Wave File Encryption using Huffman Compression and Serpent Algorithm

Parasian D.P Silitonga[#1], Imka Ardianta Singarimbun[*2], Irene Sri Morina[#3]
[#]Faculty Of Computer Science & St. Thomas Catholic University
Setia Budi Medan, Indonesia

Abstract

Wave file (.wav) is a standard audio file used by Windows. Sound recording files (.wav) tend to have a large size, according to the length of time sound recording. Large files can cause various problems in the form of large space requirements for listening and a long time in the shipping process. File compression is one solution that can be done to overcome large file size problems. One method of file compression is to use the Huffman compression method. In addition to problems in space and length of delivery time in the file processing process, data security factors are also a problem that continues to this day. The Serpent algorithm is one of the data cryptography algorithms that can be used to maintain data confidentiality. Serpent is a block cipher algorithm that has a block size of 128 bits and supports key sizes of 128, 192, or 256 bits. The results obtained in this study are the Huffman compression ratio and the implementation of the Serpent algorithm performed on wave files.*

Keywords —*Wave File, File Compression, Huffman Method, Criptography, Serpent Algorithm*

I. INTRODUCTION

Wave file (.wav) is a standard audio file used by windows [1]. Sounds in wave form are stored in digital audio data format in * .wav files. Sound recording files (* .wav) tend to have a large size, according to the length of time sound recording.

In the process of storing and sending data, large files have constraints that require large space to store and require a long time to send. To overcome this, file compression can be done. Compression is the process of encoding information using fewer bits than the initial information [10]. There are two types of compresses, namely lossless and lossy [17].

Huffman compression is a lossless compression algorithm and is ideal for compressing text or program files [16]. Huffman compression is included in the variable codeword length algorithm [14]. In this algorithm the individual symbols are replaced by bit sequences which have a distinct length [13].

Cryptography is a technique used to guarantee the security aspects of data exchange such as data confidentiality, data correctness, data integrity, and data authentication [9]. There are various kinds of encryption algorithms with their respective characteristics, one of which is the Serpent algorithm.

Serpent is a block cipher algorithm that has a block size of 128 bits and supports key sizes of 128, 192, or 256 bits [7]. This cipher is in the form of a Subtitution-Permutation Network (SP-network) which is a series of mathematical operations that are related slings. SP-networks have S-boxes that convert the input bit blocks into an output bit.

II. LITERATURE REVIEW

A. Wave File

Wave file (.wav) is a standard audio file used by Windows [1]. Wave files allow various audio forms to be recorded in various qualities, such as 8-bit or 16-bit samples with a rate of 11025 Hz, 22050 Hz or 44100 Hz [12].

Wave files are widely used in game making. Usually for sound effects and music. Wave itself tends to have a large size, but this is because the wave file format is uncompressed so it has a faster loading time. Digital audio data in wave files can have varying qualities. The quality of the resulting sound is determined by the bitrate, samplerate, and number of channels [6].

Bitrate is the bit size of each sample, which is 8 bits, 16 bits, 24 bits or 32 bits [3]. In 8-bits wav all the samples will only take as much as 1 byte. Whereas for 16-bits it will take 2 bytes.

While samplerate states the number of samples played every second. Samplerate commonly used is 8000 Hz, 1105 Hz, 22050 Hz, and 44100 Hz [3]. While the number of channels determines the sound produced whether mono or stereo [3]. Mono has only 1 channel, while stereo 2 channels will take up to 2 times more space than mono.

B. File Compression

Data compression is done to reduce the size of data or files. By compressing or compressing data, the file or data size will be smaller so that it can reduce transmission time when data is sent and not spend much storage media space [18].

Compression is the process of encoding information using fewer bits than the initial information [10]. There are two types of compresses, namely lossless and lossy [17]. n lossless compression, the data will initially be broken down into smaller sizes and finally the data is reassembled. Whereas, in lossy compression, there are bits of information which are eliminated after being done for compression [10].

The general principle in the compression process is to reduce duplication of data so that the memory represents less than representing the original digital data [17].

C. Huffman Method

The Huffman compression algorithm is named after its inventor, David Huffman, a professor at MIT (Massachusetts Institute of Technology. Huffman compression is a lossless compression algorithm and is ideal for compressing text or program files [16]. This causes why this algorithm is widely used in compression programs.

Huffman compression is included in the variable codeword length algorithm [14]. In this algorithm the individual symbols are replaced by bit sequences which have a distinct length [13]. Symbols that appear quite a lot in the file will give a short sequence while symbols that are rarely used will have a longer sequence of bits. The way the Huffman method works is by encoding the bits (a combination of bits 0 and 1) to represent the actual data [2].

D. Criptography

Cryptography is a technique used to guarantee the security aspects of data exchange such as data confidentiality, data correctness, data integrity, and data authentication [9]. To ensure the security of data exchange, it can be done in various ways, one of which is by encoding the password algorithm.

The encoding process is carried out so that the data sent cannot be understood by anyone other than those who have access to the data [Rinaldi Munir]. In the encoding process there are two main concepts, namely encryption and decryption.

Encryption is the process of changing data or information that will be sent into a form that is almost unrecognized as the initial information [5]. Encryption is usually done before the data or information is sent.

In cryptography process, data or information that is understandable means that it is known as plain text or clear text while the information that has been obscured is known as a text cipher [4]. To improve the security of information encryption, the key is added to the encryption process. Whereas decryption is the process of converting ciphertext into plain text.

E. Serpent Algortihm

Serpent was designed by Ross Anderson, Eli Biham and Lars Knudsen [11]. The Serpent algorithm is faster than DES and is safer than Triple DES [8]. This provides users with a very high level of assurance that no shortcut attacks will be found. To achieve this, algorithm designers limit themselves to well-understood cryptographic mechanisms, so they can rely on extensive experience and proven techniques of cipher block cryptanalysis.

The Serpent algorithm uses twice as many rounds as needed to block all the shortcut attacks that are currently happening. This means that the Serpent must be safe against as unknown attacks that might be able to break 16 standard rounds used in various types of encryption at this time. However, the fact that Serpent uses so many rounds.

Serpent is a cipher block algorithm that has a block size of 128 bits and supports key sizes of 128, 192, or 256 bits [7]. This cipher is in the form of a Subtitution-Permutation Network (SP-network) which is a series of mathematical operations that are related slings. SP-networks have S-boxes that convert the input bit blocks into an output bit.

III.METHODOLOGY

In this study the wave file was obtained through two sources, namely microphone and audio file. To process input from the microphone, a digitalisation mechanism is needed. Then the compression process is done to reduce the size to be entered into the encryption process.

The serpent algorithm is used to encrypt the voice message flow by changing the operating mode that is used until the characteristics resemble the flow cipher, that is, the counter operation mode.

A. Huffman Compression on Wave File

The way the Huffman method works is to encode the bit (a combination of bits 0 and 1) to represent the actual data. The compression steps with the Huffman method are as follows [16] :

1) Calculate the frequency or weight of the appearance of each character in a file.

Wave files use a standard RIFF structure that groups file contents into separate chunks [3]. Each section has its own header and data byte.

Here is a wave file where the first 60 bytes of a wave file are displayed in hexadecimal :

```
52 49 46 46 24 08 00 00 57 41 56 45
66 6D 74 20 10 00 00 00 01 00 02 00
22 56 00 00 88 58 01 00 04 00 10 00
64 61 74 61 00 08 00 00 00 00 cA 00
00 00 CA 0F 00 00 CA F0 00 00 CA FF
```

To form a Huffman frequency table, a 16 byte data wave file is taken starting from the 45th byte offset, which is data:

```
00 00 CA 00 00 00 CA 0F 00 00 CA F0
00 00 CA FF
```

The frequency distribution table obtained is presented as in Table I.

Table i. Frequency distribution table

Character	00	CA	0F	F0	FF
Frequency	9	4	1	1	1

After that, a tree node is formed for each character along with their respective frequency values like Fig. 1

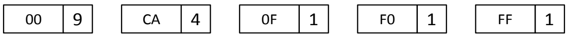

Fig. 1. Table I Node Tree

2) Take two characters that have the smallest frequency.

Based on the node in Fig 1, it is obtained that the characters of FO and FF are the smallest nodes.

3) Huffman tree form from both data taken. The sum of the two characters is set as temporary root and both characters are set as leaves.

Both characters are FO and FF combined to form a new tree with the root value is the sum of the values of the weight of F0 and FF. Then, the two tree nodes are deleted and replaced with a new tree resulting from the merger of the two nodes. The results of this stage are presented as Fig. 2.

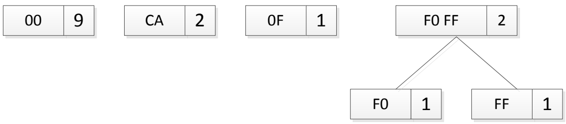

Fig. 2. Tree Results Merging FO and FF

After that each node is sorted again from the largest to the smallest value as in Fig. 3.

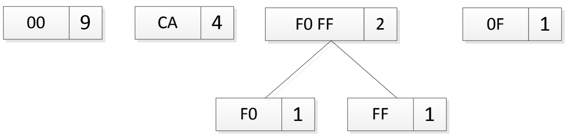

Fig. 3. Trees that have been sorted

he process to produce Fig. 2 and Fig. 3 is carried out continuously for all remaining nodes, until the Huffman tree is obtained as in Fig. 4

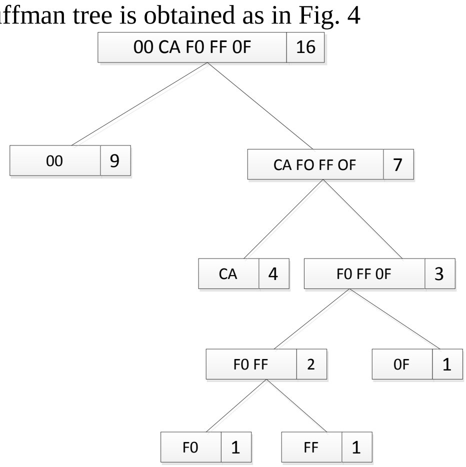

Fig. 4. Huffman Tree

4) Change the structure of the Huffman code into a binary tree form.

To read the code from this Huffman tree, start from the root and add 0 each time to the left side of the tree, and add 1 each time to the right. So that the results are obtained as shown in Fig. 5.

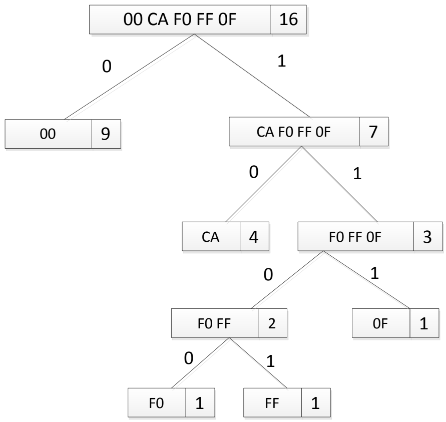

Fig. 5. Bit Code on the Huffman Tree

5) The bit pattern that will be used as a reference for each character is arranged starting from bits from roots to leaves.

Based on the Huffman tree obtained in Fig. 5, then the data bit code reading process is presented as in Table II.

TABLE II. BIT CODE HUFFMAN TREE

Symbol	Bit Code
00	0
CA	10
0F	111
F0	1100
FF	1101

To encode using the Huffman tree, for example the Wave data file is :

00 00 CA 00 00 00 CA 0F 00 00 CA F0 00 00 CA FF

Then the result by looking at the bit code table will be:

0 0 10 0 0 0 10 111 0 0 10 1100 0 0 10 1101

Determination of compression ratio is done by comparing the number of data bits before being compressed, namely the number of characters in bytes x 8 bits = 16 x 8 bits = 128 bits. While the number of bits resulting from compression is 28 bits, so the compression ratio obtained is

$$\frac{28}{128} \times 100\% = 21{,}875\ \%$$

B. Implementation Serpent Algorithm on Wave File

Suppose this wave data file starts from the 45th byte offset and is taken only 16 bytes and declared in hexadecimal.

Plaintext : 00 00 CA 00 00 00 CA 0F 00 00 CA F0 00 00 CA FF

Key :
000102030405060708090A0B0C0D0E0F

The serpent algorithm encrypts plaintext P 128 bits into 128-bit C ciphertext in 32 turns with control of 33 128-bit K_0, ..., K_{32} sub-keys. The user input key length used for this discussion is 128 bits. For the encryption process, Serpent requires 32 128-bit subkeys denoted by K_0, ..., K_{32}.

Steps to get all 33 subkeys, are:

1) Divide the input key K into eight parts, each 32 bits denoted by W_{-8}, ... , W_{-1}

00010203 04050607 08090A0B 0C0D0E0F

2) Forms 132 intermediate keys (prekey) denoted by W_0, ..., W_{131} through equations:

$W_i=(W_{i-8}\oplus W_{i-5}\oplus W_{i-3}\oplus W_{i-1}\oplus \varnothing \oplus i) <<< 11$

$\varnothing$ notation s a small part of the golden ratio ($\sqrt{5}$ + 1) / 2 or 0x9E3779B9 in hexadecimal.

$00010203 = W_{-8}, W_{-7}, W_{-6}, W_{-5}, W_{-4}, W_{-3}, W_{-2}, W_{-1}$

$W_0 = (W_{-8} \oplus W_{-5} \oplus W_{-3} \oplus W_{-1} \oplus \varnothing \oplus 0)$

$W_0 = (0 \oplus 1 \oplus 2 \oplus 3 \oplus \text{0x9E3779B9} \oplus 0)$

W_0 = 0x9E3779B9

……

3) Forming 132 round key k0 to k131 formed from the intermediate key generated from the previous process using S-boxes, S-boxes are used to change the W_i to K_i intermediate with the following conditions:

$\{K_0,K_1,K_2,K_3\}=S_3\ (W_0,W_1,W_2,W_3)$

$\{K_4,K_5,K_6,K_7\}=S_2\ (W_4,W_5,W_6,W_7)$

$\{K_8,K_9,K_{10},K_{11}\}=S_1\ (W_8,W_9,W_{10},W_{11})$

$\{K_{12},K_{13},K_{14},K_{15}\}=S_0\ (W_{12},W_{13},W_{14},W_{15})$

…

$\{K_{124},K_{125},K_{126},K_{127}\}=S_4\ (W_{124},W_{125},W_{126},W_{127})$

$\{K_{128},K_{129},K_{130},K_{131}\}=S_3\ (W_{128},W_{129},W_{130},W_{131})$

Here are S-boxes S_0 to S_7:

S0: 3 8 15 1 10 6 5 11 14 13 4 2 7 0 9 12
S1: 15 12 2 7 9 0 5 10 1 11 14 8 6 13 3 4
S2: 8 6 7 9 3 12 10 15 13 1 14 4 0 11 5 2
S3: 0 15 11 8 12 9 6 3 13 1 2 4 10 7 5 14
S4: 1 15 8 3 12 0 11 6 2 5 4 10 9 14 7 13
S5: 15 5 2 11 4 10 9 12 0 3 14 8 13 6 7 1
S6: 7 2 12 5 8 4 6 11 14 9 1 15 13 3 10 0
S7: 1 13 15 0 14 8 2 11 7 4 12 10 9 3 5 6

S-boxes taken S3, then:

K0 = S3 (W0)
K0 = 0 15 11 8 12 9 6 3 (0x9E3779B9)
K0 = C06704A4

The same thing is done to obtain round key K_0 to K_{131}.

4) Forms a 128 bit K_i subkey (For $I \in \{0, \ldots, 32\}$) from 32 bit Kj vaues by using :

$K_i=\{K_{4i}, K_{4i+1}, K_{4i+2}, K_{4i+3}\}$

KO = {KO, K1, K2, K3}

KO =
{C06704A4025A3A3462C7B9234E6C2FFB}

The sub-key formation process is carried out so that the key sub-units K_1 through K_{32} are obtained.

Then the encryption stage on the Serpent is performed, ie each function of the Ri loop (i = 0, ..., 31) only uses a replicated S-Box. For example, R_0 uses S_0, 32 copies are applied in parallel, so the copy of S0 uses bits 0,1,2, and 3 of PT⊕ K0 as input and return the first four bits of the intermediate vector as output, the next copy receives input bits 4-7 from PT⊕K0 and return the next four bits of the intemediate vector, and so on.

The intermediate vector is then transformed using linear transformations, producing CT0. In the same way, R_1 uses 32 S_1 copies in parallel on CT1⊕K1 and transform the output using linear transformation, producing CT2

.

PT⊕K0= 0 0 0 0 C A 0 0 0 0 0 0 C A 0 F 0 0 0 0 C A F 0 0 0 0 0 C A F F ⊕ C 0 6 7 0 4 A 4 0 2 5 A 3 A 3 4 6 2 C 7 B 9 2 3 4 E 6 C 2 F F B = C 0 6 7 C E A 4 0 2 5 A F A 3 B 6 2 C 7 7 3 D 3 4 E 6 C E 5 0 4

So as to produce:

Input : C 0 6 7 C E A 4 0 2 5 A F A 3 B 6 2 C 7 7 3 D 3 4 E 6 C E 5 0 4
Output : C 0 6 7 C E A 4 0 2 5 A F A 3 B 6 2 C 7 7 3 D 3 4 E 6 C E 5 0 4

The output is then transformed using a linear transformation where for each output bit of this transformation, a list of parity input bits is the output bit. Bits are registered from 0 to 127. In each row four output bits are obtained which together form input to one S-box in the next round.

The result of the output is transformed with a linear transformation resulting in CT0.

Here are the results of all CT0 to CT31 :

CT1=FF46D7451ED630CF95FF7FDD5D19F778
CT2=86DE6AB3C100339590FC1E7E6A9AF5AC
CT3=FEFE0E66161A763014099215E5E80F41
CT4=2D200ECED822AB937ECB9D1BD5CB2681

.
.
.

CT31=1A861B95BB2683797BA26A2EC8BCE3A4

So that the ciphertext in hexadecimal form is:
CT =8FC7C02454F6FF2E1112899966C965FA

IV. RESULT AND DISCUSSION

Wave file compression and decompression testing is performed on several types of Wave files. The Wave file that is tested has a large size that varies and testing is carried out on Wave files that are included in the Windows operating system. The test results are presented in Fig.6.

No.	Wave File	Initial File Size (byte)	Compression Result File Size (byte)	Compression Ratio	Process Length (second)
1.	Windows XP Menu Command.wav	1.404	948	67,52 %	16
2.	Ringout.wav	5.212	4.129	79,22 %	31
3.	Windows XP Balloon.wav	6.400	4.819	75,30 %	62
4.	Ringin.wav	10.026	8.208	81, 87 %	47
5.	Windows XP Ringout.wav	22.070	20.129	91,21 %	125
6.	Windows XP Minimize.wav	22.580	17.262	76,45 %	128
7.	Windows XP Recycle.wav	22.816	18.378	80, 55%	132
8.	Windows XP Hardware Remove.wav	36.538	31.795	87,02 %	188
9.	Windows XP Hardware Fail.wav	36.614	32.769	89, 50 %	190
10.	Windows XP Hardware Insert.wav	36.636	30.756	83, 95 %	192
11.	Chimes.wav	55.776	48.627	87,18 %	266
12.	Ding.wav	80.856	70.725	87,47 %	406
13.	Windows XP Logoff Sound.wav	179.704	141.211	78,58 %	735
14.	Windows XP ShutDown.wav	282.608	247.861	87,70 %	1.203
15.	Windows XP Startup.wav	424.644	368.269	86.72 %	1.781

Fig 6. Huffman Compression Testing Results

The results of testing Serpent encryption for wave files are presented as in Fig. 7.

Fig. 7. Wave File Encryption

From the results of the compression process testing, it was found that the huffman compression ratio has a range between 67.52% for the lowest value and the highest is 91.21%. The average compression ratio is 82.11%. This shows that the compressed file size is 0.8211 times the original file size or 17.89%.

V. CONCLUSIONS

Based on the discussion that has been carried out, the conclusion is:

1. The compression level is affected by the number of the same tone in the Wave file.
2. The process speed does not depend on the data that is processed but depends on the file size.
3. The decompression process is faster than the compression process because the decompression process occurs when the Huffman tree is formed.
4. The result of the file that has been encrypted does not change the file size.
5. Sound quality after experiencing compression and encryption still has good quality.

ACKNOWLEDGMENT

The author would like to thank the Rector of the Catholic University of St. Thomas, the Chair of the Institute for Research and Community Service, the Dean of the Faculty of Computer Science who has provided support for this research. Thank you as much as possible to the editorial of International Journal of Computer Trends & Technology - IJCTT who are willing to publish this journal. The author is open to suggestions and criticisms for improvement in this journal.

REFERENCES

[1] A. A. Tamimi and A. M. Abdalla, "An Audio Shuffle - Encryption Algorithm" The World Congress on Engineering and Computer Science 2014 WCECS 2014, San Francisco, USA, 2014.

[2] Aarti, "Performance Analysis of Huffman Coding Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, Issue 5, May 2013 ISSN: 2277 128X.

[3] Ahmad Jawahir & Haviluddin, "An Audio Encryption Using Transposition Method" International Journal of Advances in Intelligent Informatics, Vol 1, No 2, July 2015, pp. 98-106 ISSN: 2442-6571.

[4] Adil Jamil Zaru, Momeen Khan, "General Summary of Cryptography", Adil Jamil Zaru. Int. Journal of Engineering Research and Application ISSN : 2248-9622, Vol.08, Issue 02, (Part -2) Ffebruary 2018, pp.68-71, DOI: 10.9790/9622-080206871.

[5] Doni Ariyus, Data Security and Communication Cryptography, Yogyakarta : Andi, 2006.

[6] Hazem Kathem Qattous, **"**Hiding Encrypted Data Into Audio File", IJCSNS International Journal of Computer Science and Network Security, VOL.17 No.6, June 2017.

[7] Jaroslaw Sugier, "Implementing Serpent Cipher in Field Programmable Gate Arrays" ICIT 2011 The 5th International Conference on Information Technology.

[8] J. Lazaro, A. Astarloa, J. Arias, U. Bidarte, C. Cuadrado, "High Throughput Serpent Encryption Implementation", Field Programmable Logic and Application, Lecture Notes in Computer Science, vol. 3203, Springer, 2004.

[9] K Wahyudi, PDP Silitonga, "Cryptographic Applications for Exchange of Messages Using Steganography and AES Algorithms", Proceedings of the 2008 Teknoin National Seminar in Information Engineering, ISBN : 978-979-3980-15-7.

[10] K. Sayood, Introduction To Data Compression, Morgan Kaufmann Publishers Inc. 1996.

[11] Mai Hossam Taher, Ali E.Taki El Deen, Mohy E. Abo Elsoud, "Hardware Implementation Of The Serpent Block Cipher Using Fpga Technology", International Journal of Electronics and Communication Engineering & Technology (IJECET), ISSN 0976 – 6464, ISSN 0976 – 6472(Online), Volume 5, Issue 10, October (2014), pp. 34-44.

[12] M. Kaur and S. Kaur, "Survey of Various Encryption Techniques for Audio Data," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, pp. 1314-1317, 2014.

[13] Mamta Sharma, "Compression Using Huffman Coding", IJCSNS International Journal of Computer Science and Network Security, vol.10 No.5, May 2010.

[14] Nigam Sangwan, "Text Encryption with Huffman Compression", International Journal of Computer Applications, vol. 54, No.6, September 2012.

[15] Rinaldi Munir, Cryptography, Informatika, Bandung, 2006.

[16] Sanjali Gupta, Nikhil Shanker Mathur and Priyank Chauhan, "A New Approach to Encryption using Huffman Coding" ,International Journal of Progressive Sciences and Technologies (IJPSAT), Vol. 2 No. 2 April. 2016, pp.76 -82.

[17] Salomon, D. A Guide to Data Compression Methods, Springer, 2002.

[18] Shannon, C. E., A Mathematical Theory of Communication, The Bell System Technical Journal, vol 27, Juli, October, 2002.