Progressive Scrambling for MP3 Audio

Wei-Gang Fu, Wei-Qi Yan, Mohan S. Kankanhalli
School of Computing
National University of Singapore
Singapore 117543

Abstract – **Audio scrambling can be employed in audio distribution for the purpose of guaranteeing the confidentiality. Electronic commerce in audio products would be facilitated by the development of solutions which can ensure security/privacy, efficiency in compressed domain to reduce the bandwidth requirements, flexibility of implementing progressive audio quality control and computationally inexpensive to be used in real-time systems. This paper proposes an effective progressive scrambling algorithm that works with MP3 audio. We perform multiple-level scrambling based on keys to produce a set of audio outputs of differing qualities. During decoding, the keys provided and rounds of descrambling performed decide the quality of the audio output. Our experiments show that the proposed algorithm is effective, fast, simple to be implemented, and at the same time it does provide flexibility to control the quality of audio output progressively. We also present some experimental results to show its utility in real-time systems.**

I. INTRODUCTION

Audio scrambling aims to minimize the residual intelligibility of the original audio and control the access to only authorized users. It is similar to but not a direct application of normal cryptographic techniques. The main advantage is that scrambled audio are still validly formatted which can be played by the corresponding players.

With the wide-spread use of the MP3 audio format, the need for developing a new scrambling algorithm that can work in the compressed domain has become important. The past research is based on permutation in either the frequency or temporal domain [1] for normal audio data. However, not much work has been done in the compressed domain. MP3 is mainly used for online music distribution. There is a need for allowing the music owners to have flexible control over their music. An algorithm that allows them to provide music of varying quality to different users and which at same time, is able to protect the copyright for the MP3 files that can be very useful.

In this paper, we propose a progressive algorithm that does multiple rounds of scrambling of MP3 audio. We reconstruct the MP3 outputs of different qualities based on number of keys provided and rounds of scrambling performed. Each de-scrambling is based on the results of the previous step. This work is one part of our research program on media security. We had earlier developed a compressed-domain video scrambling technique [10]. In this paper, we focus on the audio domain.

The rest of paper will be structured as follows: section II will introduce the background, section III will present our proposal, section IV provides the results and the security analysis. The final section draws the conclusion and outlines the future work.

II. BACKGROUND

A. Overview of MP3

MPEG audio is a family of open standards for portable audio with the high commercial value which includes MP2, MP3 and AAC (Advanced Audio Coding). MP3 stands for MPEG (Moving Picture Experts Group Technology) Layer-III. Its file structure, as shown in Fig. 1, can be expressed in this scheme (TAGs are optional):

[TAG v2] Frame1 Frame2 Frame3,…,[TAG v1]

An MP3 file is made up of frames having a constant duration of 0.026 second. The size of one frame (in Bytes) varies according to the *BitRate*. The first four bytes of each frame are the frame header and the rest is audio data. MP3 frame header consists of information about frame, such as *BitRate*, *SampleRate* etc, each of them has its own characteristics. The MP3 specification (ISO –11172-3) does not specify on how the MP3 encoding shall be done. It only specifies the resulting output format. Developers are supposed to develop their own algorithms to meet the requirements [2]. There are many different MP3 encoders available, amount which the most popular standard is LAME. These MP3 encoders perform lossy compression by discarding the information that does not significantly contribute to the signal perception.

On top of that, these MP3 encoders minimize coding redundancy using “Huffman Coding”. Huffman coding is a byte-substitution scheme that replaces the more frequently used values with shorter bytes and less frequently used values with longer bytes, by doing this it can achieve an overall reduction in the data volume.

B. Related Work

Many algorithms have been proposed for audio scrambling.

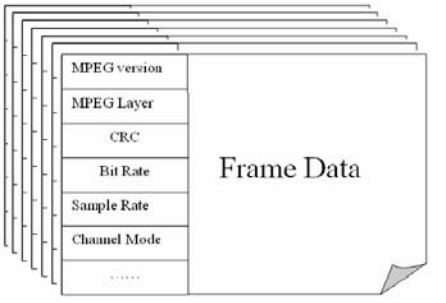


Fig. 1. Frame structure of MP3

Borujeni [1] presents a fast Fourier transform (FFT) related algorithms to deal with the permutation of FFT (Fast Fourier Transform) coefficients in a speech encryption system, which is able to provide a high level of security. Matsunaga et al [3] present another FFT based algorithm which is accompanied by adaptive dummy spectrum insertion and commanding operation, and prove that the dummy spectrum insertion will further enhance the security.

Farkash S. et al. [4] present an algorithm that modifies the Gabor representation of the input speech signal. The proposed scheme can perform any invertible modifications, and it does not need the commonly used permutation operations. Hadamard approaches are widely employed in audio scrambling [5][6]. Milosevic et al. [7] present a new algorithm that makes use of Hadamard matrices to perform scrambling. It not only changes the sequence of the data elements, but also the values. Thus it enhances security, but at the cost of increased complexity.

Wavelet transform tends to create low-value coefficients at the finer scale. Ma et al. [8] use wavelet to scramble analog signals in 2D space, it is highly secure in temporal and spatial domain. Kadambe [9] uses adaptive wavelet for speech scrambling. However, there has been no work on compressed domain audio scrambling.

Kankanhalli et al [10] have proposed a joint encryption and compression framework that works on compressed-domain video (MPEG video). The algorithm shuffles the Huffman code words and provides reasonable security level in video scrambling. In this paper, our work is different from other audio scrambling algorithms in two aspects – it works directly in the compressed domain and it supports progressive scrambling.

III. PROPOSED ALGORITHM

The idea behind our proposal is to scramble the Huffman code words in the MP3 file. However, instead of directly shuffling the Huffman code word table, we operate on the actual MP3 data, this is to eliminate the repeats of values, i.e. one value in the original audio will be shuffled to different values, which will break the statistical correlation between the source audio and the output. This is very important for thwarting statistical cryptanalysis.

Progressive quality levels are attained by conducting multiple scrambling. A few rounds of scrambling are required and the quality degrades gradually with each scrambling. By performing the inverse descrambling processes, audio with different qualities will be generated.

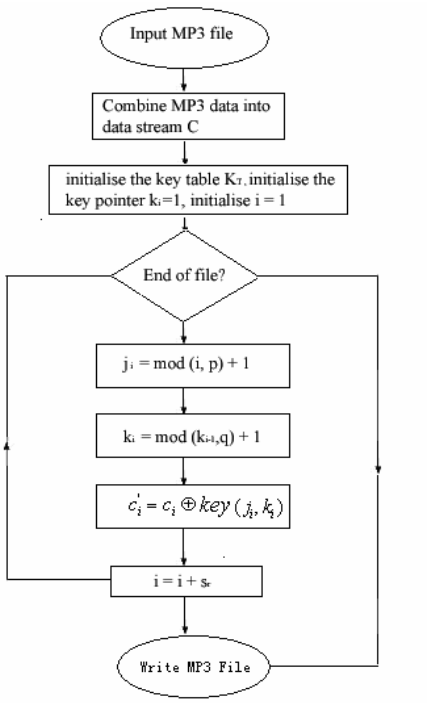


Fig. 2. Flowchart of proposed algorithm for MP3 audio scrambling

The procedure of MP3 audio scrambling/descrambling is explained in Fig. 2. Given an MP3 audio, we ignore the frame headers and only operate on the audio data $C = \{c_i, i = 1,2, \dots, n\}$, n is the length of the audio stream. A sample rate s_r needs to be predefined before each round of scrambling. To make the quality degradation in a linear way, the sampling rate $s_r(l)$ needs to be set at an exponential increment rate for different rounds.

$$s_r(l) = a^l, a \in Z \tag{1}$$

where l is the scrambling level number, $l = 1,2, \dots, L$. L is the total number of scrambling rounds to be conducted. Also a k_T will be initialized; the format of k_T is shown as follows:

$$K_T = (key(i, j))_{p \times q} \tag{2}$$

The key table generation makes use of special techniques to ensure the uniqueness and randomness of the key tables generated. For the keys used in this scheme, we employ the Arnold matrix [11] to generate random $p \times q$ matrix using an

input random generated seed, here we assign p=q. The Arnold matrix is shown as follows:

$$A = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 2 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & 2 & \cdots & p \end{bmatrix} \quad (3)$$

We first generate a random matrix $R=(r_{i,j})_{p \times q}$

$$r_{i,j} = rand(range), i = 1,2,..., p;\ j = 1,2,..., q \quad (4)$$

where $rand(\cdot)$ is a function to generate a random number sequence based on the seed *range*. Then we generate our K'_T by doing a matrix product of the Arnold matrix [11] by:

$$K_T = \begin{bmatrix} 1 & 1 & ... & 1 \\ 1 & 2 & ... & 2 \\ ... & ... & ... & ... \\ 1 & 2 & ... & p \end{bmatrix} \bullet R \ (\text{mod } p) \quad (5)$$

We use the Arnold matrix to transform the keys in the key table to the different kinds of choices for different users. The transformation based on this matrix can ensure the validity of the keys since they are bounded.

For the *i-th* data item c_i in the original audio, we do a modulo division of the current position i with the row length p of the key table and the remainder decides the row number j of the key to be used.

$$j_i = \text{mod}(i, p) + 1 \quad (6)$$

The column number k_i will cyclically range from the first column to the maximum column number q.

$$k_i = \text{mod}(k_{i-1}, q) + 1 \quad (7)$$

After we select the key $k_i = key(j_i, k_i)$, we perform an XOR operation with the MP3 audio datum c_i. XOR is chosen due to the speed and simplicity of its implementation. The changed datum c'_i is written into the corresponding position of the output stream:

$$c'_i = c_i \oplus key(j_i, k_i) \quad (8)$$

where $key(j_i, k_i)$ is the element in matrix K_T, $\oplus$ is the bit-wise XOR. This procedure is repeated till the end of the data stream and the output is also an MP3 stream. The keys used will be stored and they will be distributed to the authorized consumers for descrambling.

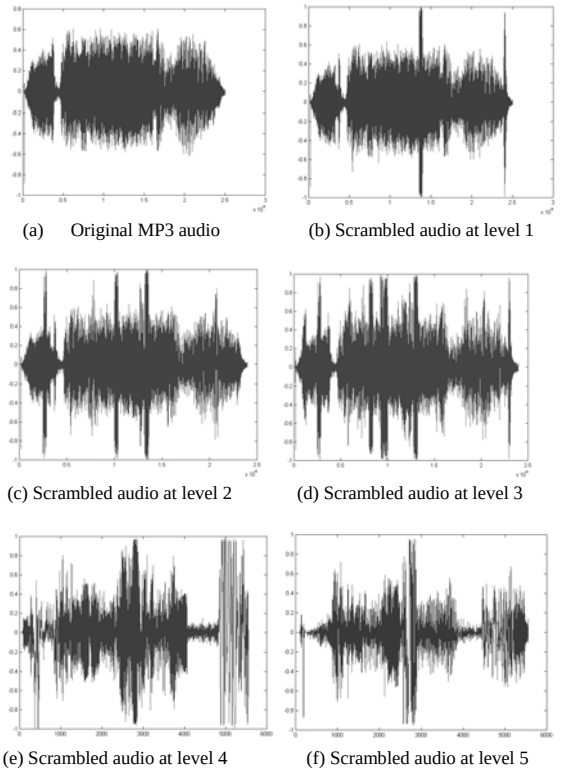
The descrambling process is exactly similar to the scrambling procedure. For descrambling level l, a MP3 file generated from descrambling level l-1 will be used as an input. We apply the same process and use the same K_T that has been used in the scramble round l.

With the implementation of the scrambling process as described above, the quality of the output MP3 file depends on how many descrambling levels involved. For example, during scrambling if an MP3 audio has been scrambled 5 times using 5 different keys, a user needs to have all the 5 keys and perform 5 rounds of descrambling before they can perfectly restore the original audio. If the user has only 4 keys or performs only 4 rounds of descrambling, he will get an audio with a small degradation of the original quality.

IV. Results and Analysis

In this section, we conduct experiments to evaluate the effectiveness of the proposed algorithm and prove the feasibility of our scheme.

Figure 3 depicts the waveform for an audio file at different scrambled levels. Figure 3(a) is the waveform of source MP3 audio, we scramble the audio 5 times and obtain Fig. 3(f). We descramble Fig. 3 (f) to get Fig.3 (e), which is of slightly better quality, we further generate Fig. 3 (d) to Fig.3 (b) by performing additional rounds of descrambling. The waveforms show that the output is gradually reconstructed and eventually we will get an audio that is exactly same as the source.



(a) Original MP3 audio (b) Scrambled audio at level 1

(c) Scrambled audio at level 2 (d) Scrambled audio at level 3

(e) Scrambled audio at level 4 (f) Scrambled audio at level 5

Fig. 3. Original and scrambled MP3 audio waveforms at different scrambled level

Subjective evaluation is a reasonable approach to measure the scrambling levels. In our work of MP3 audio scrambling, we conduct an evaluation of four audios of different categories, namely classic, pop, jazz and rock music.

Each music clip has been scrambled to five levels of degraded qualities. Seven investigators participated in this

poll, the feedbacks with the average scores are listed in Table I.

Table I: Audiences' score for different quality levels
(6 for the best quality – 1 for the worst quality)

Quality	POP	JAZZ	ROCK	CLASSIC
Level 6	6.00	6.00	6.00	6.00
Level 5	4.14	4.57	5.00	5.00
Level 4	3.86	3.71	3.29	3.86
Level 3	3.86	3.00	3.00	2.86
Level 2	1.71	2.29	2.14	1.71
Level 1	1.29	1.57	1.57	1.57

Table I shows that most of the listeners are able to perceive the gradually decrease in quality as the amount of scrambling increases. They quite accurately identify the correct sequence of the scrambled audio, especially in the better quality levels. We also calculate the computation time required for scrambling/descrambling at each level, which is shown in Table II.

As our scrambling has been done at different sampling rates for different levels and the amount of computation is closely related to the number of samples, the time taken increases with the sampling rate. (e.g. for classic music time consumption for scrambling increases from 0.94 second at level 1 to 4.59 seconds at level 5).

Table II. Time consumption for MP3 scrambling / descrambling

Audio Size /length	Operations	**POP** 3kb / 2sec	**JAZZ** 23 kb / 22sec	**ROCK** 10kb / 9 sec	**CLASSIC** 32kb / 32 sec
Level 1	Scrambling	0.82 sec	0.91sec	0.84 sec	0.94 sec
	Descrambling	0.81sec	0.90sec	0.85 sec	0.94 sec
Level 2	Scrambling	1.02sec	1.01 sec	0.92 sec	1.12 sec
	Descrambling	0.79sec	0.97 sec	0.84 sec	1.04 sec
Level 3	Scrambling	1.27sec	1.52 sec	1.34 sec	1.58 sec
	Descrambling	0.84sec	1.04 sec	0.91 sec	1.15 sec
Level 4	Scrambling	1.3sec	1.67 sec	1.43 sec	1.86 sec
	Descrambling	0.87sec	1.23 sec	0.99 sec	1.41 sec
Level 5	Scrambling	1.55sec	3.57 sec	2.25 sec	4.59 sec
	Descrambling	1.07sec	3.07 sec	1.77 sec	4.1 sec

The result shows that our method is fast. For an MP3 audio around 30 seconds, it needs only 8.7 seconds for the scrambling and descrambling processes, which is much shorter than the music duration. Thus the proposed algorithm can potentially be applied in real-time systems.

In our scheme, the security of our scheme relies on the key table. Range of values allowed in the key table determines the key space. The amount of calculation for attacker to break the whole MP3 file of length L by brute force is calculated as following: given an MP3 audio of duration 30 seconds and data length of 3.2×10^4 bytes, assume that the key table allows random values within the range from 1 to 16 and the key table is of size 16 × 16. An adversary has to complete 32000×16^{256} tries for bit permutation before they can restore the perfect audio. This will take more than 10^{250} years at a rate of 10^8 instructions per second. However, this can potentially be reduced by studying audio coherence for domain-specific cryptanalysis. Since the purpose of this proposal is to protect music clips, the security level appears to be sufficient. Incidentally, this scheme has no size blow-up. The original and scrambled audios are of exactly the same size.

V. CONCLUSION

In this paper, we have proposed a progressive audio scrambling and descrambling scheme. Our experiments show that the effectiveness of the proposal. It is a fast and simple solution, yet it can provide sufficient security for musical products. Our future work will be on enabling traitor tracing in the scrambling algorithm and on utilizing watermarks as keys to manage audio copyright.

Acknowledgement: Wei-Qi Yan's work in this paper is partially supported by a fellowship from Singapore Millennium Foundation (SMF).

REFERENCES

[1] S.E Borujeni., "Speech encryption based on fast Fourier transform permutation," in Proc. of The 7th IEEE International Conference on Electronics, Circuits and Systems, Dec. 2000, vol. 1, pp. 290–293.

[2] Y. Wang, W.D. Huang and K. Jari, "A framework for robust and scalable audio streaming," in Proc. of ACM Multimedia 2004, Oct. 10-16, 2004.

[3] A. Matsunaga, K. Koga, and M. Ohkawa, "An analog speech scrambling system using the FFT technique with high level security," IEEE Transactions on Selected Areas in Communications, 7(4), May 1989, pp. 540–547.

[4] S. Farkash, S. Raz, and D. Malah, "Analog speech scrambling via the Gabor representation," in Proc. of the 17th Convention of Electrical and Electronics Engineers in Israel, 1991, pp. 365 – 368.

[5] V. Senk, V.D. Delic, and V.S. Milosevic, "A new speech scrambling concept based on hadamard matrices," IEEE Signal Processing Letters, Jul. 1997, vol. 4, pp. 161–163.

[6] Y. Wu and B. P. Ng, "Speech scrambling with hardamard transform in frequency domain," in Proc. of IEEE ICSP, 2002, pp. 1560-1563.

[7] V. Milosevic, V. Delic, and V. Senk, "Hadamard transform application in speech scrambling," in Proc. of 13th International Conference on Digital Signal Processing, 1997, pp. 361 – 364.

[8] F.L. Ma, J. Cheng and Y.M. Wang, "Wavelet transform based analogue speech scrambling scheme," Electronics Letters, 32(8), Apr. 1996, pp.719 – 721.

[9] S. Kadambe and P. Srinivasan, "Application of adaptive wavelets for speech coding," in Proc. of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis, Oct.1994, pp. 632–635.

[10] M.S. Kankanhalli and T.G. Teo, "Compressed domain scrambler and descrambler for digital videos," IEEE Transactions on Consumer Electronics, 48(2), May 2002, pp. 356–365.

[11] D.X. Qi, J.C. Zou, X.Y. Han, "A new class of scrambling transformation and its application in the image information coveringT," Science in China (series E), 43(3), Jun. 2000, pp.305-311.