Al-Khwarizmi Engineering Journal, Vol. 8, No. 1, PP 27 - 36 (2012)

Al-Khwarizmi Engineering Journal

A New Audio Steganography System Based on Auto-Key Generator

Inas Jawad Kadhim

Department of Electric Power Engineering Techniques/College of Electric & Electronic Techniques

Email: inascnn95@yahoo.com

(Received 16 January 2011; accepted 18 0ctober 2011)

Abstract

Stenography is the art of hiding the very presence of communication by embedding secret message into innocuous looking cover document, such as digital image, videos, sound files, and other computer files that contain perceptually irrelevant or redundant information as covers or carriers to hide secret messages.

In this paper, a new Least Significant Bit (LSB) nonsequential embedding technique in wave audio files is introduced. To support the immunity of proposed hiding system, and in order to recover some weak aspect inherent with the pure implementation of stego-systems, some auxiliary processes were suggested and investigated including the use of hidden text jumping process and stream ciphering algorithm. Besides, the suggested system used self crypto-hiding pseudo random key generator. The auto-key generator has purposes to investigate the encryption and embedding processes .The hiding results shows no noise in the stego-wave file after embedding process, also no difference in size is found between the original wave audio file and stego-wave file.

Keywords: *Audio steganography, Text hiding, (LSB) technique.*

1. Introduction

Steganography, from the Greek, means covered, or secret writing, and is a long-practiced form of hiding information. Although related to cryptography, they are not the same. Steganography's intent is to hide the existence of the message, while cryptography scrambles a message so that it cannot be understood.

Steganography includes a vast array of techniques for hiding messages in a variety of media. Among these methods are invisible inks, microdots, digital signatures, covert channels and spread-spectrum communications. Today, thanks to modern technology, steganography is used on text, images, sound, signals, and more [1].

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication [2].

This paper introduces a new steganography system based on LSB technique using non-compressed wave audio file. The embedding text can be enciphered then embedded using the same bytes. Of the wave file, as an encryption and hiding keys, of course the two used bytes are different from each others to increase the complexity of the hiding algorithm. The hiding results show no noise that can be heard in the stego-wave file after embedding process.

The proposed steganography system was chosen because the wave files is common in used in all communication means like, internet, mobiles, computers etc. Since the wave file has huge data, it is considered a good container of a big message. When random LSB technique is used, it will be hard to detect or broken. The proposed hiding algorithm can be considered a simple idea for encryption and hiding, but not the best efficient method for hiding.

2. Steganography

The word Steganography comes from the Greek Steganos (covered or secret) and Graphy (writing or drawing) and it means literally covered writing. Cover is an input to the stego-system, in which the embedded will be hidden. The possible cover carriers are innocent looking carriers (images, audio, video, text, or some other digitally representative code) which will hold the hidden information. Embedded is something to be hidden in the cover. A Message is the information hidden and may be plaintext, ciphertext, images, or anything that can be embedded into a bit stream. Embedding is the process of hiding the embedded message. Stego is the output from the stego-system and is something that has the embedded message hidden in it. Together the cover carrier and the embedded message create a stego-carrier. Hiding information may require a Stegokey which is additional secret information, such as a password, required for embedding the information. For example, when a secret message is hidden within a cover image, the resulting product is a stego-image.

Extracting is getting the embedded message out of the stego message again. New terminology with respect to attacks and breaking steganography schemes is similar to cryptographic terminology; however, there are some significant differences. Just as a Cryptanalyst applies Cryptanalysis in an attempt to decode or crack encrypted messages, the Steganalyst is one who applies Steganalysis in an attempt to detect the existence of hidden information. With cryptography, comparison is made between portions of the plaintext (possibly none) and portions of the ciphertext. In steganography, comparisons may be made between the cover-media, the stego-media, and possible portions of the message. The end result in cryptography is the ciphertext, while the end result in steganography is the stego-media. The message in steganography may or may not be encrypted. If it is encrypted, then if the message is extracted, the cryptanalysis technique may be applied [3], [4].

The advantage of steganography is that it can be used to secretly transmit messages without the fact of the transmission being discovered. Often, using encryption might identify the sender or receiver as somebody with something to hide. For example, that picture of your cat could conceal the plans for your company's latest technical innovation [5].

3. Steganography Methods

The substitution technique is one of the common and important methods of hiding information.
This technique replaces data in the original file with a coded representation of the original message. The colors of "pixels", tiny elements of digital images are often represented by the value of a number contained in an eight-bit byte of data. For example, three increasingly redder shades of red might be represented as follows:
“00001100” or decimal 12 might represent basic red in a particular 8-bit color palette. Each of the following numbers would then represent a minor increase in the redness.
“00001101” or decimal 13
“00001110” or decimal 14
The likelihood of a casual observer noticing the difference in the shades in the middle of a picture is very slight. The result is that steganographers are able to use the 2 least significant bits to encode messages and while the image does degrade slightly, it is not apparent to the naked eye [6].

4. Basic Model of Information Hiding

Each steganographic technique consists of an embedding algorithm and a detection function. The embedding algorithm is used to hide secret message inside a cover (or carrier) document. The embedding process is usually protected by a keyword so that the only one who possesses the secret keyword can access the hidden message. The detector function is applied to the stego-document and results the hidden secret message.
A possible formula of the process may be represented as:

Cover media + embedded message + stegokey = stegomedia.

Figure (1) shows the general acceptable model of a steganography system.

For secure covert communication, it is important that by injecting a secret message into a cover document, no detectable changes are introduced. The main goal is not to raise suspicion and avoid introducing statistically detectable modifications into the stego-document. The quantity of embedded data and the degree of host signal modification vary from one application to one other [8].

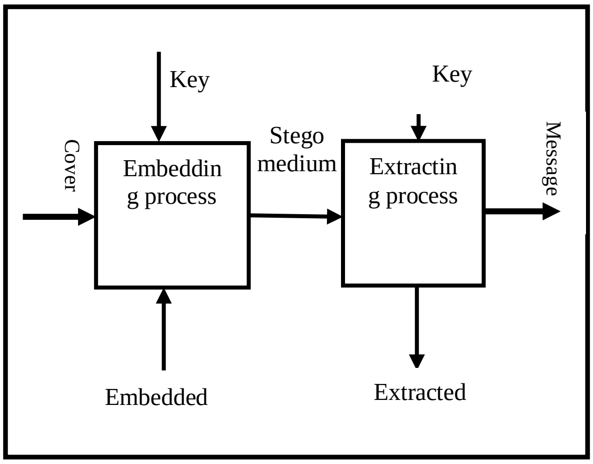


Fig. 1. General steganography system (Stego-system) [7].

5. Wave Audio files

The RIFF file format is a standard published as a joint design document by IBM and Microsoft. The data in WAVE files can be of many different types [9]. WAVE or WAV, short for Waveform Audio File Format [10] (also, but rarely, named, Audio for Windows [11]) is a Microsoft and IBM audio file format standard for storing an audio bitstream on PCs. It is an application of the RIFF bitstream format method for storing data in “chunks”, and thus is also close to the 8SVX and the AIFF format used on Amiga and Macintosh computers, respectively. It is the main format used on Windows systems for raw and typically uncompressed audio. The usual bitstream encoding is the Linear Pulse Code Modulation (LPCM) format. Though a WAV file can hold compressed audio, the most common WAV format contains uncompressed audio in the linear pulse code modulation (LPCM) format. The standard audio file format for CDs is LPCM-encoded, containing two channels of 44,100 samples per second, 16 bits per sample. Data format codes are listed in the following [9]:

5.1. Wave File Format

Wave files have a master RIFF chunk which includes a WAVE identifier followed by sub-chunks. The data is stored in little-endian byte order (See Table (1)).

Table 1,
Wave File Format .

Field	Length	Contents
ckID	4	Chunk ID: "RIFF"
Cksize	4	Chunk size: 4+*n*
WAVEID	4	WAVE ID: "WAVE"
WAVE chunks	*n*	Wave chunks containing format information and sampled data

5.2. Format Chunk

The Format chunk specifies the format of the data. There are 3 variants of the Format chunk for sampled data. These differ in the extensions to the basic Formant chunk (See Table (2)).

Table 2,
Format Chunk.

Field	Length	Contents
ckID	4	Chunk ID: "fmt "
Cksize	4	Chunk size: 16 or 18 or 40
wFormatTag	2	Format code
nChannels	2	Number of interleaved channels
nSamplesPerSec	4	Sampling rate (blocks per second)
nAvgBytesPerSec	4	Data rate
nBlockAlign	2	Data block size (bytes)
wBitsPerSample	2	Bits per sample
cbSize	2	Size of the extension (0 or 22)
wValidBitsPerSample	2	Number of valid bits
dwChannelMask	4	Speaker position mask
SubFormat	16	GUID, including the data format code

The standard format codes for waveform data are given below (See Table (3)). The references above give many more format codes for compressed data, a good fraction of which are now obsolete.

Table 3,
The Standard Format Codes for Waveform Data.

Format Code	PreProcessor Symbol	Data
0x0001	WAVE_FORMAT_PCM	PCM
0x0003	WAVE_FORMAT_IEEE_FLOAT	IEEE float
0x0006	WAVE_FORMAT_ALAW	8-bit ITU-T G.711 A-law
0x0007	WAVE_FORMAT_ MULAW	8-bit ITU-T G.711 µ-law
0xFFFE	WAVE_FORMAT_ EXTENSIBLE	Determined by Sub Format

This paper we focuses on PCM data.

5.3. Pulse Code Modulation (PCM) Format

The first part of the Format chunk is used to describe PCM data:

- For PCM data, the Format chunk in the header declares the number of bits/sample in each sample (wBitsPerSample). The original documentation (Revision 1) specified that the number of bits per sample is to be rounded up to the next multiple of 8 bits. This rounded-up value is the container size. This information is redundant in that the container size (in bytes) for each sample can also be determined from the block size divided by the number of channels (nBlockAlign / nChannels).
- This redundancy has been appropriated to define new formats. For instance, Cool Edit uses a format which declares a sample size of 24 bits together with a container size of 4 bytes (32 bits) determined from the block size and number of channels. With this combination, the data is actually stored as 32-bit IEEE floats.
- PCM data is two's-complement except for resolutions of 1-8 bits, which are represented as offset binary.

5.4. Examples

Consider sampled data e.g *voice.wav* with the following parameters,

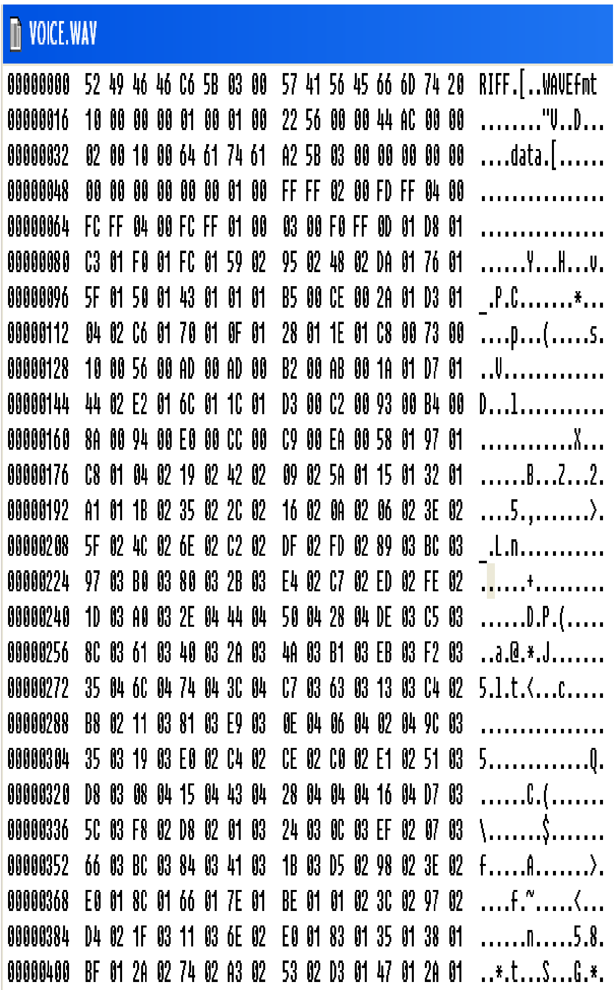
- $N_c = 1$ channels.
- The total number of blocks is N_s = 110033. Each block consists of N_c samples.
- Sampling rate F = 22050 (blocks per second).
- Each sample is M = 2 bytes long.

As shown in table (4).

Table 4,
Example of Wave File (*voice.wav*) format.

Field	Length	Contents
ckID	4	Chunk ID: "RIFF"
Cksize	4	Chunk size: $4+24+(8+M * N_c*N_s +(0 \text{ or } 1))=220102$
WAVEID	4	WAVE ID: "WAVE"
ckID	4	Chunk ID: "fmt "
Cksize	4	Chunk size = 16
wFormatTag	2	WAVE_FORMAT_ PCM = 1
nChannels	2	$N_c = 1$
nSamplesPerSec	4	$F = 22050$
nAvgBytesPerSec	4	$F * M * N_c = 44100$
nBlockAlign	2	$M * N_c = 2$
wBitsPerSample	2	rounds up to $8 * M = 16$
ckID	4	Chunk ID: "data"
Cksize	4	Chunk size: $M * N_c * N_s = 220066$
sampled data	$M * N_c * N_s$	N_c*N_s channel-interleaved M-byte samples
Pad	0 or 1	Padding byte if $M * N_c * N_s$ is odd

WAVE files often have information chunks that precede or follow the sound data (Data chunk). Some programs assume that for PCM data, the file header is exactly 44 bytes long and that the rest of the file contains sound data. This is not a safe assumption. Figure (2) shows voice.wav file in hexadecimal and character representation.



```
VOICE.WAV
00000000  52 49 46 46 C6 5B 03 00  57 41 56 45 66 6D 74 20  RIFF.[..WAVEfmt
00000016  10 00 00 00 01 00 01 00  22 56 00 00 44 AC 00 00  ........"V..D...
00000032  02 00 10 00 64 61 74 61  A2 5B 03 00 00 00 00 00  ....data.[......
00000048  00 00 00 00 00 00 01 00  FF FF 02 00 FD FF 04 00  ................
00000064  FC FF 04 00 FC FF 01 00  03 00 F0 FF 0D 01 D8 01  ................
00000080  C3 01 F0 01 FC 01 59 02  95 02 48 02 DA 01 76 01  ......Y...H...v.
00000096  5F 01 50 01 43 01 01 01  B5 00 CE 00 2A 01 D3 01  _.P.C.......*...
00000112  04 02 C6 01 70 01 0F 01  28 01 1E 01 C8 00 73 00  ....p...(.....s.
00000128  10 00 56 00 AD 00 AD 00  B2 00 AB 00 1A 01 D7 01  ..V.............
00000144  44 02 E2 01 6C 01 1C 01  D3 00 C2 00 93 00 B4 00  D...l...........
00000160  8A 00 94 00 E0 00 CC 00  C9 00 EA 00 58 01 97 01  ............X...
00000176  C8 01 04 02 19 02 42 02  09 02 5A 01 15 01 32 01  ......B...Z...2.
00000192  A1 01 1B 02 35 02 2C 02  16 02 0A 02 06 02 3E 02  ....5.,.......>.
00000208  5F 02 4C 02 6E 02 C2 02  DF 02 FD 02 89 03 BC 03  _.L.n...........
00000224  97 03 B0 03 80 03 2B 03  E4 02 C7 02 ED 02 FE 02  ......+.........
00000240  1D 03 A0 03 2E 04 44 04  50 04 28 04 DE 03 C5 03  ......D.P.(.....
00000256  8C 03 61 03 40 03 2A 03  4A 03 B1 03 EB 03 F2 03  ..a.@.*.J.......
00000272  35 04 6C 04 74 04 3C 04  C7 03 63 03 13 03 C4 02  5.l.t.<...c.....
00000288  B8 02 11 03 81 03 E9 03  0E 04 06 04 02 04 9C 03  ................
00000304  35 03 19 03 E0 02 C4 02  CE 02 C0 02 E1 02 51 03  5.............Q.
00000320  D8 03 08 04 15 04 43 04  28 04 04 04 16 04 D7 03  ......C.(.......
00000336  5C 03 F8 02 D8 02 01 03  24 03 0C 03 EF 02 07 03  \.......$.......
00000352  66 03 BC 03 84 03 41 03  1B 03 D5 02 98 02 3E 02  f.....A.......>.
00000368  E0 01 8C 01 66 01 7E 01  BE 01 01 02 3C 02 97 02  ....f.~.....<...
00000384  D4 02 1F 03 11 03 6E 02  E0 01 83 01 35 01 38 01  ......n.....5.8.
00000400  BF 01 2A 02 74 02 A3 02  53 02 D3 01 47 01 2A 01  ..*.t...S...G.*.
```

Fig. 2. Voice .Wav File in Hexadecimal and Character Representation.

6. Audio Stenography System Using LSB-Technique

The aim of this paper is to implement an algorithm for an information hiding technique using LSB in digital wave files. This section shows the hiding and extracting algorithms supported by encryption algorithm and then shows the proposed system implementation with an experimental example. The proposed systems in information hiding algorithm consist of two main algorithms. The first one interested in hiding text in wave cover file called hiding algorithm and the second one is specialized in extracting data from the stego wave file called the extracting algorithm.

6.1. Hiding Algorithm

This algorithm consist of two stages, these stages contribute to each other to obtain a secure algorithm. The first one is the enciphering stage and the second is embedding stage. Hiding algorithm is based on hoping style. The proposed algorithm details can be described in the following steps:

1. Read cover-wave file.
2. Enter plain-text characters.
3. Skip (60) byte from beginning of the file.
4. Calculate the plaintext size (n bytes≤1/(5*8)=1/40 of the size of container) (or length), n must be embedded in the container file in order to be known during the extracted process, n represented by (3) bytes (=24 bits) then embed the 1st bit in the LSB of the Hide-byte of the container, then jump (5) bytes to embed the 2nd bit,…and so on until finish embedding all the 24 bits.
5. Embedd data: this step includes two processes:

   a - Enciphering process: consider the current byte as a Encryption key-byte. This byte add (XORing) to plaintext byte according to the following equation:

   Cipher-byte = Plain-byte XOR Key-byte.

   b- Embedding process: consider the current byte as a jump-key embed the 1st bit of cipher-byte in the next byte using LSB technique, then jump a random step according to the following equation:

   Jump-step = (Jump-key MOD Mode-Byte) + Shift-Byte.

   Repeating the process to embedding the 2nd bit of cipher-byte after jumping by jump-step until finishing all bits of the cipher-byte . Repeating the process in (a) and (b) until finishing all plaintext . The hiding algorithm steps as shown in Figure (3).

```
INPUT:      Reading stego-wave file
PROCESS:  Entering plain-text
            Skip first (60) byte from
            wave file
             Let n=plain-text length (3 byte
             = 24 bits)
Embedding n in(3)bytes with(5)byte jump
using Hide-byte
             For i=1 to n
             key-byte = Encryption key-
             byte
              cipher-byte = plain-byte XOR
             key-byte
              jump-key = Hide-byte
            For j=1 to 8
            embed-byte = current byte
        embed-byte [LSB] = cipher-byte[j]
     jump-step=( jump-key MOD Mode-
                   byte)+Shift-bye
                  Endfor {j}
                  Endfor {i}
OUTPUT:      stego wave file
End
```

Fig. 3. Hiding Algorithm .

<u>**Note:**</u> the keys encryption key-byte, hide byte, Mode-byte and Shift-byte are secret keys that can be send by other communication means except the same cover wave file.

The flowchart of the hiding algorithm is shown in Figure (4).

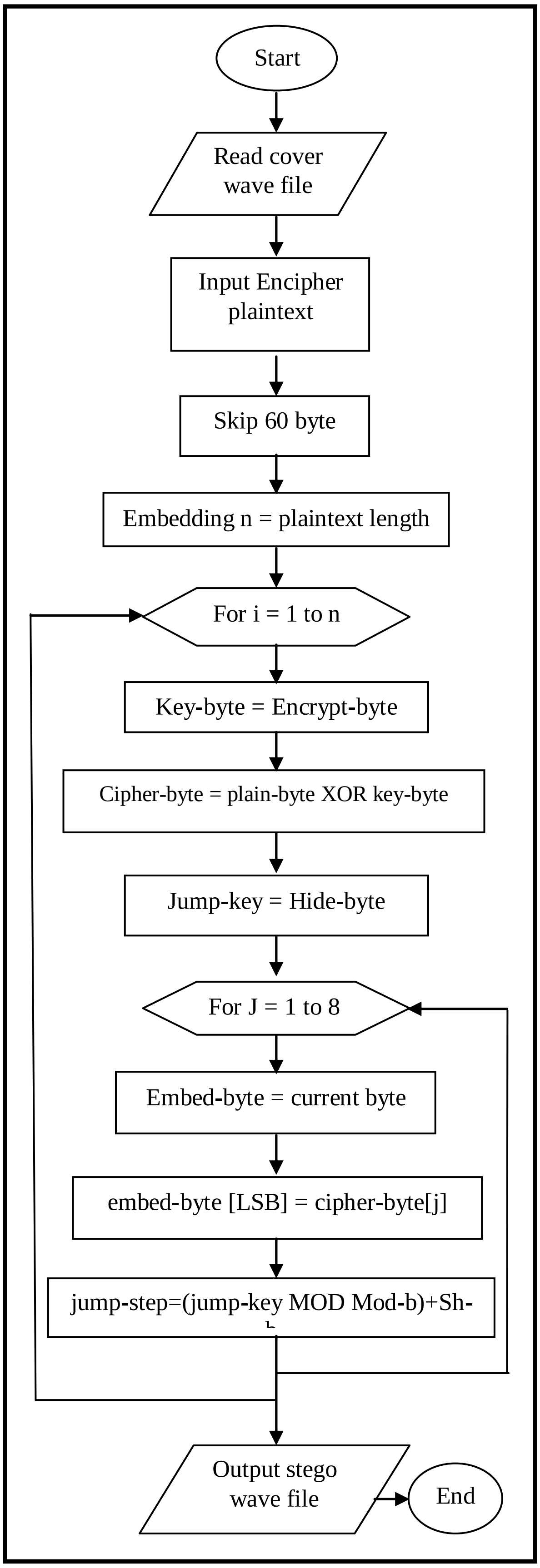
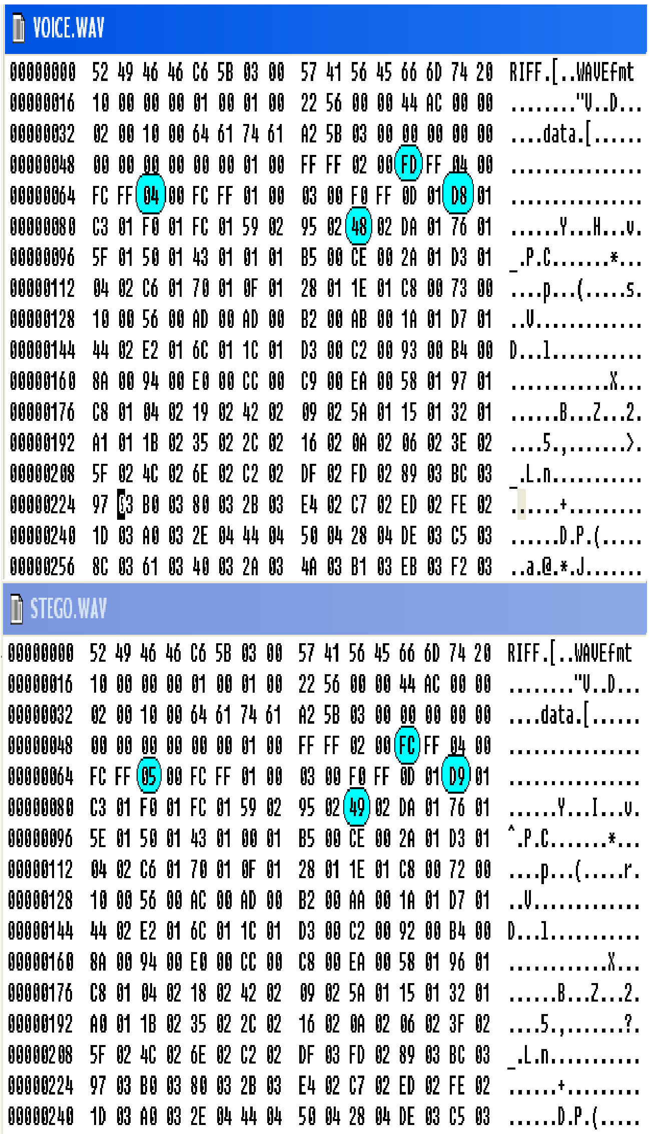


Fig. 4. Hiding Algorithm Flow Chart .

6.2. Practical Result

In this subsection we will introduce a practical example of hiding algorithm. Let's choose the plain text:" Information Hiding System for Wave Audio Files", which is to be hide.

In figure (5) the data of two files are shown. First is voice.wav represent the audio wave file before hiding, and the second is stego.wav which represent the voice.wav file after end of hiding process.



```
VOICE.WAV
00000000  52 49 46 46 C6 5B 03 00  57 41 56 45 66 6D 74 20  RIFF.[..WAVEfmt
00000016  10 00 00 00 01 00 01 00  22 56 00 00 44 AC 00 00  ........"V..D...
00000032  02 00 10 00 64 61 74 61  A2 5B 03 00 00 00 00 00  ....data.[......
00000048  00 00 00 00 00 00 01 00  FF FF 02 00 FD FF 04 00  ................
00000064  FC FF 04 00 FC FF 01 00  03 00 F0 FF 0D 01 D8 01  ................
00000080  C3 01 F0 01 FC 01 59 02  95 02 48 02 DA 01 76 01  ......Y...H...v.
00000096  5F 01 50 01 43 01 01 01  B5 00 CE 00 2A 01 D3 01  _.P.C.......*...
00000112  04 02 C6 01 70 01 0F 01  28 01 1E 01 C8 00 73 00  ....p...(.....s.
00000128  10 00 56 00 AD 00 AD 00  B2 00 AB 00 1A 01 D7 01  ..V.............
00000144  44 02 E2 01 6C 01 1C 01  D3 00 C2 00 93 00 B4 00  D...l...........
00000160  8A 00 94 00 E0 00 CC 00  C9 00 EA 00 58 01 97 01  ............X...
00000176  C8 01 04 02 19 02 42 02  09 02 5A 01 15 01 32 01  ......B...Z...2.
00000192  A1 01 1B 02 35 02 2C 02  16 02 0A 02 06 02 3E 02  ....5.,.......>.
00000208  5F 02 4C 02 6E 02 C2 02  DF 02 FD 02 89 03 BC 03  _.L.n...........
00000224  97 03 B0 03 80 03 2B 03  E4 02 C7 02 ED 02 FE 02  ......+.........
00000240  1D 03 A0 03 2E 04 44 04  50 04 28 04 DE 03 C5 03  ......D.P.(.....
00000256  8C 03 61 03 40 03 2A 03  4A 03 B1 03 EB 03 F2 03  ..a.@.*.J.......
STEGO.WAV
00000000  52 49 46 46 C6 5B 03 00  57 41 56 45 66 6D 74 20  RIFF.[..WAVEfmt
00000016  10 00 00 00 01 00 01 00  22 56 00 00 44 AC 00 00  ........"V..D...
00000032  02 00 10 00 64 61 74 61  A2 5B 03 00 00 00 00 00  ....data.[......
00000048  00 00 00 00 00 00 01 00  FF FF 02 00 FC FF 04 00  ................
00000064  FC FF 05 00 FC FF 01 00  03 00 F0 FF 0D 01 D9 01  ................
00000080  C3 01 F0 01 FC 01 59 02  95 02 49 02 DA 01 76 01  ......Y...I...v.
00000096  5E 01 50 01 43 01 00 01  B5 00 CE 00 2A 01 D3 01  ^.P.C.......*...
00000112  04 02 C6 01 70 01 0F 01  28 01 1E 01 C8 00 72 00  ....p...(.....r.
00000128  10 00 56 00 AC 00 AD 00  B2 00 AA 00 1A 01 D7 01  ..V.............
00000144  44 02 E2 01 6C 01 1C 01  D3 00 C2 00 92 00 B4 00  D...l...........
00000160  8A 00 94 00 E0 00 CC 00  C8 00 EA 00 58 01 96 01  ............X...
00000176  C8 01 04 02 18 02 42 02  09 02 5A 01 15 01 32 01  ......B...Z...2.
00000192  A0 01 1B 02 35 02 2C 02  16 02 0A 02 06 02 3F 02  ....5.,.......?.
00000208  5F 02 4C 02 6E 02 C2 02  DF 03 FD 02 89 03 BC 03  _.L.n...........
00000224  97 03 B0 03 80 03 2B 03  E4 02 C7 02 ED 02 FE 02  ......+.........
00000240  1D 03 A0 03 2E 04 44 04  50 04 28 04 DE 03 C5 03  ......D.P.(.....
```

Fig. 5. The Data of Voice.Wav (Before Embedding) and Stego.Wav (After Embedding).

The shaded and circled hexadecimals represent the change bytes after exchanging the LSB of the specified byte.

6.3. Extracting Algorithm

This algorithm consists of two stages.The first one is the extracting stage and the second is deciphering the extracting ciphertext. Extracting algorithm details can be described as follows:

1. Read stego-wave file.
2. Skip (60) byte from beginning of the file.
3. Extracting the embedding plaintext length (n) from the LSB of specifying bytes with (5) bytes jump using encryption key-byte.
4. Extracting data: this step includes two process:

a- Extracting process: preparing the key-byte considering the current byte as a key-byte, then consider the hide-byte as a jump-key, then extracting the 1st bit of cipher-byte from the LSB of the current byte, after that, jumping in a random step according to the following equation:

Jump-step = (Jump-key MOD Mode-byte) + Shift-byte.

Repeating the process until get all the bits of the cipher-byte .

b- Deciphering process: to obtain the plain-text by:

Plain-byte = Cipher-byte XOR Key-byte.

Repeating the process in (a) and (b) until all the plain-text characters are extracted. The extracting algorithm steps can be shown in Figure (6).

```
INPUT: Reading stego-wave file
PROCESS: Skip first (60) byte from stego-
         wave file
    Extracting plaintext length (n) from LSB
    of the specifying bytes
    For i=1 to n
        key-byte = Encryption-byte
        jump-key = Hide-byte
        For j=1 to 8
        embed-byte = current byte
        cipher-byte [j] = embed- byte[LSB]
jump-step=(jump-key MOD Mod-by)+Shift-byte
        Endfor {j}
  plain-byte = cipher-byte XOR key-byte
        Endfor {i}
OUTPUT: Plaintext
END
```

Fig. 6. The Extracting Algorithm.

The flowchart of the extracting algorithm is show in figure (7).

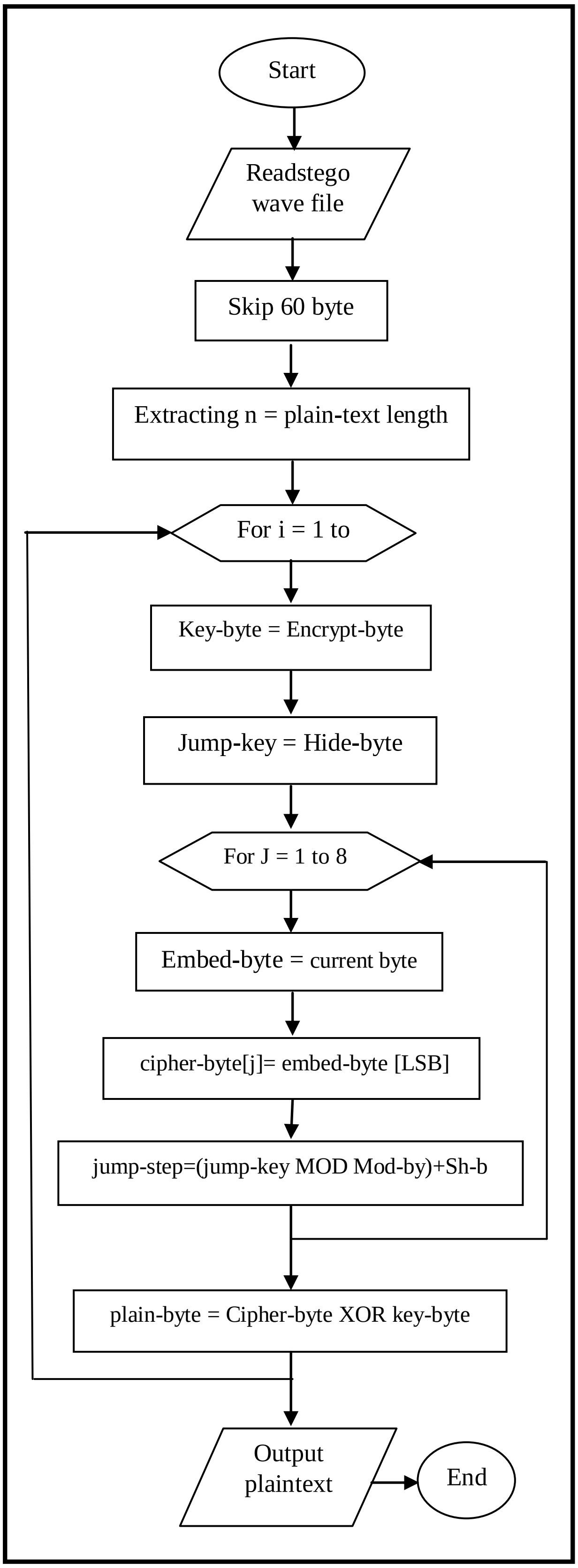


Fig.7. The Extracting Algorithm.

7. Conclusions

This paper use LSB data-hiding technique, depending on auto-key generator. It obvious the LSB-Technique can be used when the cover file is uncompressed file but it can be used when the cover file is lossless compressed file, as done in .png image files. Any random data added to stego file (not only in LSB) means that real noise in audio file can be heard and that's affects the extraction of the information. A random key generator is used for two purposes, first to encrypt the hidden message, and second to generate random jumping in the wave file to give more robustness to the steganography system.

8. References

[1] Sellars, D., “An Introduction to Steganography”, http://www.cs.uct.ac.za/courses/CS400W/NIS/papers99/dsellars/stego.ps.gz, 1999.

[2] Stinson, D. R., “Cryptography: Theory and Practice” CRC Press, 1995.

[3] Rifat Z. K. “Statistical Approach for Steganalysis”, M.Sc. Thesis, Applied Sciences / University of Technology, Baghdad-Iraq, 2003.

[4] Katzenbeisser, S. and Fabien A. P. Petitcolas, “Information Hiding Techniques for Steganograpghy and Digital Watermarking”, Artech House, Boston, London, 2000.

[5] Stalling,W., “Network and Internetwork Security”, Addison Wesley professional computing series. Addison-Wesley, 1996.

[6] Al-hamami. M., “Information Hiding Attack in Image”, M. Sc. Thesis, Iraqi Commission for Computer & Informatics, Informatics Institute for Postgraduate Studies, Baghdad-Iraq, 2002.

[7] Poluami D., Debnath B., and Tai-hoon K., “Data Hidin in audio signal : A Review”, International Journal of Database Theory and Application, vol.2, No.2, June, 2009.

[8] Johnson, N. F., “Steganography”, http://www.jjtc.com/stegdoc/, George Mason University, 2003.

[9] Kabal, P., “Audio File Format Specifications - WAVE or RIFF WAVE sound file”, McGill University,http://www.mmsp.ece.mcgill.ca/AudioFormats/WAVE/WAVE.html, Retrieved 2010.

[10] IBM Corporation and Microsoft Corporation (August 1991), "Multimedia Programming Interface and Data

Specifications 1.0", http://www.tactilemedia.com/info/MCI_Control_Info.html, Retrieved 6-12-2009.

[11] Microsoft Help and Support, “Information about the Multimedia file types that Windows Media Player supports”, Microsoft Corporation.12May 2008, http://support.microsoft.com/kb/316992. Retrieved 29 May 2009.

نظام جديد للأخفاء بالصوت يعتمد على مولد مفتاح- ذاتي

ايناس جواد كاظم
قسم هندسة تقنيات القدرة الكهربائية/ كلية التقنيات الكهربائية والالكترونية
البريد الالكتروني: inascnn95@yahoo.com

الخلاصة

علم الاخفاء هو فن اخفاء كل ماهو مهم باستخدام الاتصالات بواسطة اغمار الرسالة السرية في وثيقة غطاء عادية، كالصور الرقمية، المرئية، ملفات الصوت وغيرها من ملفات الحاسوب ذات المعلومات المتكررة تستخدم كاغطية او حوامل لاخفاء المعلومات السرية.

في هذا البحث، تم اقتراح نظام اخفاء غير متسلسل باستخدام تقنية الاغمار بالثنائي الاقل اهمية (LBS) بالاعتماد على ملفات صوتية من نوع (wav). ولزيادة درجة الامنية لنظام الاخفاء المقترح، و للقضاء على نقاط الضعف باستخدام تلك التقنية، تم اقتراح وتحقيق بعض العمليات المساعدة ومنها استخدام عملية القفز بالنص المخفي بالاضافة الى استخدام خوارزمية التشفير الانسيابي واقتراح استخدام نظام تشفير-اخفاء باستخدام مولد عشوائي ذاتي. هذا المولد الذاتي يعمل لتحقيق غرضين هما عمليات التشفير والتضمين. اظهرت النتائج انه لايمكن ان تسمع ضوضاء في ملفات الصوت المخفيه (stego) بعد عملية التضمين، كذلك لايوجد اختلاف بين ملف الصوت الاصلي وملف الصوت المخفي(stego) من حيث الحجم.