International Journal of Advances in Intelligent Informatics
Vol 1, No 2, July 2015, pp. 98-106
ISSN: 2442-6571

An audio encryption using transposition method

Ahmad Jawahir[a,1,*], Haviluddin[b,2]

[a] *Migent Software Researcher, Jakarta - Indonesia*
[b] *Dept. of Computer Science, Universitas Mulawarman, East Kalimantan - Indonesia*
[1]*ahmadjawahirabd@gmail.com*;* [2] *haviluddin@unmul.ac.id*

ARTICLE INFO

Article history:
Received May 18, 2015
Revised June 22, 2015
Accepted July 3, 2015

Keywords:
Encryption
Decryption
Audio
Transposition
WAV
MSE

ABSTRACT

Encryption is a technique to secure sounds data from attackers. In this study, transposition technique that corresponds to a WAV file extension is used. The performance of the transposition technique is measured using the mean square error (MSE). In the test, the value of MSE of the original and encrypted audio files were compared; the original and decrypted audio files used the correct password is 'SEMBILAN' and the incorrect password is 'DELAPAN'. The experimental results showed that the original and encrypted audio files, and the original and decrypted audio files used the correct password that has a value of MSE = 0, and with the incorrect one with a value of MSE 0.00000428 or ≠ 0. In other words, the transposition technique is able to ensure the security of audio data files.

I. Introduction

Currently, the exchange data among users are rapidly grown so that users require to secure their data systems (i.e. video, audio, image, and text) in order to have confidentiality of the data from attackers. These security systems are widely used in the database area such as internet banking and audio communication channels. Therefore, the security systems are important aspects in information systems that many researchers continue to develop especially in the field of audio security.

Today, most security systems used are encryption techniques. Security encryption technique is a technique that uses specific algorithms to maintain data security. In principle, the process in the encryption technique is to change the data with specific algorithms so that the data become unintelligible. Therefore, if the encryption technique is applied to secure audio data, it would be difficult for the attackers to know and to hear audio contents because the sound is scrambled [1-4].

In this paper, encryption techniques using WAV file format that has a header and a data structure is applied. This research aims to create a system that can encrypt audio files without damaging the structures of the bit file. Unlike other encryption methods that use substitution and shift, this system needs to obtain an original structure of the voice data therefore there is no reduction or increase of the structure of the bit file [2, 3, 5, 6]. The principle of this system is the exchange position of the bit file structure uses a transposition technique to become audio chunks of data. If the audio data a played back, then the users can still hear the scrambled sounds. Therefore to encrypt the audio data file, the users need to enter a key that serves to influence the outcome file encryption. Meanwhile, the key length depends on the size of the audio file. The larger the file size, the larger the key length will be.

This paper will apply one method of encryption that is the transposition technique. This method will be applied to the voice data that have WAV file extension. Section 2 of this paper presents the construction of an encryption technique using the transposition algorithm. Analysis of the proposed transposition technique is in section 3. Section 4 is the results of our experiment while section 5 presents the conclusion.

II. Methodology

In general, the transposition technique consists of encryption and description. The encryption process begins with preprocessing segmentation of voice data. Each audio chunks of data are separated by intervals of bits in which the form of chunks of audio data array will be used as an input in the transposition. The next procedure is the process of transposition. This process will swap the index array for audio encrypt of the data chunks. The key is going to affect the outcome of the exchange index. After the index interchangeable, the audio chunks of data will be redeveloped into a new audio file as a result of the system's encryption. The next process is decryption of audio file from the encryption results in order to get back to the original sounds. This process requires the correct key to encrypt the audio file. However, if the key is different, then the decryption result is not going to be as similar as the original sounds.

A. WAV Document

The WAV stands for Waveform Audio Format. The WAV format is part of Microsoft's RIFF (Resource Interchange File Format) specification as a storage of multimedia files into some chunks. The WAV file consists of three parts, namely main chunk, chunk format, and data chunk [5, 7, 8]. A WAV file structure can be seen in Table 1.

Table 1. Format of WAV document

Endian	Offset Document (byte)	Attribute Name	Attribute Size (byte)	*RIFF Chunk Descriptor*
Big	0	*Chunk ID*	4	"RIFF" chunk format, the format of concern here is the "WAVE" in which requires two sub-chunk, namely: "*fmt*" and "*data*"
Small	4	*Chunk Size*	4	
Big	8			
Big	12			
Small	16	*Format*	4	
Small	20			
Small	22			
Small	24			
Small	28	*Sub-Chunk1 ID*	4	*Sub-chunk* "*fmt*" in which describes the voice information in the *data sub-chunk*.
Small	32	*Sub-Chunk1 Size*	4	
Small	34	*Audio Format*	4	
Big	36	*Num Channels*	2	
Small	40	*Sample Rate*	2	
	44	*Byte Rate*	4	
		Block Align	2	
		Bits per sample	2	
		Sub-Chunk2 ID	4	*Sub-chunk* "data"
		Sub-Chunk2 Size	4	
		Data	*Sub-Chunk2 Size*	Indicates the size of the sound information and contains raw sound data

B. Transposition Encryption

In the transposition cipher, the letters in the plaintext remain the same but the order changes. In other words, this algorithm will perform transposition the entire range of characters in the text. Other name for this method is a permutation or scrambling every character in the same text. In this study, the system that will be created by performing the transposition of columns and rows [2, 3, 5].

1) Encryption Process

The encryption process begins with reading of the original WAV file using WAV reader. The audio processing data is completed by inserting the key to separate the audio data to be encrypted and remaining unused chunk. The encrypted audio data by the column are continued with randomization by the line. The results of randomization are then recombined with the remaining chunks and subsequently written into a WAV file. The encryption process in this study can be seen in Fig. 1.

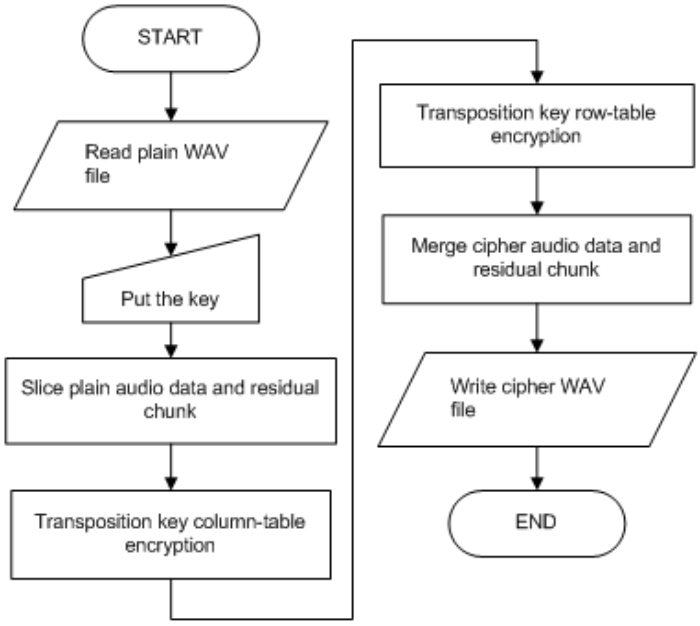


Fig. 1. Flowchart of encryption process

2) Preprocessing Audio Data and Remaining Chunks Separation

In this section, WAV reader is used to read data bit audio files. Data bits start from 0 until the end of the bit data length of audio file. Then, the interval is used to make the pieces of sounds. In this study, a sample audio file has a data length of 288,000 bytes with the value interval is 1,024 bytes so that the file has 288,000 / 1,024 = 281 chunks. When the result are written back to WAV files, the remaining bits are inserted into the output audio data. In this experiment, the remaining bits are 288 000 - (281 x 1,024) = 256 Bytes. Furthermore, after the data preprocessing has been achieved, the pieces of sounds are divided and placed in an array as can be seen in Table 2.

Table 2. Sample of audio data chunks

Array Index	First Bit	Last Bit
0	0	1,024
1	1,024	2,048
2	2,048	3,072
...	...	...
...	...	...
279	285,696	286,720
280	286,720	287,744

3) Transposition key column-table encryption

The process begins with inserting the indices of audio chunks of data into a horizontal table. The sorted key is managed and exchanged position of table column. Next, the indices of audio chunks of data is vertically read and formed into a new order indices that can be seen in Fig. 2.

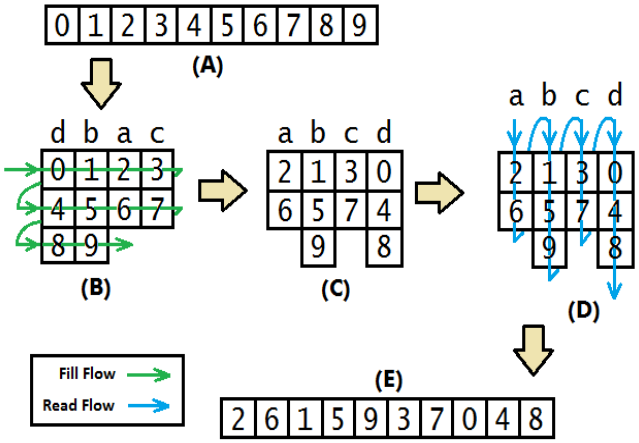


Fig. 2. Indices of audio chunks of data (A), Inserting Table Column (B), Ordering Column Based-on key character (C), Reading Table Column (D), and The results of Column Encryption (E)

4) Transposition key row-table encryption

The process begins with inserting the indices of audio chunks of data into a vertical table. The sorted key is managed and exchanged position of table row. Following this, the indices of audio chunks of data is horizontally read and formed into a new order indices that can be seen in **Fig. 3**.

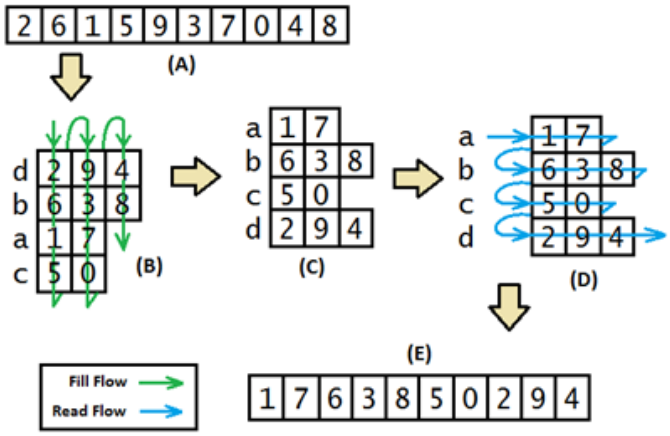


Fig. 3. Indices of Results Encryption Column (A), Inserting Table Row (B), Line Ordering Based on Key Character (C), Reading Table Line (D), and the results of the Line Encryption (E)

C. Decryption Process

Decryption process is a reversal of the encryption process. This process commences with reading encrypted WAV files using the same key of encrypted process. Next, the preprocessing of the audio data is conducted to separate the audio data to be encrypted and unused remaining chunks. The audio data are then encrypted based on the lines followed by randomization columns. The results of the randomization are then recombined with the remaining chunks and written into WAV files. The decryption process in this study can be seen in Fig. 4.

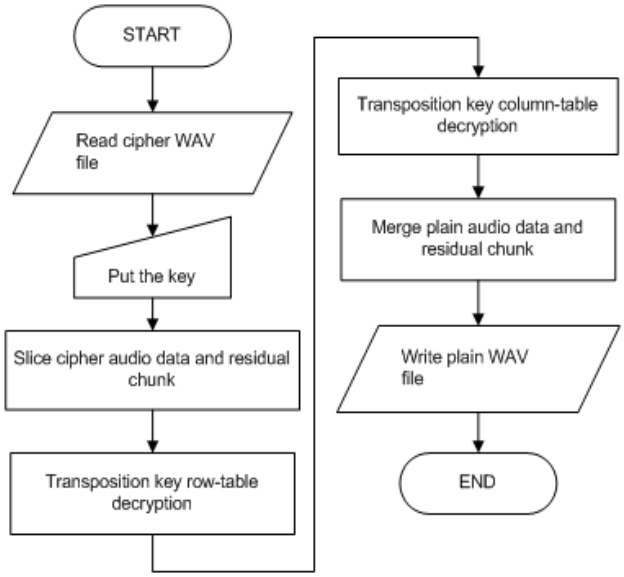


Fig. 4. Flowchart of decryption process

1) Transposition key row-table decryption

Making an array should be conducted before forming transposition tables. In this transposition method, the table cells should be bypassed therefore multiple rows or columns might incomplete. The value of *m* and *n* should be previously determined before forming the array. The *m* values could be searched with the following algorithm.

```
m = 1
While(True)
    If m x Length(Key) < Length(AudioData) Then
        Break
    End If
Loop
```

The *n* value can be found using the following calculation.

```
n = Length(AudioData) - ((m - 1) x Length(Key)
```

Thus, the same length of the array and the key will be sequentially valued. This array is labelled using a key character. Then, the value of *m* is inserted as same as the value of *n* and the rest is filled by *m - 1*. Next, the key is sorted so that the position of the array elements will also be exchanged in accordance to with the key characters. The examples of this array formation as can be seen in Fig. 5.

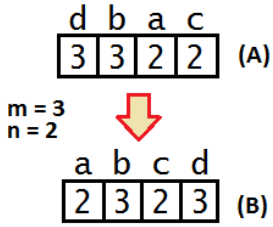


Fig. 5. Array of length of each key by unsorted key (A), Array of length of each key by sorted key (B)

The process is continued by incorporating indices of the audio chunks of data into the table vertically after getting the array. The table follows the sequenced key position. The length of lines for each character follows the key value in the array. The key is returned to its original sequence and followed by exchange table position. Next, all the audio index chunks of data are horizontally read then into a new sequence of audio data chunks, as illustrated in Fig. 6.

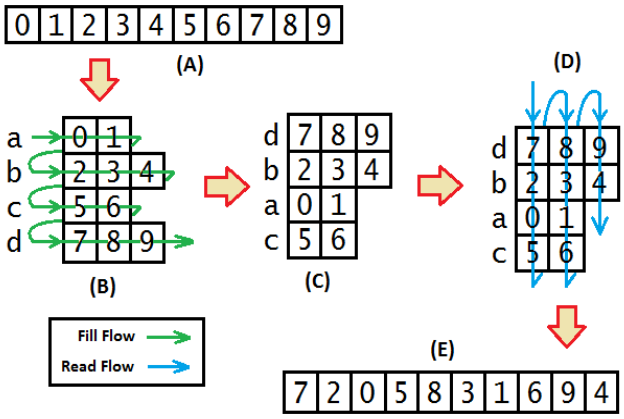


Fig. 6. Indices of audio data chunks (A), An inserting table key row (B), Line ordering based-on key character (C), Reading table lock line (D), and Decryption results classified (E)

2) Transposition key column-table decryption

From the length of the array, the process is continued by the insertion the new of audio data chunks indices which has been formed from the transposition of the key table. These indices are vertically inserted into the table. The row table follows the key sequenced positions. The column length for each character follows the key value in the array. Next, the key is returned to its original sequence and followed by exchange table position. Then, all the audio index chunks of data are horizontally read then a setup into the last sequence of indices audio data chunks, as illustrated in Fig. 7.

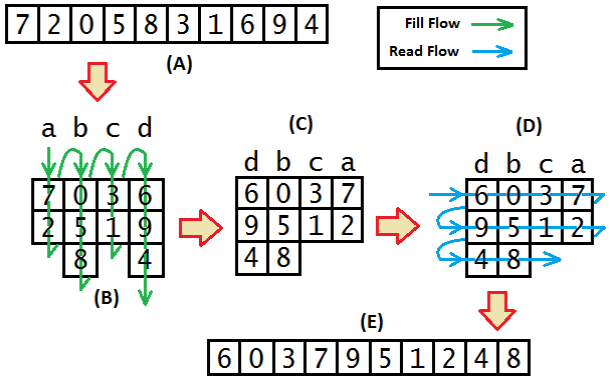


Fig. 7. Indices results line encryption (A), An Inserting table key column (B), Ordering table column based-on key character (C), Reading table key column (D), and Decryption results column (E)

D. Merger of Audio Data Chunks and Remaining Bit

The last process is the merger of indices of audio chunks and remaining bits of data. The rest of the bits that previously separated are then recombined. The results of this merger are written into WAV files using the WAV *writer*.

III. Experimental

In this experiment, the encryption and decryption program created using C # programming is used to run the transposition algorithm and to analyze amplitude sound original file using MATLAB R2013b. The original WAV data with the names of *cartoon008.wav* (http://static1.grsites.com/archive/sounds/ cartoon/cartoon008.wav) has been applied. The input of encryption and decryption program is simply made, with two buttons "*Encrypt*" and "*Decrypt*" and a "*bar*" property of WAV data. The process of the "*Encrypt*" is used to encrypt the WAV data, while the "*Decrypt*" is used to perform the decryption process of the encrypted WAV data. The

graphical user interface (GUI) of input program can be seen in Fig. 8. Then, the results of the original amplitude of the sound file are displayed on MATLAB which are illustrated in Fig. 9

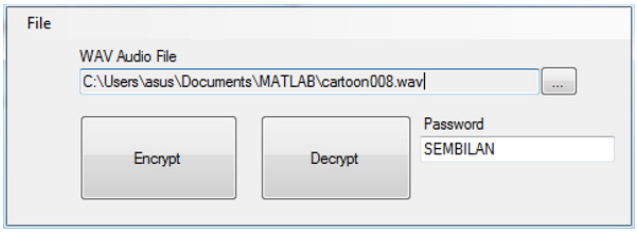


Fig. 8. GUI of input program

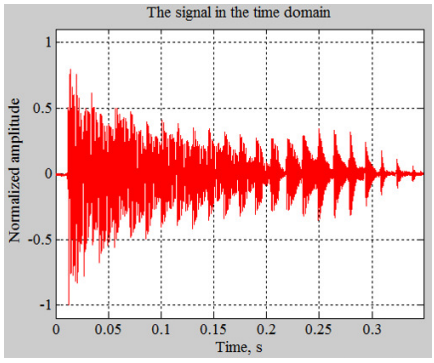
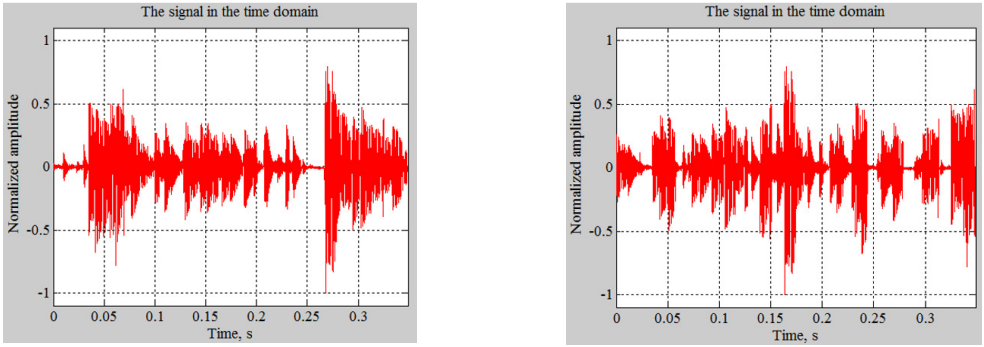


Fig. 9. The original sound of amplitude

In the first experiment, the encryption and decryption use a correct password of "SEMBILAN". The results show that the sounds return to its original. The amplitude results of sounds of encryption and decryption as illustrated in Fig. 9 and Fig. 10.(a). The results of the second experiment using the incorrect password "DELAPAN" indicate that the sounds do not return to its original but more randomized. The amplitude result shown in Fig. 10. (b).



(a) (b)

Fig. 10. (a). The amplitude result of sounds encryption by using correct-password "SEMBILAN", and (b). The amplitude result of sounds encryption by using incorrect-password "DELAPAN"

IV. Results and Discussions

This system has sounds output to provide evidence the success of the completed encryption. In this study, the MSE (Mean Square Error) is applied to prove the performance of the algorithm transposition test. The first experiment is done by comparing the MSE value between the original audio files and the encrypted audio files, then the original audio files and the decrypted audio files, in which both use the same password "SEMBILAN". If the encrypted file has a value of MSE $\neq 0$ then it indicates an exchange of index position of the audio data of WAV file. Meanwhile, if

decrypted file has a value of MSE = 0, then it indicates that the file is exactly the same as the original one. The results of the comparison WAV files as in Table 3 and Table 4.

The second test is comparing the value of MSE between the original file and encrypted file using the correct password "SEMBILAN". Then, the original and decrypted file are compared using the wrong password "DELAPAN".

Table 3. A comparison of MSE values of audio by using correct password

File Name	File Size	Vector Size	MSE
cartoon008.wav	30,832 bytes	15,394 x 1	-
encrypt.wav	30,832 bytes	15,394 x 1	0.00000362
decrypt.wav	30,832 bytes	15,394 x 1	0

Table 4. A comparison of MSE values of audio by using incorrect password

File Name	File Size	Vector Size	MSE
cartoon008.wav	30,832 bytes	15,394 x 1	-
encrypt.wav	30,832 bytes	15,394 x 1	0.00000362
decrypt.wav	30,832 bytes	15,394 x 1	0.00000428

The results of the second test display that the ratio of the MSE between the original and the decryption file do not have the same value of 0.00000428 or MSE ≠ 0. In other words, the array of the data is more scrambled in which audio and sound never return to the original.

V. Conclusions

The results of this study have shown that the transposition audio files through randomization algorithm can be used to secure the sounds files. The original sounds can be encrypted with various combinations using a password, and the results of randomization sounds can be restored to the original sounds using the correct password. The sounds will be randomized if the used password is incorrect. The use of password must be the user's concern because the results also indicate that the use of different password with the same order will make the encryption be easily solved. This could be a weakness of the transposition method. Therefore, it is recommended that users need to employ complicated password, such as long-character or mix-character passwords. Since the focus of analysis in this study is MSE, future research with similar interest could use sounds frequency analysis as a performance of the encryption and decryption.

Acknowledgment

The Authors would like to thank to Mulawarman University, Migent Software, and colleagues who have given support to complete this study.

References

[1] R. Gnanajeyaraman, K. Prasadh, and Ramar, "Audio encryption using higher dimensional chaotic map," *International Journal of Recent Trends in Engineering,* vol. 1, pp. 103-107, 2009.

[2] 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.

[3] R. Munir, *Kriptografi. 2006. Bandung: Penerbit INFORMATIKA*. Bandung, Indonesia.: Informatika, 2006.

[4] T. Xiang, C. Yu, and F. Chen, "Secure MQ coder: An efficient way to protect JPEG 2000 images in wireless multimedia sensor networks," *Signal Processing: Image Communication,* vol. 29, pp. 1015–1027, 2014.

[5] A. A. Tamimi and A. M. Abdalla, "An Audio Shuffle -Encryption Algorithm," in *The World Congress on Engineering and Computer Science 2014 WCECS 2014, 22-24 October, 2014*, San Francisco, USA, 2014.

[6] S. M. Seyedzadeh, B. Norouzi, and S. Mirzakuchakib, "RGB color image encryption based on Choquet fuzzy integral," *The Journal of Systems and Software,* vol. 97, pp. 128–139, 2014.

[7] A. M. S. Rahma and A. A. k. Maisaa, "To Modify the Partial Audio Cryptography for Haar Wavelet Transform by Using AES Algorithm," *Eng. & Tech. Journal,* vol. 32, pp. 170-182, 2013.

[8] A. G. Soriano-Sánchez, C. Posadas-Castillo, M. A. Platas-Garza, and D. A. Diaz-Romero, "Performance improvement of chaotic encryption via energy and frequency location criteria," *Mathematics and Computers in Simulation,* vol. 112, pp. 14–27, 2015.