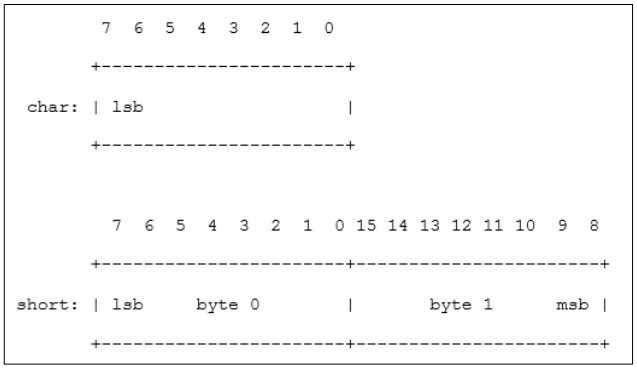

Fig. 1: Default little-endian byte ordering of wave files

The second part of the figure above () shows the reverse order of the little-endian byte ordering of wave files and shows the least significant bytes and bits. Understanding the ordering of the bytes and bits in wave files is important of the purpose of software implementation.

2.1.3 Least Significant Bits

[18] and [13] describe Least Significant Bits (LSB) methodology as the simplest for hiding information within a file. The technique depends on replacement of the required to be hidden bits with the least significant bits of an image pixels or audio file samples. In other words, the least significant bits of a cover file (host file) will be replaced by the bits of the message (could be a file) to be hidden (guest file or guest data). Of course, some pixels or audio samples will be left without change to prevent host file format corruption such as the file header.

[13] introduce another method of bits replacement but this time, only a subset of the pixels or samples of audio files will be replaced by the hidden message. This increases the strengths of the hidden data and reduces the chance of its discovery. Unfortunately, this methodology has a major disadvantage that is reducing the capacity of data that can be hidden in a file since only some of the carrier file bits will be used for hiding data. However, this technique is suitable in watermarking where few bits only are required to be hidden.

"Transform Domain Technique" is another steganographic algorithm that depends on embedding a mark into the transform domain of the host file. However, it is much more complicated than LSB technique and used by compression algorithms. Another famous algorithm that is used in steganography is "Patchwork Algorithm". It works by randomly selecting pixels in an image and hide data into them in a way that makes brighter pixels brighter and darker pixels darker. In audio files it selects random audio samples and increases their amplitude contrast. Later on, filters are applied on the files to minimize the noise. "Spread-Spectrum Techniques" are also used for hiding data in audio files by adding a maximal length sequence to the signal in the spatial domain. The last two techniques work perfectly for watermarking as they hide only small amount of data but not suitable for the purpose of hiding large amount of data. Accordingly, they were not used in purpose of this research and its attached software [13].

2.1.4 Tools

[12] introduces a software tool that is used for hiding data within WAV file and then compresses the result into Mp3 files. The software name is Mp3Stego. Another tool that is used to hide data within a wave file is introduced with explanation to the wave file structure by [8]. [8] introduces the tool with the advantages of hiding data into wave file as that the file size will not change, high bit rate of hiding and ease of wave file generation.

[20] introduces the difference between the tools of steganography and those of steganalysis. The first is used to hide data and the second is to discover this hidden data. Steganalysis tools work depends mainly on discovering the message and “simply” destroying it to prevent it from

reaching its destination. [3] claims that through an experiment he could discover many hidden data fragments within images published on ebay's website. This shows that investigating cover files and destruction of the messages should be done before publishing the cover file itself on a website. However, this may need extensive work and delay, which may affect the business itself.

2.2 Cryptography

[15] defines cryptography as "the art and science of keeping messages secure". [3] considers cryptography as the only technology that is used for network security and there are several solutions under this technology.

Cryptography is old as Julius Caesar used it to encrypt his communication regarding political and military secrets and his way of encryption is called "Caesar cipher" which is also referred to as "rotation scheme" [3]. This way depends on replacing each character with another on a regular base. So, if the regular base is one character forward, then character 'a' will be 'b' and 'b' will be 'c' and so on. [18] differentiates between steganography and cryptography. The main difference point is that in steganography, the data is not touched or modified. On the contrast, in cryptography, the data is modified and transformed into another form or format to prevent its understanding unless it is decrypted using specific manner and keys. The generated encrypted data is called "cipher text". He introduced the idea of combining both techniques to increase the security in transferring secrete messages.

2.2.1 Cryptography Techniques and Algorithms

2.2.1.1. Symmetric

Symmetric encryption, sometimes called "single-key encryption" depends on the same key for encryption and decryption [3]. The sender and receiver agree on one key or secret keys for encryption and decryption and this should be done before they start communicating using encrypted messages. According to [3] this enforces the two communicating parties to find a secure channel to transfer the keys first before starting sending and receiving messages. The existence of such channel eliminates the need for encryption [3]. It is believed that steganography also can be used as a technique for exchanging a key to be later used in encrypting messages between two parties.

Symmetric cryptography has some schemes such as substitution, permutation and XOR. [16] and [15] introduce more schemes; some of them are important as Vigenere cipher and One-Time Pad.

2.2.1.1.1.Substitution

Substitution depends mainly on replacing one letter with another. This methodology is weak because the same value for each letter is used each time the letter is encrypted which allows breaking the algorithm depending on statistical background of English letters histograms [3] and [16].

2.2.1.1.2Permutation

In contrast to substitution, in permutation the message letters are not replaced or mapped to any other letters. Instead, message letters are moved from their places into other positions. This method depends on a key that specifies the new position of each letter to help in decrypting the cipher text. This methodology is also considered as straightforward and could be broken easily [3], [16] and [15].

2.2.1.1.3 One-Time Pad

This encryption scheme is also called Vernam encryption. It depends on large non-repeating set of random letters to encrypt the message [15]. The sender encrypts the message using the key and destroys that key to ensure that it will not be used again. The receiver decrypts the cipher text using the same key and destroys it as well. This ensures that the key is only used once for only one message. Using the same key again leads to overlapping and the algorithm can be broken [15]. [14] says that this methodology cannot be broken and it is highly secure if it is used correctly.

An example adopted from [19] of such encryption technique is as follows:

H E L L O
9 0 2 1 0
Q E N M O

The first line in the above example is the message and the second line is the key. When both are added to each other the third line, which represents the cipher text, is produced. However, [19] considers the use of One-Time-Pad as impractical because of the need to exchange the One-Time-Pad (the key) securely. [15] concentrates on another disadvantage of this methodology which is the difficulty to produce the random generation for the key as it needs gaga bytes of keys.

2.2.1.1.4 Vigenere Cipher

Vigenere cipher is considered as an example of polyalphabetic substitution ciphers where multiple substitution keys are used [15]. The following is an example adopted from [16] to clarify the polyalphabetic substitution.

"Plaintext alphabet A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Cipher alphabet one T M K G O Y D S I P E L U A V C R J W X Z N H B Q F
Cipher alphabet two D C B A H G F E M L K J I Z Y Z W V U T S R Q P O N"

If a plaintext is to be encrypted, the first letter is substituted by the mapped letter from the first line of cipher and the

second letter is substituted by the mapped letter from the second line. Of course more cipher alphabet keys could be used, which increases the probability of the key to reach (26!) to the power of the number of keys. But in this case the number of the keys that are needed to be remembered by the users also goes up [16].
The main advantage of such method is that it breaks the link between the plaintext and the cipher text, so this method limits the ability of cryptanalysts to use the statistical characteristics of language to break the encryption. This is done by substituting the same letter of the language each time with different letter. As an example, HELLO could be ciphered using the above key as SHLJV. What could be noticed is that the first L is substituted by L while the second one is substituted by J, [16].
Vigenere cipher introduces a new way of encryption depending on the principles of the above method and handles the problem of the users' need to remember the keys. Vigenere cipher depends on producing a key of a number of letters and using this key for simple substitution for the message letters. If "SESAME" is used as a key then the following example, which is adopted from [16] shows the Vigenere ciphering method.
THISISATESTMESSAGE
SESAMESESAMESESAME
The result will be "LLASUWSXWSFQWWKASI". This also shows again that the letter 'A' was replaced by two different letters in the cipher text.
[15] says that such methods of encryption could be broken easily if the length of the key is discovered and he calls this key length as "period".

2.2.1.2 Asymmetric

Asymmetric algorithms are characterized by using two keys instead of one as in the case of symmetric. The first key is a public key which is used for encryption, and the second is a private key which is used for decryption. There are several asymmetric algorithms that are used extensively such as the RSA algorithm which is one of the most powerful asymmetric algorithms [15].

2.2.1.2.1 RSA

RSA is as [15] describes it "…the easiest to understand and implement" and it is the most popular one. The RSA power and security comes from the difficulty of factoring large numbers to use RSA, public and private keys of 100 to 200 digits prime numbers are required. [19] agrees with [15] that RSA is the most popular between other algorithms and its power comes from choosing large numbers (1024 to 2048 bits) to generate the public and private keys.

2.2.1.3Hash

Hash algorithms are one way transformation which means that the plaintext is encrypted but cannot be decrypted. Its input is the plaintext with any size; which is converted and comes out as a fixed length output. This output is irreversible and cannot be decrypted to retrieve the plaintext again. This type of algorithms is used and useful in maintaining and keeping passwords and digital signatures [3].

2.2.2 Investigating Suitable Encryption Algorithms

For the purpose of this research and its attached software implementation, there is a need for a quick, easy to implement, suitable size key, and secure algorithm. The size of the key matters here because it should not take space from that available to hide the message itself.
Two algorithms were selected as candidates, Vigenere Cipher algorithm and One-Time Pad. Vigenere algorithm was chosen because it handles efficiently the weakness in the substitution and permutation algorithms as discussed by [16]. However, a weakness including use of one key of a specific length to encrypt the whole message, which may lead to break the algorithm, is still valid. So, One-Time-Pad appeared as an alternative. The main disadvantage of this algorithm, which is the need to exchange the pad between the sender and receiver, is neglected because this is solved by the combination between the cryptography and steganography, and this is the advantage that cryptography takes from this combination. So, the key will be hidden and sent with the message. Unfortunately, this algorithm violates one of the third requirement stated above because the message needs a key equivalent to its size [15], [16] and [19].
[19] states that there are some trials to improve this algorithm and eliminate its disadvantages. One of the introduced ideas is that the same key segment is used to encrypt the whole message. This concludes that there is a way to overcome the disadvantages of this strong algorithm. It is believed that this modification on One-Time-Pad is just a combination between it and Vigenere algorithm. [19] introduces the combination between the two algorithms, One-Time-Pad and Vigenere algorithm, and called the result as One-Time-Pad algorithm. This research will call the modified algorithm as One Time Pad as well.
Thinking again in the modified algorithm introduced by [19] leads to the fact that the only weakness in it is that it repeats the same key over the message which allows a cryptanalyst to discover the key length (the period). After that, discovering the key itself and decrypting the message is just a matter of time.

2.2.2.1 Modified One Time Pad Algorithm

To overcome the above discussed disadvantage, it was decided to perform a personal modification on the algorithm that could be a solution for the problem. The proposed solution depends on providing more than one key, as an example three, of different lengths and using these keys together to encrypt the message. Applying the algorithm

could be in the form of using the first key to encrypt, as an example, 10 bytes of the message then using the next key to encrypt the next 10 bytes of the message and the third key as well. In the next period, the second key will be used to encrypt the next 10 bytes of the message then the third key and finally the first one. In the third period, the third key will be used first.

From the first look, using this algorithm makes discovering the key length based on repeated patterns, as [16] discusses, something very hard because there are more than one key of different lengths and the keys are used in different order each time (period). Consequently, this will reduce the chance of breaking the algorithm and decrypting the original message using the expectation of the key length. One drawback of such algorithm for this research is the need for more space to store a key of three parts (three keys) instead of a small one key. But on the other hand, if one key is used then this key is needed to be long to increase the security. This, the long key, is one of the bases that [15] puts as a requirement for a secure symmetric algorithm. From this point of view, using three short keys instead of long one could be considered as fair.

The proposed encryption algorithm was not found in any reference during the investigation. This does not mean it has not been suggested before; however it could be true that this is the first time that such a modification is suggested. It is believed after detailed investigation that this algorithm solves the problems and disadvantages that are generated from the combination between One-Time-Pad and Vigenere algorithms.

Other algorithms were investigated through the research and one of the most interesting algorithms was Pohlig-Hellman algorithm. [15] introduces this algorithm and comments that it is "similar to RSA". This concludes that the security of this algorithm is similar to the RSA one. According to [5], Pohlig-Hellman algorithm was introduced before the RSA algorithm. Pohlig-Hellman algorithm is not considered as symmetric algorithm because two different keys are used in encryption and decryption. However it is not a public key algorithm as well because its keys should stay secret, otherwise they can be derived [15].

Pohlig-Hellman algorithm has two main advantages over the RSA. The first is that it solves the RSA problem of the need for large numbers to be hidden within the wave file as keys. Instead, small keys are used and can be hidden with smaller space. Secondly, when the key of Pohlig-Hellman is hidden with the encrypted cipher within a wave file, this action allows taking the full advantage of combining cryptography with steganography. That is the similar case of using the One Time Pad algorithm discussed before. Consequently, using Pohlig-Hellman is considered as an advantage as it takes the full benefits from the combination between cryptography and steganography.

As a conclusion, One-Time-Pad and Pohlig-Hellman encryption algorithms are selected to be used in this research and its attached software.

2.3 Compression

Data compression can be defined as the production of output string from combining an input string with a model. This output is usually a compressed version, smaller in size than the input string and is called the coded string [21]. [19] justifies the importance of using compression when data is to be hidden as:

- Low size data is easier to manipulate and handle
- Compressed data looks like another thing other than the original data, which can be considered as encryption.

Both points are important for this research as there is a need to hide more data into the cover file and transforming the form of data from the original one can help in securing the message. This encourages compression investigation in this research and its usage in the attached software. Throughout this research, compression of different file types has been reviewed. However, it has been discovered that each file type requires a compression algorithm to achieve an optimized compression. Accordingly, it has been decided that compression will only be performed on the text that the attached software allows the user to enter as a message.

2.3.1 Compression Techniques

[10] defines two main types for compression, lossless and lossy techniques. In lossless technique, if the compression operation is inversed, the original data will be retrieved exactly as it was. This technique is used to compress text and other discrete data, computer-generated data, and some images and video data types [10]. [1] also document that lossless compression techniques are used for the purpose of text compression. [6] justifies the need for lossless techniques to be used with text as losing any character during the compression and decompression processes may lead to a misleading text meaning. In the contrast, lossy techniques, when decompression of data is performed, the output is data that is similar to the original but not exactly the same [10]. This technique can be applied to compress voice data [10] or images and video data [19].

Based on the above and previous conclusions and discussions, it is important to emphasize again that text is only compressed using lossless techniques. As the research concentrates only on text compression, therefore it will focus on lossless compression techniques only.

For this research, the adopted compression algorithm should be simple in implementation, quick and has a high compression rate. Several algorithms have been investigated and some of these algorithms are the following:

2.3.1.1 Run Length Encoding (RLE)

RLE is a compression algorithm that performs compression depending on the redundancy of characters. It can compress redundant characters into only three characters. The following example, which is adopted from [6] shows the work of this algorithm:
Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx that's all, folks!
The above text is compressed by RLE to: <ESC>X<31> that's all, folks!
The limitation of using such algorithm is that it works fine if there is a redundancy of more than three characters following each other [6] which is very rare in a text message.

2.3.1.2 Huffman Coding

It is an efficient technique for compressing data [6]. However, it is not suitable for this research because it depends on building a tree data structure for compression and the same structure is required for decompression. So, the tree structure needs to be transferred with the compressed message which will take a space in the cover file that should be left to hide the message itself. Arithmetic Coding algorithm works in a similar manner as it is a development of Huffman Coding algorithm [11].

2.3.1.3. LZ-77 Encoding

This algorithm depends on the redundant nature of the text. Its work depends on discovering the repetitions in the text and replaces the repeated text with pointers for the first occurrence of the repetition [6]. LZ-77 has been used in combination with steganography for watermarking purposes by [2]. They used LZ-77 to hide information for authentication purposes in compressed text documents.
It operates using a buffer that could be called a "sliding window" [6] which stores the text that has been read recently. A pointer is used to perform the reference to the uncompressed text. The size of the buffer is very important; if it is too large, the pointers will be too large which will need extra space to hid it, and if it is too small, the chance to find similarity in strings is less [6].
To show the algorithm while it is working, an example that is adopted from [6] is discussed below.
If the text to be compressed: "the_rain_in_Spain_falls_mainly_in_the_plain"
where (_) means a space.
The algorithm searches for similarities between characters and finds the first similar words in the text (in_) in (rain_) and (in_). The algorithm will replace the (in_) to <3, 3>, so the compressed text will be:
the_rain_<3, 3>…….
The first number represents the number of characters to which the pointer should return back and is called the offset. So the pointer returns back to the (i) in (rain_). The second number represents the number of characters that are similar between the reference and the original text. This represents (in_). By this reference the original text could be decompressed.
The algorithm continues and the final results looks like:
The_rain_<3,3>Sp<9, 4>falls_m<11, 3>ly_<16, 3><34, 4>pl<15, 3>.
From the work of LZ-77, it is clear that its work does not depend on any type of tables or data structures. This makes it suitable for this research as it needs no more space in the cover file to hide any more information than the compressed message itself. However, to be able to adopt this algorithm and apply it practically, some adaptations should be taken into consideration.
First of all, the reference needs space and its space should be reduced to the lowest size. To achieve that, a special character (#) is used to flag the reference. This flag means that the following is a reference and it is not part of the text. So, this mark is followed by two bytes, each represents a number of the reference numbers (the offset and the number of compressed characters). To differentiate between the offset and the number of compressed characters, another flag (~) is used between them. This is to prevent any mistakes that could be generated at decompression time. Noticing the previous action, it gives a byte for each number. This allows the buffer size (the offset) to be 256 characters, which is considered to be a good offset or size for a buffer to give the chance to find similarities between texts and, at the same time, does not take more space than 1 byte. Another thing that should be taken into consideration is that the reference marks and the two numbers occupy 4 bytes. This means that the reference will not be used unless the length of similar characters is bigger than or equal to 5. If it is less than 5, say 3 as an example, the algorithm is using 4 bytes to reference 3 bytes, which works against compression. These were the most important practical issues that were taken into consideration when the algorithm was implemented.

3. The Implemented Software

The figure below () shows the GUI of the implemented software. To start the process of hiding, the user selects the cover file (to hide data into it) by pressing on the browse button for "File to Hide In". A file chooser papers that allows the user to select a wave file. The software presents for the user some information about the selected wave file in "File Description" window. The user then selects the destination location where the new generated wave file, after hiding data, will be saved. Note that the system shows two buttons "Hide File" and "Hide Text" as disabled until the user enters the cover file and the stego (destination) file locations. At this point, the system enables the two buttons, "Hide File" and "Hide Text". Based on the button the user selects, the system enables the require fields in the GUI and disable the other button.
If the user decides to hide a file, the system enables the "Hide File" area and the user selects, through a file chooser,

the required file to be hidden. Then the user selects to decrypt the file either using Pohlig-Hellman or One Time Pad algorithm as shown in ().
If the user decides to hide text, s/he fills the message text in the area under "Insert Text to be Hidden" label. In this case, the "Hide File" area will be disabled. After finishing the text message, the user selects the required encryption algorithm, either Pohlig-Hellman or One Time Pad. The user then presses "Hide" button either under the "Hide File" area or under "Hide Text" area depending on what s/he is hiding ().

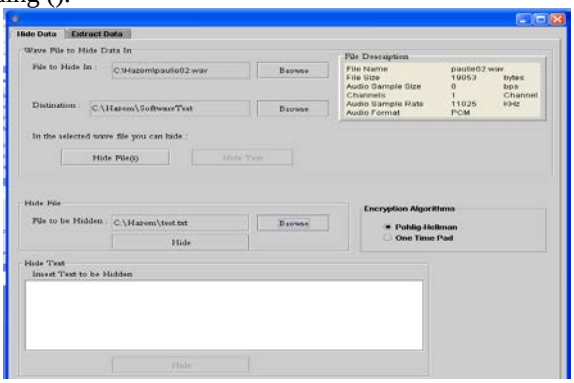

Fig. 2 Software running after selecting the wave file to hide data in and the file to be hidden.

At this point, the software will open the file to be hidden (guest file), create another copy of it and encrypt its content using the selected encryption technique. Similarly, a new version of the cover wave file will be generated. The encrypted file will be hidden within the new generated file. The software in its current version hides two bits in each audio sample. Finally, the cover wave file will be saved in the selected destination folder and the software shows a confirmation message to confirm that the process is completed and the cover file is ready to be used. Hiding text message will be done in the same way except that the text will be compressed before hiding it. In case the size of the guest file or the text message is bigger than the available space in the cover file, the system will not complete the process and will show a warning message to the user.

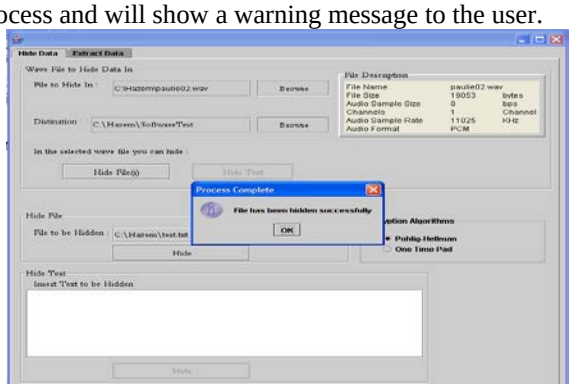

Fig. 3 Software running after performing file hiding process.

When the cover file reached safely to the other side of communication, the original file can be extracted using the same software ().
The user selects the "Extract Data" tab then browses and selects the cover file location. The user then selects the destination folder where the original file should be saved at and presses "Extract" button. The original file will be extracted, decrypted and saved in the destination folder.
If the stego file contains text data, the software will extract the data from the cover file, decrypt it, decompresses it and presents the text message under the "Extracted Text" area. In both cases, the software shows a confirmation message telling the user that the extraction has been done successfully ().

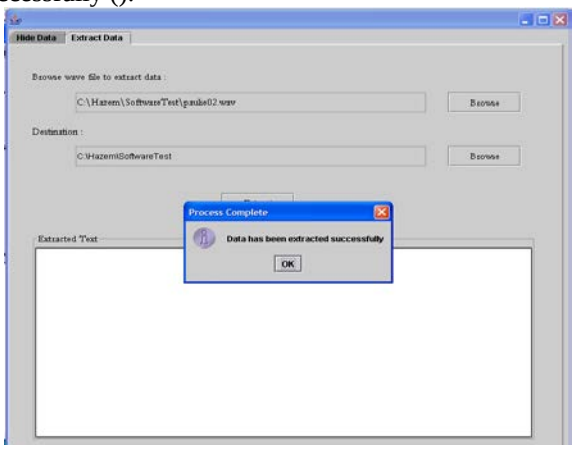

Fig. 4 Software running after performing data extraction process.

4. Conclusion

This paper introduces a software system that implements the idea of combining the two security techniques, cryptography and steganography, together. This combination allows each of the two techniques to take the advantage of the other in increasing the security of data to be transferred over an insecure communication channel. The paper reviews the literature to find a suitable cover file type to hide data into it through steganography. Wave audio file was selected because of the available size space for hiding data without noticeable effect on the file quality. Least Significant Bit (LSB) steganography technique is selected to be applied between all reviewed techniques because of its implementation simplicity and suitability for the research.
The paper introduces comparison of different cryptography techniques to choose between them for the purpose of implementation. Two basic encryption algorithms, Pohlig-Hellman and One Time Pad, are selected as they are suitable to be applied in this research and implemented in the attached software. A modification was applied to the One Time Pad algorithm to increase its encryption security.

The implemented software allows the user to hide data in two forms, either file or text. In case of text hiding, the software implements a compression algorithm, LZ77, which is selected to be implemented between other reviewed compression algorithms because of its suitability for text compression and this research. The user of the software hides data through selecting the cover file, selecting the destination to save the stego file, selecting the preferred encryption algorithm and finally to select to hide another file or text. If the user selects to hide a file, s/he should select the required file to be hidden (guest file). In case of selecting hiding text message, the user should write the required message. The software gives messages indicating the success of the process or failure because of the low capacity of the cover file. The software also allows the user to retrieve the hidden file or text by reversing the process of hiding. The hidden file or text will be decrypted and, in case of text, it will be decompressed.

Acknowledgement

The author is grateful to the Applied Science Private University, Amman, Jordan, for the full financial support granted to this research.

References

[1] Abel, J., and Teahan, W., 2005. Universal Text Preprocessing for Data Compression. IEEE Transactions on Computers, 54 (5), pp. 497 – 507.

[2] Atallah, M., and Lonardi, S., 2003. Authentication of LZ-77 Compressed Data. Proceedings of the 18th ACM Symposium on Applied Computing, Melbourne, Florida, pp. 282-287.

[3] Cole, E., 2003. Hiding in plain sight: Steganography and the art of covert communication. Indianapolis : Wiley.

[4] Cvejic, N. and Seppanen, T., 2004. Increasing robustness of LSB audio steganography using a novel embedding method. Proceedings of the international conference on information technology: coding and computing.

[5] Flinn, P., and Jordan, J., 1997. Using the RSA Algorithm for Encryption and Digital Signatures: Can You Encrypt, Decrypt, Sign and Verify without Infringing the RSA Patent? [online]. Available from http://www.cryer.co.uk/glossary/r/rsa/The_RSA_Algorithm_and_Patent.html. [Accessed 20 June 2017].

[6] Goebel, G., 2005. Introduction / Lossless data compression [online]. Available from: http://craymond.no-ip.info/awk/ttdcmp1.html. [Accessed 20 June 2017].

[7] Jiruwala, F., 2004. Moment preserving data hiding: Analysis and experiments [online]. Available from http://www.ece.stevens-tech.edu/~mouli/FatemaThesis.pdf. [Accessed 28 January 2006].

[8] John, C., 2004. Steganography VIII - Hiding Data in Wave Audio Files [online]. Available from https://www.codeproject.com/Articles/6960/Steganography-VIII-Hiding-Data-in-Wave-Audio-Files. [Accessed 20 June 2017].

[9] Johnson, N. and Jajodia, S., 1998. Steganalysis: The investigation of hidden information. IEEE information Technology Conference, Syracuse, New York, USA, September 1998.

[10] Martínez, H., and Moreno, J., 1998. Evolution of Cellular Automata for Digital Image Compression. Proceedings World Multiconference on Systemics, Cybernetics and Informatics SCI'98, 2, pp.452-456.

[11] Moffat, A. and Isal, R., 2005. Word-based text compression using the Burrows–Wheeler transform. Information Processing and Management 41, pp. 1175–1192.

[12] Petitcolas, F., 2006. mp3stego [online]. Available from http://www.petitcolas.net/fabien/steganography/mp3stego/. [Accessed 20 June 2017].

[13] Petitcolas, F., Anderson, R., and Kuhn, M., 1998. Attacks on copyright marking systems. David Aucsmith, Ed., Second workshop on information hiding in Vol. 1525 of Lecture Notes in Computer Science, Portland, Oregon, USA, 14-17 April, 1998, pp. 218-238.

[14] Ranum, M., 1995. One-Time-Pad (Vernam's Cipher) Frequently Asked Questions [online]. Available from http://www.ranum.com/security/computer_security/papers/otp-faq/. [Accessed 20 June 2017].

[15] Schneier, B., 1996. Applied cryptography, second edition: protocols, algorithms, and source code in C. USA : John Wiley & Sons, Inc.

[16] Smart, N., 2003. Cryptography: An introduction. Published by UK : McGraw-Hill Education.

[17] Smith, R., 1996. Data encryption standard [online]. Available from http://rae.falkor.gen.nz/DES.html. [Accessed 20 June 2017].

[18] Watkins, J., 2001. Steganography – Messages hidden in bits [online]. Available from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.8901. [Accessed 20 June 2017].

[19] Wayner, P., 2002. Disappearing cryptography Information hiding: Steganography & Watermarking, 2nd edition. Published by USA : Morgan Kaufmann.

[20] Westphal, K., 2003. Steganography revealed [online]. Available from https://www.symantec.com/connect/articles/steganography-revealed. [Accessed 20 June 2017].

[21] Witten, I., Neal, R., and Cleary, J., 1987. Arithmetic coding for data compression. Communications of the ACM, 30 (6), pp. 520 - 540.

Hazem Qattous is currently an associate professor at Applied Science Private University in Amman, Jordan. Dr. Qattous holds a Ph.D. degree in Computing from University of Glasgow, UK. His research interests are in Human-Computer Interaction.