Repositorio Institucional de la Universidad Autónoma de Madrid

https://repositorio.uam.es

Esta es la **versión de autor** del artículo publicado en:
This is an **author produced version** of a paper published in:

Multimedia Tools and Applications 71.3 (2014): 1803 – 1822

DOI: http://dx.doi.org/10.1007/s11042-012-1306-7

Copyright: © 2014 Springer US

El acceso a la versión del editor puede requerir la suscripción del recurso
Access to the published version may require subscription

*Manuscript
Click here to download Manuscript: hiary_mtap2012.tex
Click here to view linked References

Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Audio Scrambling Technique Based on Cellular Automata

Alia Madain · **Abdel Latif Abu Dalhoum** · **Hazem Hiary** · **Alfonso Ortega** · **Manuel Alfonseca**

Received: date / Accepted: date

Abstract Scrambling is a process that has proved to be very effective in increasing the quality of data hiding, watermarking, and encryption applications. Cellular automata are used in diverse and numerous applications because of their ability to obtain complex global behavior from simple and localized rules. In this paper we apply cellular automata in the field of audio scrambling because of the potential it holds in achieving a high scrambling degree. We also analyze the effect of using different cellular automata types on audio scrambling and we test different cellular automata rules with different Lambda values. The relation between the robustness and the scrambling degree is also studied. Experimental results show that the proposed technique is robust to data loss attack and can be applied to different applications based on the scrambling degree required.

Keywords Audio Scrambling · Cellular Automata · Game of Life · Lambda Parameter

1 Introduction

The term audio scrambling has a long history. A century ago, audio scrambling was the only way to hide analog audio information transmitted, and the scrambling relied mainly on altering the audio signal in the time domain, the frequency domain, or both. Later, scrambling of other media types was used. For example, scrambling techniques were used in copyright protection of cable TV broadcast; secure image transfer from satellites to ground stations, and military communications [1].

Alia Madain, Abdel Latif Abu Dalhoum, Hazem Hiary
University of Jordan, Amman 11942, Jordan
Tel.: +962-6-5355000, Ext.: 22575
E-mail: hazemh@ju.edu.jo

Alfonso Ortega, Manuel Alfonseca
Universidad Autónoma de Madrid, Madrid, Spain

With the rapid development of information technology, the applications of audio scrambling varied significantly; although it is still used in the field of security, it is considered as a pre-process or post-process of watermarking, information hiding, fingerprinting, and encryption.

The techniques that use scrambling have many applications; for example, Fingerprinting is one of the effective means of copyright protection of multimedia transmitted to massive users using the multicast method [2], the development of robust data hiding system helps more technologies find new and promising applications [3], and watermarking is one of the methods used in Intellectual Property (IP) protection which is an important element in multimedia transmission and delivery systems [4].

In our proposed algorithm we aim to get a high scrambling degree using cellular automata (CA). In addition, this work seeks to find out how to best benefit from cellular automata characteristics, what types will lead to better scrambling and analyzing the complex and chaotic behavior of CA in terms of audio scrambling. We also test the robustness of cellular automata techniques against data loss attack where $1/3$ the data is lost.

The remaining of this paper is organized as follows: in section 2 we review related work briefly; section 3 covers essential cellular automata background information; this is followed by a description of the scrambling algorithm proposed and the scrambling degree measurements in section 4; section 5 gives the analysis and discussion of the experimental results; and finally, in section 6, we conclude the work done and discuss possible directions for future work.

2 Related works

A lot has been done in the field of scrambling. For digital images, many scrambling methods are available, such as those based on Arnold transformation [5], Advanced Encryption Standard (AES) and error-correcting code [6], cat chaotic mapping [7], and dynamic twice interval-division [8], among others.

Cellular automata have also been used for digital image scrambling: Ye and Li [9] described the use of CA with chaotic behavior, while Abu Dalhoum et al. [10] proved that CA with complex behavior scrambles better than those with chaotic behavior. In this paper we extend work done in digital image scrambling and apply it to digital audio.

Many researchers have worked on audio scrambling: the work done in [11,12] and used in [13], for example, uses variable dimension operation to address problems in one dimensional linear mapping. The algorithm changes the dimension of coordinates and uses a transformation matrix to scramble the original audio. The process requires padding with zeros, and calculating the new position is highly dependent on the dimension.

In [14], audio scrambling depends on combining two algorithms. The combined algorithm takes two positive integers as keys; the first is used in one of the algorithms, while the other determines the execution sequence of the two algorithms.

Our main goal in this paper is to achieve a high scrambling degree by breaking the correlations between audio samples, and to get a high degree of confusion and diffusion. The algorithm proposed uses 2D cellular automata without any additional padding; moreover the key values are independent from the audio file.

3 Cellular Automata

Cellular automata (CA) are simple and highly parallel models of computation which can exhibit complex behavior [15]. They are used in many applications covering different fields, for example, to model complex dynamic systems in chemistry and biology [16], as in [17] and [18].

This paper uses two dimensional cellular automata. The model implies a grid of identical cells, each cell can be in one of two states, zero or one (also called *on* or *off*, *dead* or *alive*). From a given initial state, the state of the grid cells changes from one iteration (generation) to the next, or stays the same, based on a transition function (or rule) which depends on the state of the specified cell and the states of its neighbors in the previous generation. The transition function is applied simultaneously to every cell in the grid.

3.1 Cellular automata neighborhood and boundary condition

The actual neighborhood chosen is usually crucial for the global behavior of a CA [19]. In this paper we consider two common 2D neighborhoods, von Neumann and Moore. In von Neumann's neighborhood every cell has four neighbors: the cells at its North, South, East, and West, whereas in Moore's neighborhood the cells at the four diagonals are also considered.

In an infinite grid, every cell has a full neighborhood, but in a finite grid there must be a way to handle cells on the edges. We will consider both *null* and *periodic* boundary conditions in our experiments. A CA is said to have a null boundary if the left (right) neighbor of every leftmost (rightmost) cell is supposed to be neighbor to a cell in the zero state, and a periodic boundary if the extreme cells at opposite sides are adjacent to one another [20].

3.2 Cellular automata Lambda parameter (λ)

Not all types of cellular automata result in a complex behavior and there are many attempts to group cellular automata with the same behavior. According to Wolfram, it seems that the patterns which arise from different types of cellular automata can almost always be assigned to one of just four basic classes [21]. In *class*1, patterns evolve into a stable, homogeneous state; *class*2 patterns evolve to periodic state; in *class*3 a chaotic behavior appears; and in *class*4, configurations contain structures which interact in complex and interesting ways.

A way to describe the relation between the transition rules and the behavior of the CA is the λ parameter, defined by Chris Langton [22]. A small value of λ indicates that the CA evolves to a stable state, which corresponds to Wolfram *class*1; a higher value describes a periodic behavior, as in *class*2; for values close to 1, the CA behavior tends to be chaotic, as in *class*3; for some critical values of λ, the CA exhibits complex behavior, as in *class*4. Ideally, all transition functions with the same λ exhibit similar behavior [23].

3.3 Calculating the Transition Rule Number

In [9] the Rule number (C) is calculated as follows:

$$C = \sum_{s=0}^{1} \sum_{n=0}^{8} f(s,n) \times 2^{2n+s} \qquad (1)$$

where f is the transition function, s is the current state at time t and n is the number of neighboring cells with $s = 1$. The transition function f produces the state s at time $t+1$, if the combination between the current state at time t and the neighboring cells with $s = 1$ results in $s^{t+1} = 1$, then the amount 2^{2n+s} is added to the rule number. The Moore neighborhood is assumed in this equation.

3.4 Conway's Game of Life (GOL)

The rules of Conway's game of life are simple [24], and can be described as follows:

– Survival. If a cell is alive at time t, it will remain alive at time $t+1$ if and only if it has either two or three neighbors alive at time t.
– Birth. If a cell is dead at time t, it becomes alive at time $t+1$ if exactly three of its neighbors are alive at time t.
– Death. If a cell is alive at time t, it will die at time $t+1$ if less than two (isolation) or more than three (overpopulation) neighbors are alive at time t.

4 Audio scrambling based on cellular automata

The technique we are proposing can be divided into two processes: first the audio is scrambled and used as the base for further processing, or perhaps sent directly; then the audio is descrambled to generate the final version for distribution (in case of watermarking) or to retrieve the ciphered information (in case of encryption). Both the scrambling and descrambling processes depend on the *key generation process* to produce the indices used for scrambling.

4.1 Scrambling algorithm (ASCA)

The scrambling algorithm (ASCA) takes the audio file as input and produces a scrambled audio plus a key as the output. The key specifies the indices used for scrambling. The procedure starts by determining the length of the audio file which will be used to increase the dimension of the audio, then a list of the new indices is generated using rules of the game of life, a Moore neighborhood and a periodic boundary. This list is used to fill the two dimensional matrix with the original audio samples, then the dimension is decreased and the scrambled audio file is written with the same sample rate and number of bits per sample as its original. The algorithm can be described as follows:

(1) Read audio file X and determine its length: $length(X)$.
(2) Select M such that $(M-1)^2 < length(X) < M^2$. Build an $M \times M$ empty matrix A.
(3) Calculate the last index (in row first order) occupied by the wave samples if they were inserted into a matrix of size $M \times M$. All positions before this point are within range.
(4) Get the new list of indices L, using the key generation process described in section 4.2.
(5) Initialize a pointer to point to the first cell in the audio array: $ptr = 1$.
(6) For each index in L, if it is within range, then $A[index] = X[ptr]$. Increment ptr.
(7) While ptr is less than $length(X)$, insert the remaining values in A, in row first order using the same procedure.
(8) Decrease the dimension of A from 2D to 1D.
(9) Repeat steps 5 to 8, *N times* if a higher scrambling degree is required by the application.

After the scrambling process, the scrambled audio file R is written in the same format, with the same sample rate and number of bits per sample. Fig. 1(a) shows an audio wave file, Fig. 1(b) shows a list of indices retrieved from the key generation process described in section 4.2, Fig. 1(c) shows the audio file scrambled before decreasing the dimension. The length of the key is less than the length of the audio, so the remaining values are inserted in row first order. The black cell is out of range and will be discarded when the dimension is decreased.

-0.0820
-0.1017
-0.1222
-0.1435
-0.1657
-0.1886
-0.2122
-0.2362

(a)

(1,3)
(3,1)
(1,1)
(2,3)
(1,2)
(2,2)

(b)

-0.1222	-0.1657	-0.0820
-0.2122	-0.1886	-0.1435
-0.1017	-0.2362	■

(c)

Fig. 1 Audio Scrambling Process, (a) Original Audio, (b) List of New Indices, (c) Scrambled Audio

4.2 Key generation process

The key generation process is used by the scrambling and descrambling processes. It takes the value of M calculated in section 4.1, and produces a list of indices L. The key generation process is based on 2D cellular automata and can be described as follows:

(1) Create a CA with an $M \times M$ grid and a random initial state configuration.
(2) Initialize to zero a status $M \times M$ matrix B.
(3) Run the Game of life rules (using Moore neighborhood and a periodic boundary) for (NOG) generations starting from the initial state configuration. Subsequent state configurations are $K_1, K_2, \cdots, K_{NOG}$.
(4) For $k = 1, 2, \cdots, NOG$, if $K_k[i, j] = 1$ and $B[i, j] = 0$, add (i, j) to the list of new indices L, and set $B[i, j]$ to 1.
(5) Return L.

The number of generations (NOG) must not exceed the number needed to scramble the whole audio file, in most applications a small number of generations is enough to get a high scrambling degree (as shown in the experiments in section 5.1), because the difference in the scrambling degree decreases when the number of generations increases, so there is little benefit from running the algorithm for too many generations.

4.3 Descrambling Algorithm

The descrambling algorithm is the inverse of the scrambling algorithm; it simply takes the scrambled audio file generated by the scrambling algorithm and the initial configuration of CA to return the original file. If the algorithm is repeated or the number of generations is changed, then the algorithm requires NOG and N. Note that the descrambling algorithm can use the initial configuration to reproduce the key, and $K_1, K_2, \cdots, K_{NOG}$ are not required. If no attacks or audio processing operations were made to the scrambled file, the algorithm returns a file identical to the original.

4.4 Measuring the Scrambling Degree

We have used the average scrambling degree measurement proposed in [9] for image files to analyze the effect of different cellular automata types on audio scrambling. In order to use it, two important things must be considered: first, the audio file amplitude contains negative values and needs to be normalized; second, audio and image files are different, so the difference should be computed in a different way, as shown in the equations below. Let $P(i)$ denote the original audio data, and N is the length of the audio file, then the difference D for cell (i), is calculated as follows:

$$D(i) = \frac{1}{4}\sum_{\acute{i}} \left[P(i) - P(\acute{i})\right]^2 \tag{2}$$

where $(\acute{i}) = \{(i-1),(i-2),(i+1),(i+2)\}$. After computing the cell difference, the mean difference M for the audio is calculated as:

$$M = \frac{\sum_{i=3}^{N-2} D(i)}{N-4} \tag{3}$$

Finally the scrambling degree SD is defined as:

$$SD = \frac{\grave{M} - M}{\grave{M} + M} \tag{4}$$

Where $\grave{M}$ is the mean difference of the scrambled file and M is the mean difference of the original audio file; the value of the scrambling degree in Eq. 4 ranges from -1 to 1, where higher values indicate better scrambling.

5 Experimental Results and Analysis

In this section, we study different types of CAs in terms of audio scrambling, and we evaluate the robustness of the proposed algorithm against data loss attack, focusing on the relation between the algorithms robustness and the scrambling degree. We also compare the algorithm with three different algorithms.

The data set used to study the behavior of CAs contains 16 audio files with different waveforms. The resolution of all the audio files used is 16 bit and the audios are in WAVE format. Table 1 shows the details of the audio files. Those numbered from one to eight are speech audio files; those numbered from nine to sixteen are music audio files.

Although some of the audio files have multiple channels, only one channel is used in the experiments.

Table 1 Details of Audio files used to study CA behavior

Audio file	Duration (seconds)	Sample rate (Hz)	Bit rate (kbps)	Channels
1.wav	0.810562	16000	256	1
2.wav	9.87994	16000	256	1
3.wav	2.75594	16000	256	1
4.wav	3.95619	16000	256	1
5.wav	2.06294	16000	256	1
6.wav	1.63025	16000	256	1
7.wav	3.85844	16000	256	1
8.wav	2.32894	16000	256	1
9.wav	1.9805	44100	705	1
10.wav	1.83991	44100	705	1
11.wav	4.40367	48000	1536	2
12.wav	8.80733	48000	1536	2
13.wav	8.80733	48000	1536	2
14.wav	1.84154	44100	705	1
15.wav	2.18181	44100	1411	2
16.wav	0.901859	44100	1411	2

5.1 Correlation Analysis of the Number of Generations (NOG)

The scrambling key in the proposed algorithm depends on cellular automata, so the number of generations parameter is vital to produce keys. Table 2 shows the experiments made using the data set described. The experiments were made with no repetition ($N = 0$). From Table 2 it can be seen that the greater the NOG, the better the scrambling degree obtained.

Based on the proposed algorithm, the scrambling degree increases as the number of generations increases, because if no more indices are specified by GOL rules the algorithm will insert the remaining values in a row first order (step 7 of the scrambling algorithm in section 4.1), which will make

Table 2 Different NOGs effect on scrambling degree

Audio	**Number of Generations (NOG)**					
file	**1**	**5**	**10**	**15**	**20**	**25**
1.wav	0.83620	0.92165	0.92573	0.92747	0.92875	0.92888
2.wav	0.86472	0.91528	0.92161	0.92549	0.92788	0.92876
3.wav	0.87451	0.93195	0.93663	0.93893	0.94077	0.94120
4.wav	0.78555	0.89342	0.90488	0.91776	0.92102	0.92423
5.wav	0.84423	0.90007	0.91566	0.91871	0.91884	0.91963
6.wav	0.81119	0.89137	0.89788	0.90219	0.90362	0.90404
7.wav	0.82025	0.90550	0.91556	0.92028	0.92228	0.92243
8.wav	0.80172	0.89939	0.90461	0.90673	0.90768	0.90909
9.wav	0.99794	0.99883	0.99885	0.99895	0.99898	0.99899
10.wav	0.99091	0.99468	0.99535	0.99574	0.99580	0.99586
11.wav	0.87958	0.92720	0.93791	0.94070	0.94394	0.94440
12.wav	0.90406	0.94644	0.95319	0.95558	0.95684	0.95765
13.wav	0.86702	0.90716	0.92831	0.93027	0.93052	0.93075
14.wav	0.87101	0.91189	0.91424	0.92479	0.92690	0.92691
15.wav	0.97422	0.98468	0.98687	0.98728	0.98743	0.98761
16.wav	0.88890	0.92236	0.93936	0.93952	0.93959	0.93966

the correlation between the samples higher and the scrambling degree lower, especially when there are gaps between the indices specified by the key.

Scrambling 1.wav for one generation produces an audio file scrambled with the degree 0.83620. By increasing the number of generations to five, the scrambling degree goes up significantly to 0.92165. Increasing the number of generations to twenty, the scrambling degree increases to 0.92875. The difference between using one generation and five is 0.08545 whereas the difference between five generations and twenty is 0.0071. The experimental results suggest that the influence of changing the number of generations decreases gradually when the number of generations increases, because when the key becomes longer, less values are inserted in order. The increase also stops when the length of the key is equal to the length of the audio file or when the whole audio file is scrambled.

In all the following experiments we use fifteen generations. Fig. 2 shows the test audio files scrambled for 15 generations with $N = 0$. It can be seen that the scrambled waves have very different shape and form from their original.

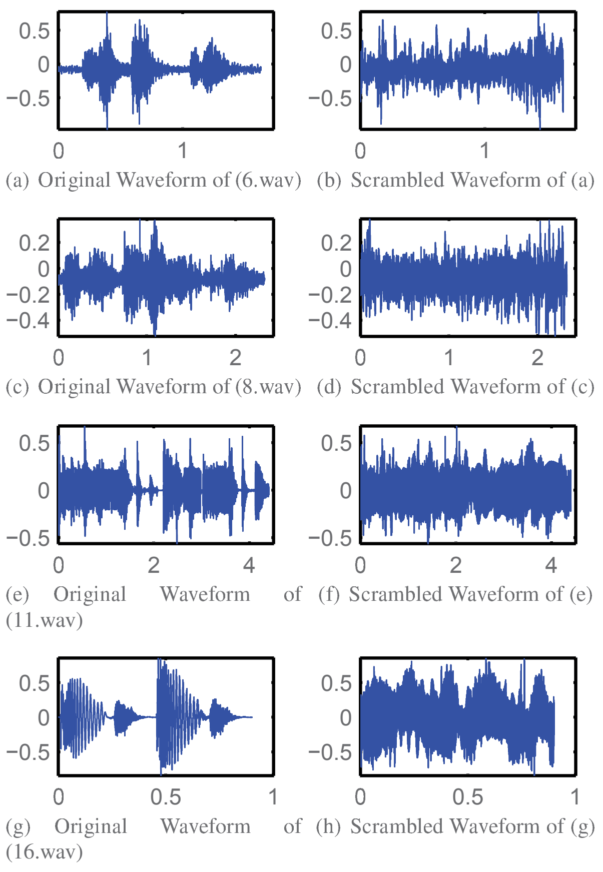


Fig. 2 Scrambled wave plots of different audio files with $NOG = 15$

5.2 Correlation analysis of neighborhood type

Although many different possible neighborhood types exist, we have only tested von Neumann and Moore neighborhood types.

Table 3 shows the scrambling degree obtained when the audio files are scrambled using Moore and von Neumann neighborhood types. The scrambling experiments use the same key for both neighborhood types, and the scrambling is not repeated ($N = 0$).

The Moore neighborhood provides significantly better scrambling results than von Neumann neighborhood, this is because the neighborhood effect applies to all cells in the grid and many of the CA properties are strongly dependent on the neighborhood [19]. Fig. 3 shows the result of scrambling 4.wav and 13.wav using different neighborhood types.

5.3 Correlation analysis of boundary condition

Table 4 shows the scrambling degree obtained when the audio files are scrambled using periodic and null boundaries with no repetition ($N = 0$) and $NOG = 15$. Generally the periodic boundary gives a slightly higher scrambling degree than the null boundary, but not always, because the boundary condition affects only the cells on the edges of the grid. The periodic boundary gives better randomness quality [10].

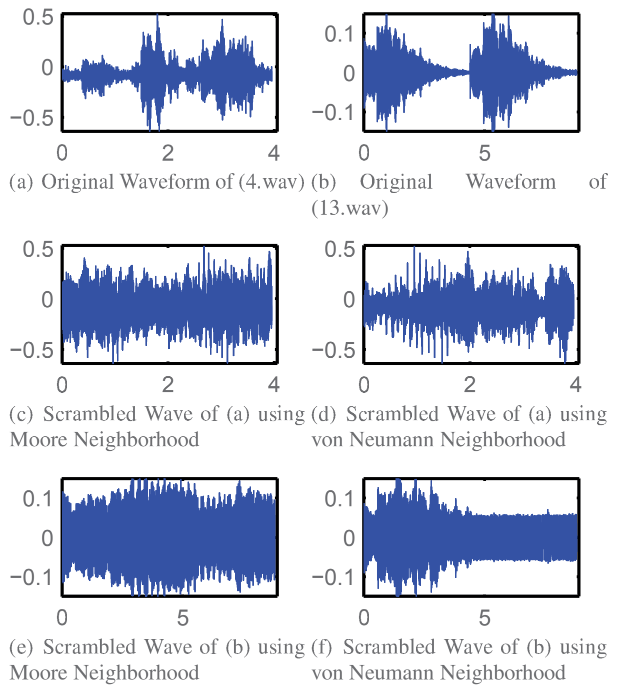
5.4 Correlation analysis of Lambda values

Table 5 shows different transition rules which we have chosen to test their effect on audio scrambling. The table also shows their rule numbers.

Table 6 shows the scrambling degrees obtained for different lambda values, in the experiments. No repetition was used ($N = 0$). The results show that the complex behavior of the Game of Life rule (GOL) which occurs around

Table 3 Scrambling degree when different neighborhood types are used with $NOG = 15$

	Neighborhood	
Audio file	**von Neumann**	**Moore**
1.wav	0.92747	0.89216
2.wav	0.92549	0.89339
3.wav	0.93893	0.90888
4.wav	0.91776	0.87439
5.wav	0.91871	0.87036
6.wav	0.90219	0.84349
7.wav	0.92028	0.87767
8.wav	0.90673	0.87753
9.wav	0.99895	0.99826
10.wav	0.99574	0.99373
11.wav	0.94070	0.91001
12.wav	0.95558	0.93194
13.wav	0.93027	0.88261
14.wav	0.92479	0.88907
15.wav	0.98728	0.97899
16.wav	0.93952	0.90297



(a) Original Waveform of (4.wav) (b) Original Waveform of (13.wav)

(c) Scrambled Wave of (a) using Moore Neighborhood (d) Scrambled Wave of (a) using von Neumann Neighborhood

(e) Scrambled Wave of (b) using Moore Neighborhood (f) Scrambled Wave of (b) using von Neumann Neighborhood

Fig. 3 Audio files scrambled using different neighborhood types

Table 4 Scrambling degree when different CA types are used

	Boundary	
Audio file	**Periodic**	**Null**
1.wav	0.92747	0.92587
2.wav	0.92549	0.92522
3.wav	0.93893	0.93758
4.wav	0.91776	0.91731
5.wav	0.91871	0.91690
6.wav	0.90219	0.90085
7.wav	0.92028	0.91932
8.wav	0.90673	0.90493
9.wav	0.99895	0.99893
10.wav	0.99574	0.99572
11.wav	0.94070	0.94023
12.wav	0.95558	0.95495
13.wav	0.93027	0.93007
14.wav	0.92479	0.92372
15.wav	0.98728	0.98737
16.wav	0.93952	0.93810

Table 5 The Lambda value and rule number of different transition functions

Lambda Value (λ)	**Transition Function**	**Rule No.**
0.27340	$f(0,3) = 1$, $f(1,2) = 1$, $f(1,3) = 1$, f equals zero otherwise	GOL
0.30078	$f(0,4) = 1$, $f(1,2) = 1$, $f(1,3) = 1$, f equals zero otherwise	416
0.32812	$f(0,4) = 1$, $f(1,2) = 1$, $f(1,4) = 1$, f equals zero otherwise	800
0.41601	$f(1,3) = 1$, $f(0,1) = 1$, $f(0,2) = 1$, $f(0,3) = 1$, $f(0,5) = 1$, $f(0,7) = 1$, $f(0,8) = 1$, f equals zero otherwise	83156
0.47070	$f(1,2) = 1$, $f(1,3) = 1$, $f(0,1) = 1$, $f(0,2) = 1$, $f(0,3) = 1$, $f(0,5) = 1$, $f(0,7) = 1$, $f(0,8) = 1$, f equals zero otherwise	83188

$\lambda = 0.2734$ gives the highest scrambling effect. This result could have been foreseen from Wolfram analysis of CAs, but it is nice to see it confirmed.

5.5 Robustness experiments

In an effort to measure the algorithm robustness, we applied the data loss attack where we eliminated $1/3$ of the data samples of the data set audio files. Each audio file was scrambled twice, once with no repetitions ($N = 0$) and once with five repetitions ($N = 5$), in order to show the relation between the scrambling degree and the robustness of the algorithm.

Fig. 4 shows the recovered waves of 1.wav and 12.wav after data loss. The scrambling degree for 1.wav is 0.92747 when $N = 0$ and 0.93951 when $N = 5$. The scrambling degree for 12.wav is 0.95558 when $N = 0$ and 0.96559 when $N = 5$.

From Fig. 4 it can be seen that the structure of the recovered wave is so similar to the original, especially when $N = 5$ (because with better scrambling the samples are distributed in a way that breaks the correlation between samples), so

Table 6 Scrambling Degree when using different transition rules and $NOG = 15$

Audio	**Rule No.**				
file	**GOL**	**416**	**800**	**83156**	**83188**
1.wav	0.92747	0.90355	0.90964	0.89675	0.89046
2.wav	0.92549	0.89740	0.90613	0.90563	0.89999
3.wav	0.93893	0.91107	0.91898	0.90940	0.90419
4.wav	0.91776	0.86127	0.88539	0.88648	0.88524
5.wav	0.91871	0.88793	0.89275	0.88076	0.87433
6.wav	0.90219	0.87829	0.88081	0.85902	0.84297
7.wav	0.92028	0.88667	0.89171	0.88442	0.88037
8.wav	0.90673	0.86287	0.88331	0.86251	0.86054
9.wav	0.99895	0.99835	0.99841	0.99852	0.99831
10.wav	0.99574	0.99276	0.99335	0.99435	0.99393
11.wav	0.94070	0.90653	0.91013	0.91485	0.91386
12.wav	0.95558	0.93144	0.93686	0.94175	0.93873
13.wav	0.93027	0.87874	0.88853	0.89457	0.89018
14.wav	0.92479	0.88041	0.88895	0.89209	0.89223
15.wav	0.98728	0.98169	0.98288	0.98439	0.98244
16.wav	0.93952	0.89533	0.90633	0.90861	0.91605

even if the data is lost the audio recovers most of its original structure.

5.6 Comparison with previous schemes

The proposed scheme was compared to three algorithms proposed in [14]: The cyclic displacement scrambling transformation (CDST), the complete binary tree's inorder traversal scrambling transformation (ITST), and the combination algorithm. The parameters used in the experiments are the same parameters used in [14] as follows: in the combination algorithm $k = 3$, $p = 5$, and in CDST $k = 3$.

The speech audio files where sampled at 16000 Hz, while the music files where sampled at 44100 Hz, 48000 Hz, respectively. As shown in section 5.1 and section 5.5, achieving a high scrambling degree is dependent on setting the number of generations and iterations, so we set the NOG to 15 and $N = 6$. Table 7 shows the results of the comparison.

Table 7 Comparing ASCA (proposed algorithm) with previous schemes

Audio file	**ASCA**	**ITST**	**CDST**	**Combination algorithm**
Speech1.wav	0.92445	0.91933	0.92367	0.92335
Speech2.wav	0.87971	0.86153	0.87858	0.87924
Music1.wav	0.93523	0.92637	0.93398	0.93478
Music2.wav	0.91490	0.90825	0.91475	0.91476

The waveform of the audio after scrambling using the ASCA is shown in Fig. 5. The scrambled waves do not show any of the original audio structure, so no information about the original file can be retrieved from them.

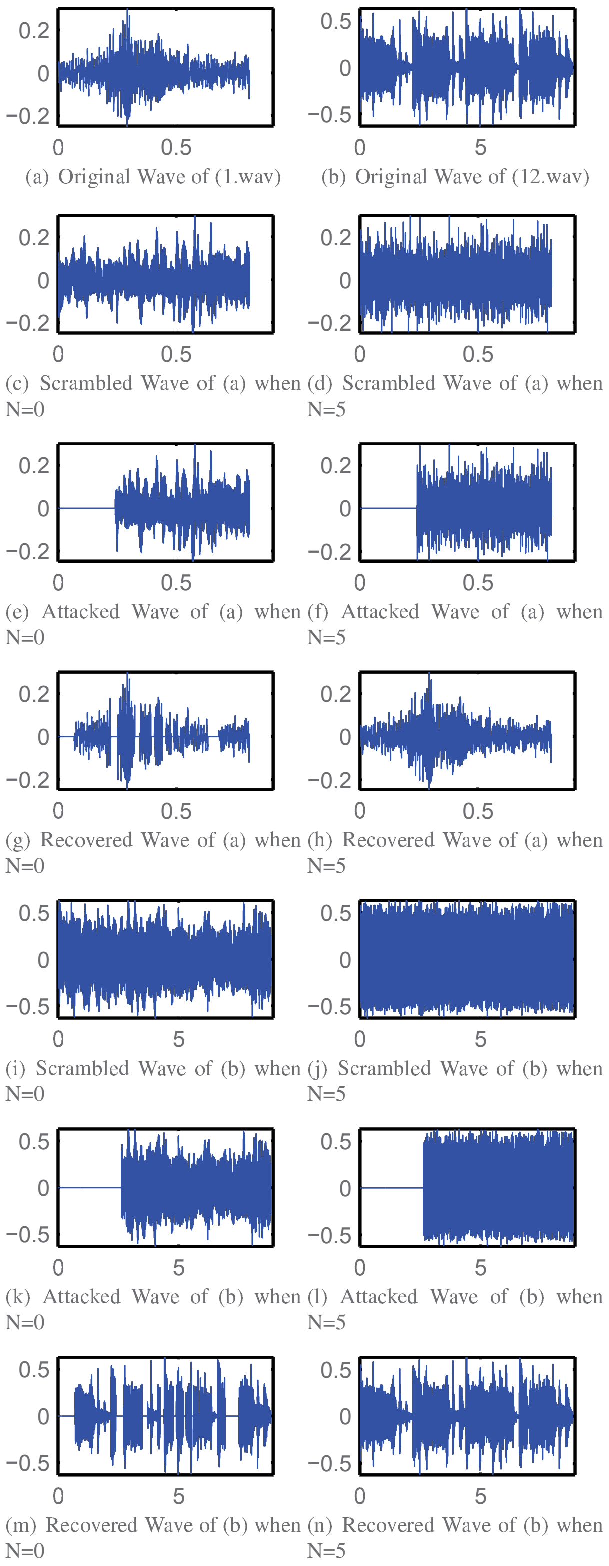
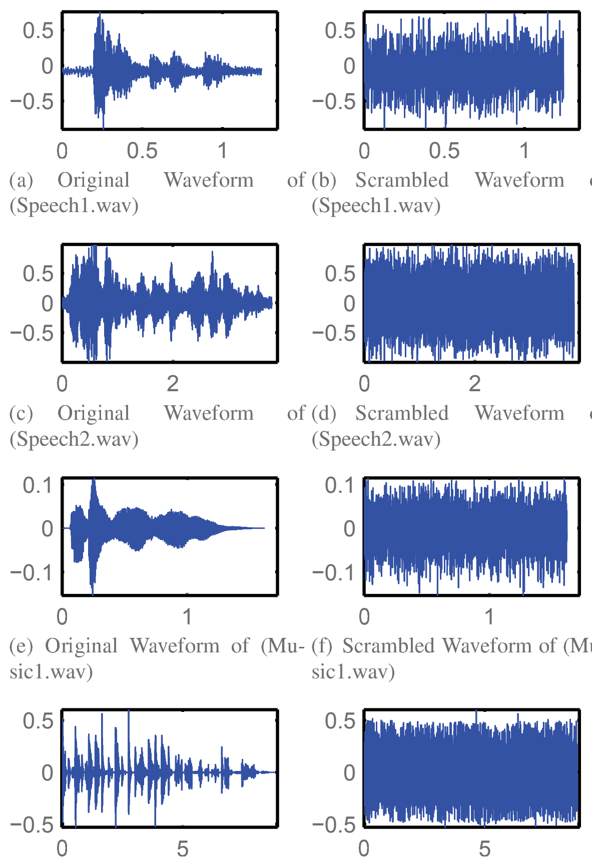

(a) Original Wave of (1.wav) (b) Original Wave of (12.wav)
(c) Scrambled Wave of (a) when N=0 (d) Scrambled Wave of (a) when N=5
(e) Attacked Wave of (a) when N=0 (f) Attacked Wave of (a) when N=5
(g) Recovered Wave of (a) when N=0 (h) Recovered Wave of (a) when N=5
(i) Scrambled Wave of (b) when N=0 (j) Scrambled Wave of (b) when N=5
(k) Attacked Wave of (b) when N=0 (l) Attacked Wave of (b) when N=5
(m) Recovered Wave of (b) when N=0 (n) Recovered Wave of (b) when N=5

Fig. 4 Recovered audio file after data loss attack with $NOG = 15$

6 Conclusions and Future work

A new scrambling technique for digital audio has been introduced. The proposed scheme takes advantage of 2D cellular



(a) Original Waveform of (Speech1.wav) (b) Scrambled Waveform of (Speech1.wav)

(c) Original Waveform of (Speech2.wav) (d) Scrambled Waveform of (Speech2.wav)

(e) Original Waveform of (Music1.wav) (f) Scrambled Waveform of (Music1.wav)

(g) Original Waveform of (Music2.wav) (h) Scrambled Waveform of (Music2.wav)

Fig. 5 Audio files scrambled using ASCA with $NOG = 15$

automata with complex behavior to achieve a high scrambling degree. The paper studies the effect of using von Neumann neighborhood versus Moore neighborhood and the periodic boundary versus the null boundary. Five transition rules with different Lambda values were tested. The process is suitable for speech and music clips of different sizes and no extra padding is needed. The descrambling process is straightforward when the right key is available.

Experimental results suggest that the proposed technique breaks the correlation of adjacent data samples effectively. The relation between the scrambling degree achieved and the robustness of the algorithm is also studied, the results show that the algorithm is robust to data loss attack and the robustness becomes better when the scrambling degree is higher.

Some of the most popular CA types were studied in terms of digital audio scrambling, but many more exists; future plans include the extensive study of other CA types and other possible combinations, and extending this scheme to scramble video files.

Studying the best approach to extend the algorithm to include multi-channel audio is also left for future work. This extension can be done by treating each channel separately, or by considering inter-channel dependencies.

Other future plans will include the use of this algorithm as a part of watermarking, information hiding, fingerprinting, and encryption applications.

Acknowledgements This work is partially supported by the Spanish Ministry of Science and Innovation under coordinated research projects TIN2011-28260-C03-00 and TIN2011-28260-C03-02 and by the Comunidad Autónoma de Madrid under research project e-madrid S2009/TIC-1650

References

1. Yan W, Fu W, Kankanhalli M S (2008) Progressive audio scrambling in compressed domain. IEEE Transactions on Multimedia 10(6):960–968
2. Huang H C, Chen Y H (2009) Genetic fingerprinting for copyright protection of multicast media. Soft Computing - A Fusion of Foundations, Methodologies and Applications 13(4):383–391
3. Martínez-Noriega R, Nakano M, Kurkoski B, Yamaguchi K (2011) High payload audio watermarking: Toward channel characterization of MP3 compression. Journal of Information Hiding and Multimedia Signal Processing 2(2):91–107
4. Chang F C, Huang H C, Hang H M (2007) Layered Access Control Schemes on Watermarked Scalable Media. Journal of VLSI Signal Processing Systems 49(3):443–455
5. Shang Z, Ren H, Zhang J (2008) A Block Location Scrambling Algorithm of Digital Image Based on Arnold Transformation. Proc. 9th International Conference for Young Computer Scientists, Hunan, China, pp 2942–2947
6. Jiping N, Yongchuan Z, Zhihua H, Zuqiao Y (2008) A digital image scrambling method based on AES and error correcting code. Proc. International Conference on Computer Science and Software Engineering, Wuhan, Hubei, China, pp 677–680
7. Zhu L, Li W, Liao L, Li H (2006) A Novel Algorithm for Scrambling Digital Image Based on Cat Chaotic Mapping. Proc. International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06), Pasadena, CA, USA, pp 601–604
8. Xiangdong L, Junxing Z, Jinhai Z, Xiqin H (2008) A New Chaotic Image Scrambling Algorithm Based on Dynamic Twice Interval-Division. Proc. International Conference on Computer Science and Software Engineering, Wuhan, Hubei, China, pp 818–821
9. Ye R, Li H (2008) A Novel Image Scrambling and Watermarking Scheme Based on Cellular Automata. Proc. International Symposium on Electronic Commerce and Security, Guangzhou City, China, pp 938–941
10. Abu Dalhoum A, Mahafzah B, Awwad A, Al-Dhamari I, Ortega A, Alfonseca M (2011) Digital Image Scrambling Method Based On Two Dimensional Cellular Automata: A Test of the Lambda Value. IEEE Multimedia. doi: 10.1109/MMUL.2011.54
11. Li H, Qin Z (2009) Audio Scrambling Algorithm Based on Variable Dimension Space. Proc. International Conference on Industrial and Information Systems, Haikou, China, pp 316–319
12. Li H, Qin Z, Shao L, Zhang S, Wang B (2009) Variable Dimension Space Audio Scrambling Algorithm Against MP3 Compression. In: Hua A, Chang S (ed) Algorithms and Architectures for Parallel Processing. Springer, Berlin, Heidelberg, pp 866–876
13. Li H, Qin Z, Shao L (2009) Audio Watermarking Pre-process Algorithm. Proc. IEEE International Conference on e-Business Engineering, Macau, China, pp 165–170

14. Chen G, Hu Q (2010) An audio scrambling method based on combination strategy. Proc. International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, China, pp 62–66
15. Sarkar P (2000) A Brief History of Cellular Automata. ACM Computing Surveys (CSUR) 32(1):80–107
16. Kier L, Witten T (2005) Cellular Automata Models of Complex Biochemical Systems. In: Bonchev D, Rouvray D (ed) Complexity in Chemistry, Biology, and Ecology. Springer, pp 237–301
17. Kier L, Seybold P, Cheng C (2005) Water as a System. In: Modeling Chemical Systems Using Cellular Automata. Springer, pp 39–55
18. Bonnet N, Matos M, Polette M, Zahm J, Nawrocki-Raby B, Birembaut P (2004) A density-based cellular automaton model for studying the clustering of noninvasive cells. IEEE Transactions on Biomedical Engineering 51(7):1274–1276
19. Nishio H (2006) How Does the Neighborhood Affect the Global Behavior of Cellular Automata. In: Yacoubi S, Chopard B, Bandini S (ed) Cellular Automata. Springer, Lecture Notes in Computer Science 4173:122–130
20. Shin S, Yoo K (2009) Analysis of 2-State, 3-Neighborhood Cellular Automata Rules for Cryptographic Pseudorandom Number Generation. Proc. International Conference on Computational Science and Engineering (CSE'09), Vancouver, BC, Canada, pp 399–404
21. Wolfram S (2002) A New Kind of Science. Wolfram Media, USA
22. Langton C (1990) Computation at the edge of chaos: Phase transitions and emergent computation. Physica D: Nonlinear Phenomena 42(1-3):12–37
23. Aleksić Z (2000) Artificial life: growing complex systems. In: Bossomaier T, Green D (ed) Complex Systems. Cambridge University Press, pp 91–126
24. Gardner M (1970) Mathematical Games – The fantastic combinations of John Conway's new solitaire game "life". Scientific American 223:120–123