Multimed Tools Appl (2014) 71:1803–1822
DOI 10.1007/s11042-012-1306-7

Audio scrambling technique based on cellular automata

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

Published online: 29 December 2012

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 breaking the correlation between audio samples effectively. 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 scrambling degree is measured and the relation between the robustness and the scrambling degree obtained is studied. Experimental results show that the proposed technique is robust to data loss attack where 1/3 of the data is lost and that the algorithm can be applied to music and speech files of different sizes.

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 [22].

A. Madain · A. L. Abu Dalhoum · H. Hiary (✉)
University of Jordan, Amman 11942, Jordan
e-mail: hazemh@ju.edu.jo

A. Ortega · M. 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 [8], the development of robust data hiding system helps more technologies find new and promising applications [15], 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 this paper we propose a new Audio Scrambling algorithm based on Cellular Automata (ASCA). Cellular automata (CA) is a discrete model of computation that could be used to solve any computable task. This work seeks to find out how to best benefit from CA characteristics in audio scrambling, where different CA types were tested to see which one will lead to a higher scrambling degree. Also, the experiments were applied to 30 audio files in WAV format which contains music and speech files of different sizes and the robustness of cellular automata techniques was tested against data loss attack where 1/3 of 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; Section 4 describes the scrambling algorithm proposed and the scrambling degree measurement; Section 5 gives the analysis and discussion of the experimental results; Section 6 compares the work done with previous schemes, and finally, in Section 7, we conclude the work done and discuss possible directions for future work.

2 Related works

A lot of research has been done in the field of scrambling. For digital images, many scrambling methods are available, such as those based on Arnold transformation [18], Advanced Encryption Standard (AES) and error-correcting code [9], cat chaotic mapping [24], and dynamic twice interval-division [21], among others.

Cellular automata have also been used for digital image scrambling: Ye and Li [23] described the use of CA with chaotic behavior, while Abu Dalhoum et al. [1] 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 [12] and used in [13, 14], 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 audio.

In [5], two scrambling algorithms are given, namely, the cyclic displacement scrambling transformation (CDST) and the complete binary tree's inorder traversal scrambling transformation (ITST), then the two algorithms are combined. The combined algorithm takes two positive integers as keys; the first is used in CDST, while the other determines the order of execution of the two algorithms.

Work done in [6] focuses on the compressed domain, where a progressive audio scrambling and descrambling scheme is applied to MP3 audio, the algorithm is progressive in the sense that it does multiple rounds of scrambling based on keys to produce a set of audio outputs of differing qualities. In order to reconstruct the original, all the keys must be available; if some of the keys are missing then the reconstructed audio has lower quality than the original.

In this paper we propose a new audio scrambling technique that depends on CA in generating the scrambling key, and we study different types of CA in terms of audio scrambling. The proposed algorithm is applied to audio in WAV format, and does not require any additional padding; moreover, experimental results show that the algorithm is applicable to speech and music audio files with different sizes, and can break the correlations between audio samples effectively, in addition to being robust to data loss attacks. Section 6 discusses scrambling algorithms applied to WAV audio files further.

3 Cellular automata

Cellular automata (CA) are simple and highly parallel models of computation which can exhibit complex behavior [17]. They are used in many applications covering different fields, for example, CA is used to model complex biochemical systems [10], and to build vehicular traffic models [3].

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 [16]. 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 set to a zero state cell, and a periodic boundary if the extreme cells at opposite sides are adjacent to one another [19].

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 [20]. In *class*1, patterns evolve into a stable, homogeneous state; in *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 ways.

A way to describe the relation between the transition rules and the behavior of the CA is the λ parameter, defined by Langton [11]. 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 [2].

3.3 Calculating the transition rule number

In [23] the rule number (C) is calculated as follows:

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

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 [7], 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

The scrambling algorithm 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 Q, 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 Q, if it is within range, then $A\left[index\right] = X\left[ptr\right]$. 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 needed.

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. Figure 1a shows an audio wave file, Fig. 1b shows a list of indices retrieved from the key generation process described in Section 4.2, Fig. 1c 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.

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 Q. 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 a status matrix B of size $M \times M$ to zero.
(3) Run the Game of life rules (using Moore neighborhood and periodic boundary) for (NOG) generations starting from the initial state configuration. Subsequent state configurations are $K_1, K_2, \cdots, K_{NOG}$.

Fig. 1 Audio scrambling process, (**a**) original audio, (**b**) list of new indices, (**c**) scrambled audio

-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)

(4) For $k = 1, 2, \cdots, NOG$, if $K_k\left[i, j\right] = 1$ and $B\left[i, j\right] = 0$, add (i, j) to the list of new indices Q, and set $B\left[i, j\right]$ to 1.
(5) Return Q.

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

The average scrambling degree measurement is used to evaluate image scrambling schemes as in [1, 23]; we have used the same measurement to analyze the effect of different cellular automata types on audio scrambling, but before applying it to audio files, 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 L 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:

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

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 (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.

The data set used to study the behavior of CAs contains 30 audio files with different waves. The resolution of all the audio files used is 16 bit and the audios are in WAV format. Those numbered from 1 to 15 are speech audio files; while those numbered from 16 to 30 are music audio files. All speech audio files have one channel, with a sampling rate of 16000 Hertz and a Bit rate of 256 kbps. The speech audio files duration ranges from 0.810562 to 23.1853 s. Table 1 shows the details of the music audio files.

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

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 higher the scrambling degree obtained.

Based on the proposed algorithm, the scrambling degree increases as the number of generations increase, 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 the correlation between the samples higher

Table 1 Details of audio files used to study CA behavior

Audio file	Duration (seconds)	Sample rate (Hz)	Bit rate (kbps)	Channels
16.wav	1.9805	44100	705	1
17.wav	1.83991	44100	705	1
18.wav	4.40367	48000	1536	2
19.wav	8.80733	48000	1536	2
20.wav	8.80733	48000	1536	2
21.wav	1.84154	44100	705	1
22.wav	2.18181	44100	1411	2
23.wav	0.901859	44100	1411	2
24.wav	3.66782	44100	1411	2
25.wav	2.07238	44100	705	1
26.wav	1.87728	44100	705	1
27.wav	2.00615	44100	1411	2
28.wav	3.02077	44100	705	1
29.wav	5.04311	44100	1411	2
30.wav	1.67435	44100	1411	2

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	0.85515	0.91662	0.92466	0.92713	0.92919	0.93069
10.wav	0.83170	0.89973	0.90841	0.91162	0.91227	0.91233
11.wav	0.81258	0.88852	0.90110	0.90587	0.90846	0.90975
12.wav	0.88058	0.92639	0.93450	0.93736	0.93813	0.93878
13.wav	0.84069	0.89677	0.90909	0.91043	0.91338	0.91340
14.wav	0.84991	0.91386	0.91930	0.92142	0.92304	0.92388
15.wav	0.85201	0.92929	0.93458	0.93804	0.94066	0.94080
16.wav	0.99794	0.99883	0.99885	0.99895	0.99898	0.99899
17.wav	0.99091	0.99468	0.99535	0.99574	0.99580	0.99586
18.wav	0.87958	0.92720	0.93791	0.94070	0.94394	0.94440
19.wav	0.90406	0.94644	0.95319	0.95558	0.95684	0.95765
20.wav	0.86702	0.90716	0.92831	0.93027	0.93052	0.93075
21.wav	0.87101	0.91189	0.91424	0.92479	0.92690	0.92691
22.wav	0.97422	0.98468	0.98687	0.98728	0.98743	0.98761
23.wav	0.88890	0.92236	0.93936	0.93952	0.93959	0.93966
24.wav	0.87336	0.93423	0.93662	0.94107	0.94454	0.94675
25.wav	0.92484	0.94620	0.95227	0.95433	0.95543	0.95604
26.wav	0.95391	0.97193	0.97488	0.97594	0.97689	0.97765
27.wav	0.86357	0.92056	0.93116	0.93448	0.93498	0.93716
28.wav	0.89125	0.94178	0.94663	0.95070	0.95209	0.95306
29.wav	0.92964	0.95775	0.96139	0.96608	0.96721	0.96872
30.wav	0.87511	0.92300	0.92803	0.93216	0.93580	0.93738

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 our experiments we will use fifteen generations. Figure 2 shows the test audio files scrambled for 15 generations with $N = 0$. Although the scrambled waves shown in the figure have very different shape and form from their original, the level of scrambling is enhanced significantly when N is greater than zero as given in Section 5.5.

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

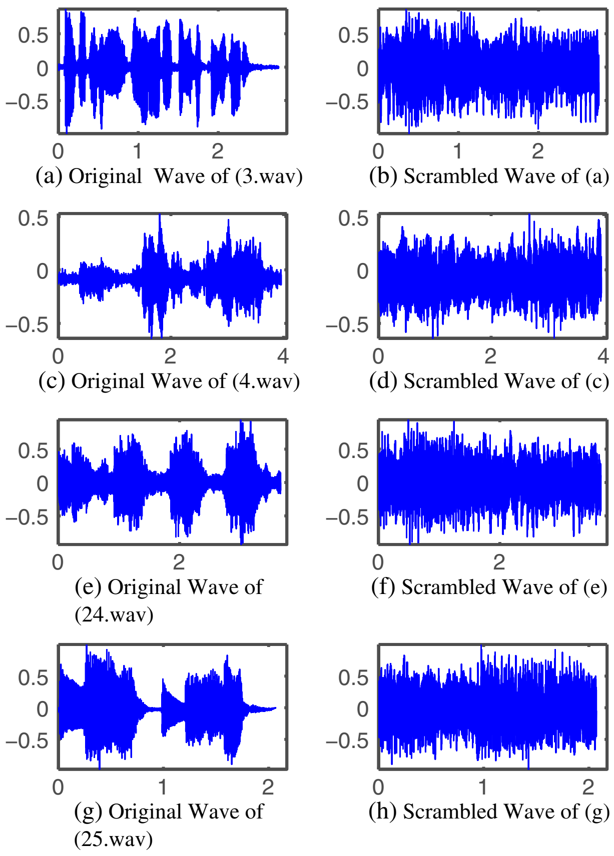


Table 3 Scrambling degree when different neighborhood types are used with $NOG = 15$

Audio file	Neighborhood		Audio file	Neighborhood	
	Moore	von Neumann		Moore	von Neumann
1.wav	0.92747	0.89216	16.wav	0.99895	0.99826
2.wav	0.92549	0.89339	17.wav	0.99574	0.99373
3.wav	0.93893	0.90888	18.wav	0.94070	0.91001
4.wav	0.91776	0.87439	19.wav	0.95558	0.93194
5.wav	0.91871	0.87036	20.wav	0.93027	0.88261
6.wav	0.90219	0.84349	21.wav	0.92479	0.88907
7.wav	0.92028	0.87767	22.wav	0.98728	0.97899
8.wav	0.90673	0.87753	23.wav	0.93952	0.90297
9.wav	0.92713	0.89604	24.wav	0.94107	0.90352
10.wav	0.91162	0.85531	25.wav	0.95433	0.93171
11.wav	0.90587	0.85552	26.wav	0.97594	0.96288
12.wav	0.93736	0.90851	27.wav	0.93448	0.90197
13.wav	0.91043	0.86898	28.wav	0.95070	0.92723
14.wav	0.92142	0.89487	29.wav	0.96608	0.95085
15.wav	0.93804	0.90971	30.wav	0.93216	0.89838

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 [16]. Figure 3 shows the result of scrambling 15.wav and 28.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 $NOG = 15$. The periodic boundary gives better randomness quality [1], but based on the results shown in Table 4, the difference between the two boundary types is not as significant as the difference between the tested neighborhood types. Also it can be seen from this table that on average the periodic boundary gives a slightly higher scrambling degree than the null boundary.

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.

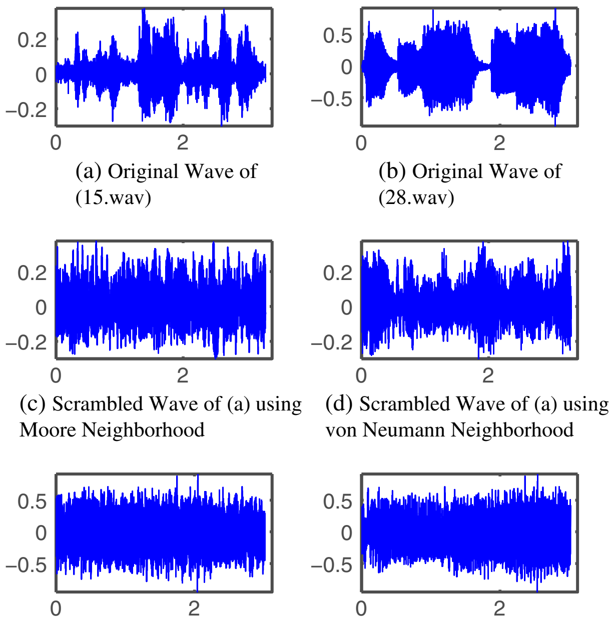


Fig. 3 Audio files scrambled using different neighborhood types

(a) Original Wave of (15.wav)
(b) Original Wave of (28.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

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

Table 4 Scrambling degree when different CA types are used

Audio	Boundary		Audio	Boundary	
file	Periodic	Null	file	Periodic	Null
1.wav	0.92747	0.92587	16.wav	0.99895	0.99893
2.wav	0.92549	0.92522	17.wav	0.99574	0.99572
3.wav	0.93893	0.93758	18.wav	0.94070	0.94023
4.wav	0.91776	0.91731	19.wav	0.95558	0.95495
5.wav	0.91871	0.91690	20.wav	0.93027	0.93007
6.wav	0.90219	0.90085	21.wav	0.92479	0.92372
7.wav	0.92028	0.91932	22.wav	0.98728	0.98737
8.wav	0.90673	0.90493	23.wav	0.93952	0.93810
9.wav	0.92713	0.92656	24.wav	0.94107	0.94086
10.wav	0.91162	0.90842	25.wav	0.95433	0.95344
11.wav	0.90587	0.90560	26.wav	0.97594	0.97549
12.wav	0.93736	0.93727	27.wav	0.93448	0.93361
13.wav	0.91043	0.90984	28.wav	0.95070	0.94977
14.wav	0.92142	0.92123	29.wav	0.96608	0.96602
15.wav	0.93804	0.93772	30.wav	0.93216	0.93051

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

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

Audio file	Rule no.				
	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	0.92713	0.89360	0.90054	0.90368	0.90060
10.wav	0.91162	0.88852	0.89222	0.87261	0.86123
11.wav	0.90587	0.86327	0.87035	0.87866	0.87469
12.wav	0.93736	0.90891	0.91892	0.91685	0.91084
13.wav	0.91043	0.88061	0.88571	0.89263	0.88540
14.wav	0.92142	0.89169	0.90214	0.90917	0.90449
15.wav	0.93804	0.90007	0.91365	0.91378	0.91430
16.wav	0.99895	0.99835	0.99841	0.99852	0.99831
17.wav	0.99574	0.99276	0.99335	0.99435	0.99393
18.wav	0.94070	0.90653	0.91013	0.91485	0.91386
19.wav	0.95558	0.93144	0.93686	0.94175	0.93873
20.wav	0.93027	0.87874	0.88853	0.89457	0.89018
21.wav	0.92479	0.88041	0.88895	0.89209	0.89223
22.wav	0.98728	0.98169	0.98288	0.98439	0.98244
23.wav	0.93952	0.89533	0.90633	0.90861	0.91605
24.wav	0.94107	0.91349	0.91628	0.92185	0.92103
25.wav	0.95433	0.93845	0.93981	0.93195	0.92978
26.wav	0.97594	0.96321	0.96602	0.97001	0.96903
27.wav	0.93448	0.90217	0.91217	0.91665	0.91470
28.wav	0.95070	0.92798	0.93666	0.93856	0.93705
29.wav	0.96608	0.94266	0.94532	0.95795	0.95549
30.wav	0.93216	0.90016	0.90845	0.91578	0.91248

Table 7 Relation between repetition (N) and scrambling degree

Audio file	$N = 0$	$N = 1$	Audio file	$N = 0$	$N = 1$
1.wav	0.92747	0.93757	16.wav	0.99895	0.99912
2.wav	0.92549	0.94025	17.wav	0.99574	0.99644
3.wav	0.93893	0.94795	18.wav	0.94070	0.95063
4.wav	0.91776	0.93023	19.wav	0.95558	0.96351
5.wav	0.91871	0.93111	20.wav	0.93027	0.94006
6.wav	0.90219	0.91506	21.wav	0.92479	0.93485
7.wav	0.92028	0.93164	22.wav	0.98728	0.98967
8.wav	0.90673	0.91975	23.wav	0.93952	0.94921
9.wav	0.92713	0.94101	24.wav	0.94107	0.95251
10.wav	0.91162	0.92461	25.wav	0.95433	0.96300
11.wav	0.90587	0.92293	26.wav	0.97594	0.98102
12.wav	0.93736	0.94821	27.wav	0.93448	0.94699
13.wav	0.91043	0.92989	28.wav	0.95070	0.96097
14.wav	0.92142	0.93897	29.wav	0.96608	0.97258
15.wav	0.93804	0.94826	30.wav	0.93216	0.94623

5.5 Correlation between confusion and repetition

In all the previous experiments, the audio files were scrambled with no repetition ($N = 0$), in Table 7, it can be seen that the repetition for only one time increases the scrambling degree and hence the confusion. Figure 4 shows the wave of 13.wav and 20.wav with and without repetition. Although in both the scrambled waves do not show any of the original audio structure, scrambling with repetition is better and have a higher scrambling degree.

Fig. 4 Audio files scrambled before and after repetition where $NOG = 15$

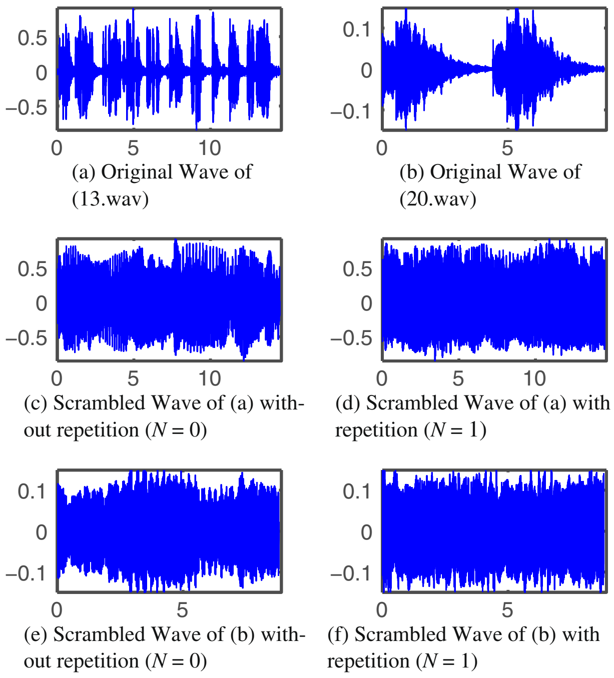
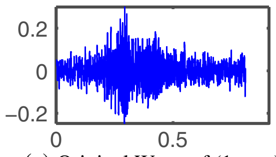
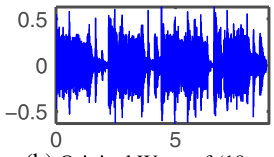


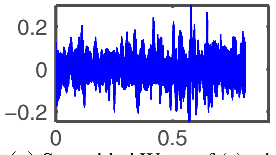
Fig. 5 Recovered audio file after data loss attack with $NOG = 15$



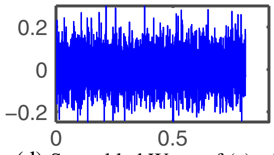
(a) Original Wave of (1.wav)



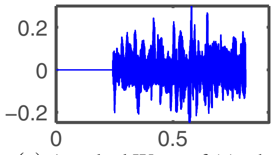
(b) Original Wave of (19.wav)



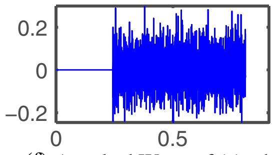
(c) Scrambled Wave of (a) when $N = 0$



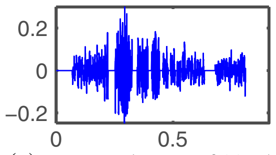
(d) Scrambled Wave of (a) when $N = 4$



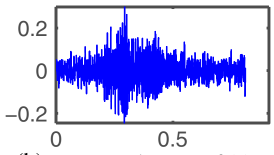
(e) Attacked Wave of (a) when $N = 0$



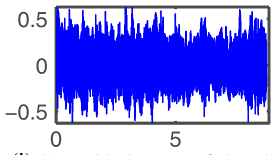
(f) Attacked Wave of (a) when $N = 4$



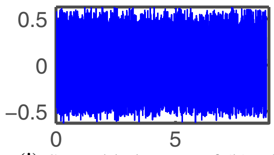
(g) Recovered Wave of (a) when $N = 0$



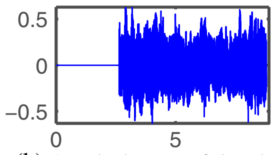
(h) Recovered Wave of (a) when $N = 4$



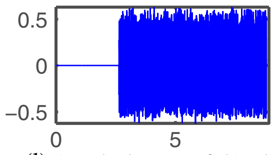
(i) Scrambled Wave of (b) when $N = 0$



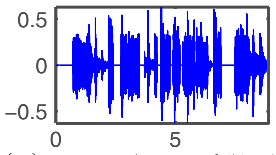
(j) Scrambled Wave of (b) when $N = 4$



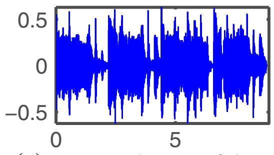
(k) Attacked Wave of (b) when $N = 0$



(l) Attacked Wave of (b) when $N = 4$



(m) Recovered Wave of (b) when $N = 0$



(n) Recovered Wave of (b) when $N = 4$

5.6 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, the first time with no repetitions ($N = 0$) and the second time the scrambling was repeated four times ($N = 4$), in order to show the relation between the scrambling degree and the robustness of the algorithm. Figure 5 shows the recovered waves of 1.wav and 19.wav after data loss. The scrambling degree for 1.wav is 0.92747 when $N = 0$ and 0.93951 when $N = 4$. The scrambling degree for 19.wav is 0.95558 when $N = 0$ and 0.96559 when $N = 4$.

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

When listening to the recovered audio we can absolutely understand it, although with the lower scrambling degree it can be noticed that there are isolated clicks in the audio, and with the higher scrambling degree some noise is introduced.

6 Comparison with previous schemes

Audio scrambling algorithms can be evaluated by many aspects, including high security and difficulty to descramble, complexity and applicability to real time applications, robustness, key size, in addition to the scrambled audio length and the need for padding, restrictions on the audio length, audio type, dimension, etc. the choice of which algorithm to use is usually dependent on the application and resources available. In this section we will discuss those different aspects and compare between some of the audio scrambling algorithms proposed.

Some of the scrambling algorithms map the original audio samples $X_1, X_2, \ldots, X_{length(x)}$ to the scrambled audio samples $X'_1, X'_2, \ldots, X'_{length(x)}$ where the length of X is preserved as in our proposed algorithm (ASCA), other algorithms require padding so the original audio samples are mapped to $X'_1, X'_2, \ldots, X'_{length(x)+y}$ where y is the length of padded audio samples. Also in some algorithms suitable padding maybe occasionally required [22].

Cyclic displacement scrambling transformation (CDST) and the complete binary tree's in order traversal scrambling transformation (ITST), and their combined algorithm proposed in [5] does not require any additional padding, while the audio scrambling algorithm based on variable dimension space (ASVDS) proposed in [12] requires padding.

ASVDS was proposed to address the problems of one-dimensional linear mapping, it increases the audio dimension to a variable dimension space, although changing the dimension gives better scrambling, it is notable that increasing the audio dimension higher than 2D does not necessarily increases the algorithm efficiency. CDST, ITST, and the combination between them scramble the audio in one dimension, while our proposed algorithm (ASCA) increases the dimension to 2D only.

Because audio scrambling applications are mostly in the security domain, the key strength and length are important, and the requirement of both aspects is dependent on the application, for example, some of the scrambling algorithms does not need any

Table 8 Comparing ASCA with previous schemes

	ASCA	ASVDS	CDST	ITST	Combination
Scrambled audio duration	No change	Longer than the original	No change	No change	No change
Key length dependency	Depends on the original audio length	Depends on the dimension	Independent integer number	No key required	Independent two integer numbers
Dimension	2D	Variable dimension	1D	1D	1D
Variables needed to descramble	Number of generations, repetitions, key	Original audio length, dimension used, transformation matrix	One integer number	No key required	Two integer numbers

key and does not achieve efficient scrambling as the others as in the case of ITST, but it might be very beneficial in case it is used as part of a process where at some step a key is added.

CDST requires only one integer to descramble, which is possible to decrypt if available to the public [5]. The combination algorithm uses two integer numbers to descramble, and finally ASVDS uses a transformation matrix of size $n \times n$ where n is the dimension. In ASVDS if the algorithm increases the audio to two dimensions, the key will be a 2×2 matrix and the determinant of that matrix is a relative prime of the square root of the scrambled audio length. Table 8 summarizes the comparison between the audio scrambling algorithms.

Other than the aspects we discussed above, there are benefits from relying on CA, for example, it is not a specific approach to scrambling but a general purpose discrete model of computation that could be used to solve any computable task which can be of a special benefit in systems that rely on this model. Moreover, CA is known for being a parallel model, which increases the performance of applications relying on it.

7 Conclusions and future work

A new scrambling technique for digital audio has been introduced. The proposed scheme takes advantage of 2D cellular 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. 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. In: IEEE multimedia. doi:10.1109/MMUL.2011.54
2. Aleksić Z (2000) Artificial life: growing complex systems. In: Bossomaier T, Green D (eds) Complex systems. Cambridge University Press, pp 91–126
3. Aponte A, Moreno J (2006) Cellular automata and its application to the modeling of vehicular traffic in the city of Caracas. In: El Yacoubi S, Chopard B, Bandini S (eds) Cellular automata. Springer, Lect Notes Comput Sci 4173:502–511
4. Chang FC, Huang HC, Hang HM (2007) Layered access control schemes on watermarked scalable media. J VLSI Signal Process Syst 49(3):443–455
5. Chen G, Hu Q (2010) An audio scrambling method based on combination strategy. In: Proc. international conference on computer science and information technology (ICCSIT), Chengdu, China, pp 62–66
6. Fu W, Yan W, Kankanhalli MS (2005) Progressive scrambling for MP3 audio. In: Proc. IEEE international symposium on circuits and systems (ISCAS), vol 6. Kobe, Japan, pp 5525–5528
7. Gardner M (1970) Mathematical games—the fantastic combinations of John Conway's new solitaire game "life". Sci Am 223:120–123
8. Huang HC, Chen YH (2009) Genetic fingerprinting for copyright protection of multicast media. Soft Comput 13(4):383–391
9. Jiping N, Yongchuan Z, Zhihua H, Zuqiao Y (2008) A digital image scrambling method based on AES and error correcting code. In: Proc. international conference on computer xcience and software engineering, Wuhan, Hubei, China, pp 677–680
10. Kier L, Witten T (2005) Cellular automata models of complex biochemical systems. In: Bonchev D, Rouvray D (eds) Complexity in chemistry, biology, and ecology. Springer, pp 237–301
11. Langton C (1990) Computation at the edge of chaos: phase transitions and emergent computation. Physica D 42(1–3):12–37
12. Li H, Qin Z (2009) Audio scrambling algorithm based on variable dimension Space. In: Proc. international conference on industrial and information systems, Haikou, China, pp 316–319
13. Li H, Qin Z, Shao L (2009) Audio watermarking pre-process algorithm. In: Proc. IEEE international conference on e-business engineering, Macau, China, pp 165–170
14. 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 (eds) Algorithms and architectures for parallel processing. Springer, Lect Notes Comput Sci 5574:866–876
15. 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 (JIHMSP) 2(2):91–107
16. Nishio H (2006) How does the neighborhood affect the global behavior of cellular automata. In: El Yacoubi S, Chopard B, Bandini S (eds) Cellular automata. Springer, Lect Notes Comput Sci 4173:122–130
17. Sarkar P (2000) A Brief History of Cellular Automata. ACM Comput Surv (CSUR) 32(1):80–107
18. Shang Z, Ren H, Zhang J (2008) A block location scrambling algorithm of digital image based on Arnold transformation. In: Proc. 9th international conference for young computer scientists, Hunan, China, pp 2942–2947

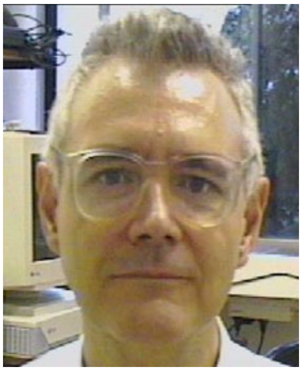
19. Shin S, Yoo K (2009) Analysis of 2-state, 3-neighborhood cellular automata rules for cryptographic pseudorandom number generation. In: Proc. international conference on computational science and engineering (CSE'09), Vancouver, BC, Canada, pp 399–404
20. Wolfram S (2002) A new kind of science. Wolfram Media, USA
21. Xiangdong L, Junxing Z, Jinhai Z, Xiqin H (2008) A new chaotic image scrambling algorithm based on dynamic twice interval-division. In: Proc. international conference on computer science and software engineering, Wuhan, Hubei, China, pp 818–821
22. Yan W, Fu W, Kankanhalli MS (2008) Progressive audio scrambling in compressed domain. IEEE Trans Multimedia 10(6):960–968
23. Ye R, Li H (2008) A novel image scrambling and watermarking scheme based on cellular automata. In: Proc. international symposium on electronic commerce and security, Guangzhou City, China, pp 938–941
24. Zhu L, Li W, Liao L, Li H (2006) A novel algorithm for scrambling digital image based on cat chaotic mapping. In: Proc. international conference on intelligent information hiding and multimedia signal processing (IIH-MSP'06), Pasadena, CA, USA, pp 601–604

Alia Madain received her B.Sc. and M.Sc. degrees in Computer Science from the University of Jordan in 2009 and 2011 respectively. Her research interests are multimedia processing and security.

Abdel Latif Abu Dalhoum received his PhD degree in computer science from the University Autonoma De Madrid, Spain in 2004. He is currently an Associate Professor of evolutionary algorithms and complex systems at the University of Jordan. He has published about 25 papers in evolutionary algorithms, cellular automata, Fractals and DNA computing. He is a member of GHIA research group at the University Autonoma De Madrid.

Hazem Hiary received his PhD in Computer Science / Image Processing from the University of Leeds (Leeds, UK) in 2008, B.Sc. and M.Sc. degrees in Computer Science from the University of Jordan (Jordan) in 2001 and 2003 respectively. He is currently an Assistant Professor in the Computer Science Department at the University of Jordan. His research interests include Image Processing, Pattern Recognition, Paper Watermarks, Document Analysis and Recognition, Audio Processing and Data Hiding.

Alfonso Ortega received the Doctorate degree in computer science from the Universidad Autónoma de Madrid, España. He is currently a Professor at the Universidad Autónoma. He formerly lectured at the Universidad Pontificia de Salamanca and worked at LAB2000 (an IBM subsidiary) as a Software Developer. He has published about 30 technical papers on computer languages, complex systems, graphics, and theoretical computer science, and has collaborated in the development of several software products.



Manuel Alfonseca is a doctor in Electronics Engineering (1972) and Computer Scientist (1976), both degrees obtained at the Universidad Politecnica of Madrid. He teaches and does research at the Department of Computer Science of the Universidad Autónoma of Madrid, where he was director of the Escuela Politécnica Superior (2001–2004). Previously, he was Senior Technical Staff Member at the IBM Madrid Scientific Center, where he worked from 1972 to 1994. He has published over 200 papers and books on computer languages, simulation, complex systems, graphics, artificial intelligence, object-orientation and theoretical computer science, as well as popular science and juvenile literature, with different awards in all of these fields.