electronics

MDPI

Article

A Novel Audio Encryption Algorithm with Permutation-Substitution Architecture

Krasimir Kordov

Department of Computer Informatics, Faculty of Mathematics and Computer Science, Konstantin Preslavsky University of Shumen, 9700 Shumen, Bulgaria; krasimir.kordov@shu.bg

Received: 7 April 2019; Accepted: 6 May 2019; Published: 11 May 2019

Abstract: In this paper, a new cryptographic method is proposed, designed for audio files' security. The encryption algorithm is based on classic symmetric models using pseudo-random number generator composed with chaotic circle map and modified rotation equations. The scheme of a new pseudo-random generator is presented and used as basis for chaotic bit-level permutations and substitutions applied to audio files structure for successful encryption. The audio encryption and decryption algorithms are described and explained. Proving the high level of security we provide extensive cryptographic analysis including key sensitivity analysis, key-space analysis, waveform and spectrogram analysis, correlation analysis, number of sample change rate analysis, level of noise analysis and speed performance test.

Keywords: audio encryption; permutation; substitution; pseudo-random number generator; circle map; rotation equation

1. Introduction

The cryptography in general is an art of secret transferring information from sender to receiver or a group of receivers. The modern technologies changed the way information is being sent simply because the information is now digital in a form of bits transferred in computer networks around the world. This factor requires standard cryptography algorithms used before digital ages to be applied and/or modernized to work with digital information. In this paper, we present a new method for encryption designed for audio files security in order to safely store and transfer this specific type of files. The other important aspect of cryptology is the cryptographic analysis with the main purpose to reveal the encrypted messages. The cryptographic analysis often uses different kind of empirical experiments that can be used to establish if any proposed algorithm has the necessary level of security or to prove that the algorithm is not secure enough. In this paper, we provide extensive cryptographic analysis for confirming the level of security of the proposed audio encryption scheme.

Previous research in this area has shown that the use of chaotic maps for construction audio encryption algorithms leads to high levels of security. In [1] Liu, Kadir and Li proposed encryption scheme using confusion and diffusion based on multi-scroll chaotic system. Hato and Shihab performed evaluation of Lorenz and Rossler chaotic system for speech signal encryption in [2]. More valuable research is presented in [3], where Sathiyamurthi and Ramakrishnan used Bernoulli's chaotic map for constructing encryption algorithm. Tamimi and Abdalla provide one more similar approach for audio shuffle-encryption algorithm in [4].

Important research [5] demonstrates that using a single chaotic map is not always a guarantee for highest levels of cryptographic security. The research overviews how encryption algorithms, using Arnold's Cat Map, Baker's Map or Two-Dimensional Logistic Chaotic Map may be vulnerable. This is one of the reasons we are using combination of two chaotic maps in order to extend the key-space (all the possible values for secret keys) for additional cryptographic security.

Considering the previous experience in this area we constructed a novel audio encryption algorithm with permutation-substitution architecture and compared the results with other algorithms.

2. Pseudo-Random Generator Based on Circle Map and Modified Rotation Equations

The pseudo-random generators (PRG) are software designed tools designed to provide endless sequence of random bits. PRGs are often used as main resource for symmetric cryptographic algorithms by using the random bits for encryption and decryption of digital files. Chaotic maps are widely preferred for constructing pseudo-random generators because of their chaotic behavior [6–8].

2.1. Circle Map

The Circle map is one dimensional dynamical system often used in cryptography because of its chaotic behavior. Examples of PRGs based on circle map are proposed in [9,10]. The iterations of the standard circle map are calculated by:

$$\theta_{n+1} = (\theta_n + \Omega - \frac{K}{2\pi}\sin(2\pi\theta_n)) \mod 1, \tag{1}$$

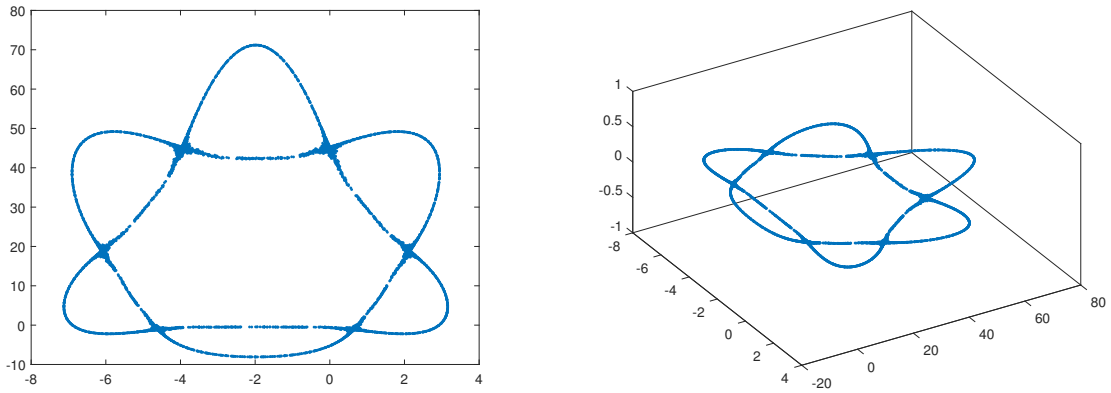
where Ω is a fixed constant playing the role of polar angle of the sinusoidal oscillator, and K is the coupling strength. The initial values we used for our scheme are: $\theta_0 = -0.25$, $\Omega = 0.7128281828459045$, $K = 0.5$. The values of the constants are chosen by considering the results of the experiments in our previous work of constructing PRGs based on circle map [9].

2.2. Modified Rotation Equations

A Modified form of rotation equation, presented in [11,12] is rarely used in encryption algorithms, but in our previous work we have determined the good chaotic properties of the Modified rotation equations [13]. The formula we used is given by:

$$\begin{aligned} x_{t+1} &= -a - (x_t - a)\cos\theta + y_t \sin\theta / r_t \\ y_{t+1} &= -x_t r_t \sin\theta - y_t \cos\theta \\ r_t &= \sqrt{0.5(x_t^2 + \sqrt{x_t^4 + 4y_t^2})}, \end{aligned} \tag{2}$$

where the parameters for chaotic behavior are $\theta = 2$ and $a = 2.8$. The initial values we used are $x_0 = 0.2343214592$ and $y = -0.742190593$. The plots of 5000 points with the initial values are shown in Figure 1.



(**a**) Primary view (**b**) 3D view

Figure 1. Modified rotation equations with $\theta = 2$, $a = 2.8$, $x_0 = 0.2343214592$ and $y = -0.742190593$.

2.3. Pseudo-Random Generation Algorithm

The chaotic formulas presented in Sections 2.1 and 2.2 are used for pseudo-random binary sequence generation by following the next steps:

Step 1: The initial values θ, Ω and K from Equation (1) are determined.
Step 2: The initial values θ, a, x_0 and y_0 from Equation (2) are determined.
Step 3: The two chaotic maps from Equations (1) and (2) are iterated for M times, without extracting any results. This step is for additional security and extending the secret key.
Step 4: The current iteration of Equation (1) is used for obtaining θ and its decimal value is post-processed as follows:

$$s_i = mod(integer(\theta_i \times 10^9)), 2)$$

where $integer(x)$ returns the integer part of x, truncating the value at the decimal point, and $mod(x, y)$ returns the reminder after division.
Step 5: The current iteration of Equation (2) is used for obtaining y and its decimal value is post-processed as follows:

$$j_i = mod(integer(y_i \times 10^5)), 2)$$

where $integer(x)$ returns the integer part of x, truncating the value at the decimal point, and $mod(x, y)$ returns the reminder after division.
Step 6: Calculated values for s_i and j_i are combined with XOR operation to get a single output bit.
Step 7: Return to Step 4 until the necessary bit stream is reached.

2.4. Statistical Results

Using pseudo-random binary streams for cryptographic algorithms requires extended statistical analysis to ensure the randomness of the stream [14,15]. The PRG described in the previous section is tested by generating 1 billion bits and the binary sequence was subjected to the following analysis:

2.4.1. NIST-Test

NIST-Statistical Test Suite [16] is one of the most used software when the PRGs are concern. The NIST package evaluates the randomness by performing 17 tests over given binary sequence. The actual evaluation is made by dividing the 1,000,000,000 bits into 1000 subsequences with 1,000,000 bits. The results for all tests are presented in Table 1.

Table 1. NIST test suite results.

NIST Test	*p*-Value	Pass Rate
Frequency (monobit)	0.875539	990/1000
Block-frequency	0.474986	986/1000
Cumulative sums (Forward)	0.231956	988/1000
Cumulative sums (Reverse)	0.404728	987/1000
Runs	0.737915	993/1000
Longest run of Ones	0.504219	991/1000
Rank	0.268917	988/1000
FFT	0.332970	985/1000
Non-overlapping templates	0.527913	990/1000
Overlapping templates	0.908760	984/1000
Universal	0.452173	989/1000
Approximate entropy	0.942198	988/1000
Random-excursions	0.587477	611/617
Random-excursions Variant	0.499704	611/617
Serial 1	0.395940	993/1000
Serial 2	0.448424	991/1000
Linear complexity	0.345650	991/1000

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 980 for a sample size = 1000 binary sequences. The minimum pass rate for the random excursion (variant) test is approximately = 603 for a sample size = 617 binary sequences. All the NIST tests have *p*-values in the acceptable range [0,1) indicating that all tests are passed.

2.4.2. DIEHARD-Test

DIEHARD software contains 19 test for randomness and we applied the tests again over the same bitstream of 1,000,000,000 bits provided by our PRG. The acceptable range for passing the individual tests once again is $[0, 1)$ for calculated *p*-values [17]. The results for all tests are presented in Table 2.

Table 2. DIEHARD statistical test results.

DIEHARD Test	*p*-Value
Birthday spacings	0.6936326
Overlapping 5-permutation	0.1870995
Binary rank (31 × 31)	0.3212080
Binary rank (32 × 32)	0.7507100
Binary rank (6 × 8)	0.4515738
Bitstream	0.4843810
OPSO	0.5597304
OQSO	0.4843929
DNA	0.5809645
Stream count-the-ones	0.4507350
Byte count-the-ones	0.5006025
Parking lot	0.5674729
Minimum distance	0.2135240
3D spheres	0.5417610
Squeeze	0.0204010
Overlapping sums	0.5881176
Runs up	0.8360805
Runs down	0.3161460
Craps	0.2423550

All the tests in Table 2 have *p*-values within the desired range meaning all the tests for randomness are passed.

2.4.3. ENT-Test

The last statistical test software we used for randomness determination is ENT [18]. The tests are 6—Entropy, Optimum compression, χ^2 distribution, Arithmetic mean value, Monte Carlo π estimation, and Serial correlation coefficient. The results are presented in Table 3.

Table 3. ENT statistical test results.

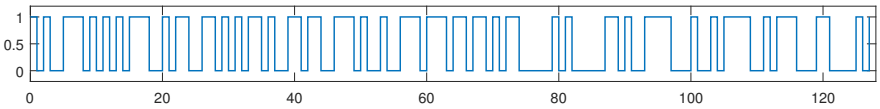
ENT Test	Result
Entropy	7.999999 bits per byte
Optimum compression	OC would reduce the size of this 125,000,000 byte file by 0%.
χ^2 distribution	For 125,000,000 samples is 253.43, and randomly would exceed this value 51.61% of the times.
Arithmetic mean value	127.4968 (127.5 = random)
Monte Carlo π estimation	3.141710066 (error 0.00%)
Serial correlation coefficient	−0.000087 (totally uncorrelated = 0.0)

2.4.4. Key Space Analysis

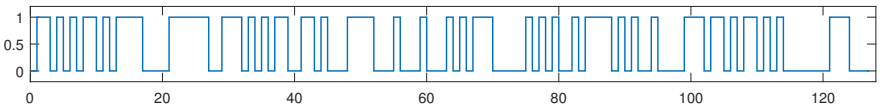
The key space is defined by initial values from Equations (1) and (2) and represents all the possibilities that can be used as a secret key for encryption. The parameters from the equations are not included in key space because their values do not change, so the included variables are θ_0 from Equation (1), x_0 and y_0 from Equation (2) and M from the proposed algorithm. Considering IEEE floating-point standard [19] the precision of 64bits double variables is about 10^{-15}. In our case we have three double variables so the final key space is about $10^{45} \approx 2^{149}$. This is large enough to resist against brute force attack methods [20].

2.4.5. Key Sensitivity Analysis

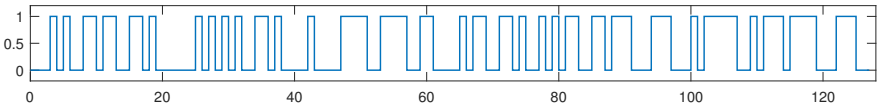
Key sensitivity test requires using very similar keys and comparing the results of the produced random sequences. To compare the bit streams produced by our PRG we used five different but very similar keys. For the first key (K1) we used the initial values from Sections 2.1 and 2.2. For the second key (K2)—θ_0 is changed to 0.251, for K3—θ_0 is 0.249, for K4—x_0 is 0.2343214593 and for K5—y_0 is -0.742190594. Figure 2 visually represents the completely different binary sequences produced when the different secret keys are used, even if they are very similar.



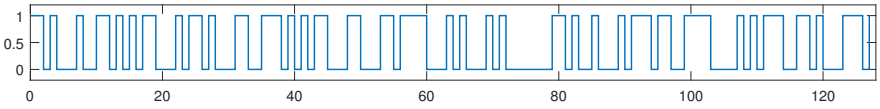
(**a**) Plot of binary sequence using secret key K1



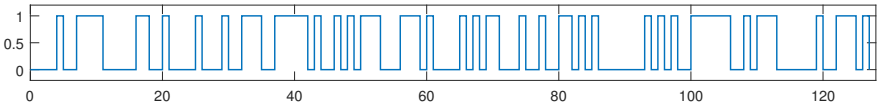
(**b**) Plot of binary sequence using secret key K2



(**c**) Plot of binary sequence using secret key K3



(**d**) Plot of binary sequence using secret key K4



(**e**) Plot of binary sequence using secret key K5

Figure 2. Plot of binary sequences with similar keys.

3. Audio Encryption Algorithm with Permutation-Substitution Architecture

In this section, we present an audio encryption/decryption algorithm using the presented pseudo-random generator described in Section 2.3 and previous research [21]. Considering the audio file structure our algorithm leaves the header bits intact and process only the audio data part of the files. For further empirical experiment header bits of the audio files are not modified because they contain information about the file size, number of samples, bits per sample etc. The audio data part is divided into samples with digital value representing the actual sound signal. Our scheme processes

the samples by shifting the bits in every sample, performing permutation and changing the values of the bits in the sample performing substitution.

3.1. Encryption Algorithm

The audio encryption algorithm consists of the following steps:

Step 1: The initial values from Equations (1) and (2) are determined then the PRG is iterated M times.
Step 2: The header bits from plain audio file A are transferred into file A' without cryptographic modifications.
Step 3: The audio data from file A is processed sample by sample.
Step 4: S bits are extracted from the proposed PRG, where S is the number of bits in the sample. The bits are converted into integer value—S_1
Step 5: Integer number P is calculated as follows: $P = S_1 modulo S$.
Step 6: The bits from current sample are shifted with the obtained value P from the previous step.
Step 7: The bits in the result sample are modified using XOR operation with the same amount of bits produced by the proposed PRG.
Step 8: The encrypted sample from Step 7 is transferred into file A'.
Step 9: Repeat Steps 4–8 until end of plain file A is reached.
Step 10: The produced output file A' is the final encrypted audio file.

Figure 3 illustrates the encryption process.

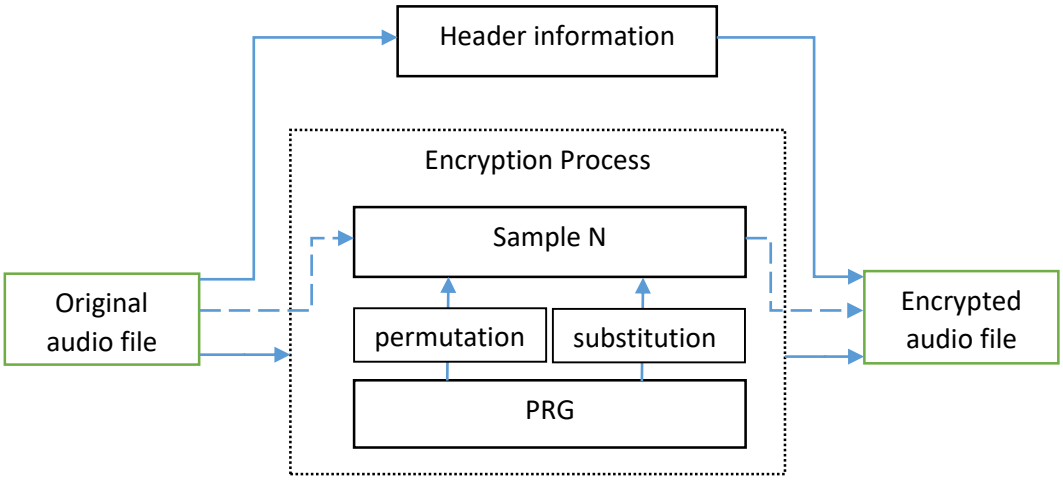


Figure 3. Encryption scheme.

3.2. Decryption Algorithm

The description method needs to consider the linear bits output of the PRG, but the opposite order of the bit-shifting and bit-modification in every sample. The decryption steps are:

Step 1: The initial values from Equations (1) and (2) are determined then the PRG is iterated M times.
Step 2: The header bits from encrypted audio file A are transferred into file A' without cryptographic modifications.
Step 3: The audio data from file A is processed sample by sample.
Step 4: S bits are extracted from the proposed PRG, where S is the number of bits in the sample. The bits are converted into integer value—S_1
Step 5: Integer number P is calculated as follows: $P = S_1 modulo S$.
Step 6: The bits in the result sample are modified using XOR operation with the same amount of bits produced by the proposed PRG.

Step 7: The bits from current sample are shifted back with the obtained value P from the step 5.
Step 8: The result sample from Step 7 is transferred into file A'.
Step 9: Repeat Steps 4–8 until end of plain file A is reached.
Step 10: The produced output file A' is the final decrypted audio file.

Figure 4 illustrates the decryption process.

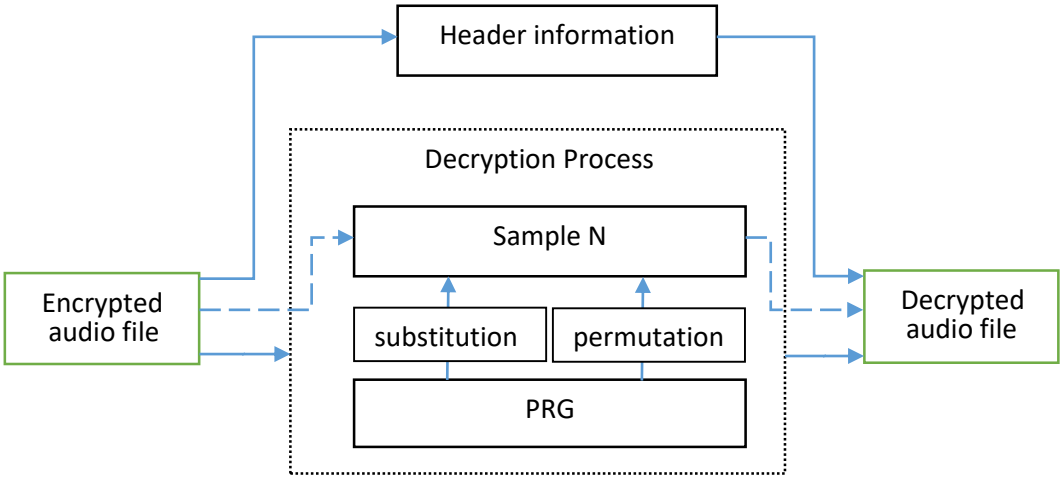


Figure 4. Encryption scheme.

The encryption and the decryption methods are implemented using programming language C++ for further evaluation by performing extended cryptographic analysis presented in the next section.

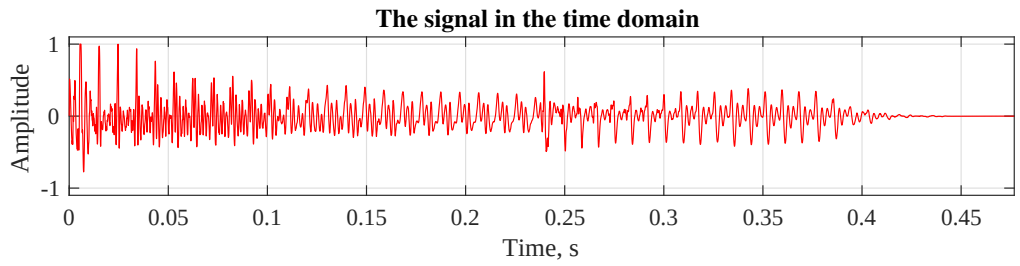
4. Cryptographic Analysis

The main purpose of the cryptographic analysis is restoring the plain message from the encrypted message. In this section, in order to prove the audio encryption efficiency, we performed various empirical tests to compare plain files and their corresponding encrypted files.

4.1. Waveform Plotting

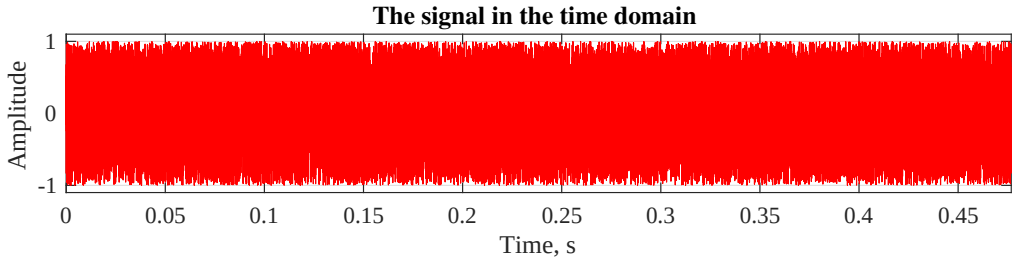
One of the most common approaches, concerning audio signal analysis is waveform plotting to display the audio signal amplitude distributed in time. To compare the plain audio files with the encrypted ones we present the visualization of one of the tested files. Figure 5a represents the waveform of normal file before encryption, Figure 5b represents the changes in the file after encryption and Figure 5c demonstrates the restored file after decryption.

The difference between the plain file plot and the encrypted file plot is indication of successful encryption. Furthermore, the strong difference also means the original file cannot be restored even partially.

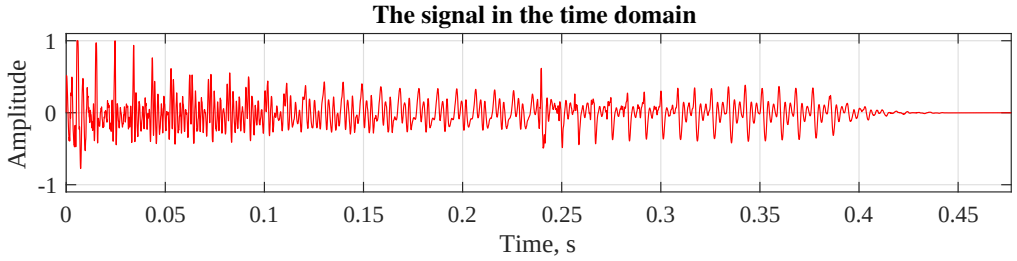


(**a**) Plot of plain file

Figure 5. *Cont.*



(**b**) Plot of encrypted file

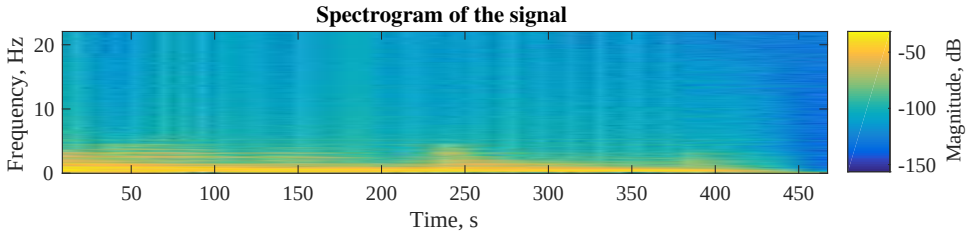


(**c**) Plot of decrypted file

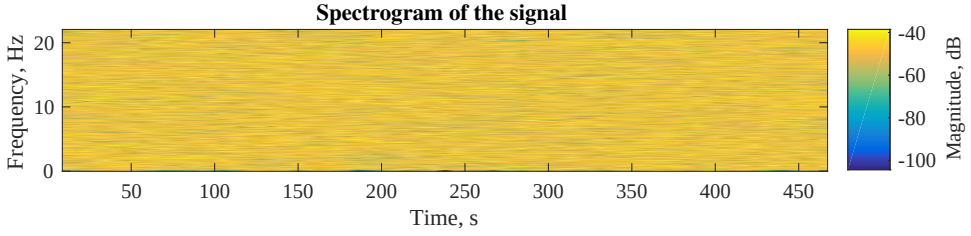
Figure 5. Waveform plotting.

4.2. Spectrogram Plotting

The spectrogram plotting is another important approach for analyzing audio signals. In this case the main focus is the frequency of the sound against time domain. Comparing plain files with encrypted files allows us to see the difference between the files and to evaluate the proposed audio encryption algorithm. Figure 6a shows the spectrogram of plain file, Figure 6b represents the changes in the file after encryption and Figure 6c demonstrates the restored file after decryption. The spectrogram plot of the encrypted file means the frequency of the original signal in the plain file is completely destroyed. This test is another indicator of the high encryption properties of the proposed audio encryption algorithm.

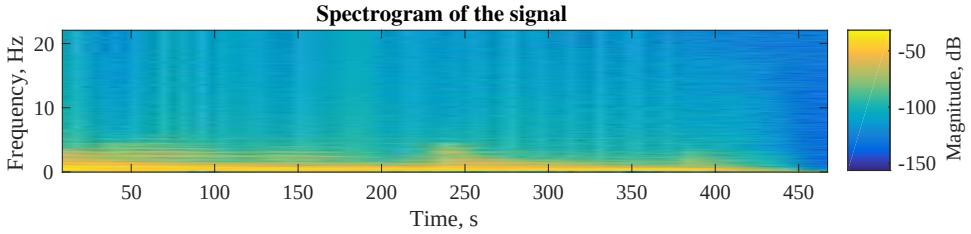


(**a**) Plot of plain file



(**b**) Plot of encrypted file

Figure 6. *Cont.*



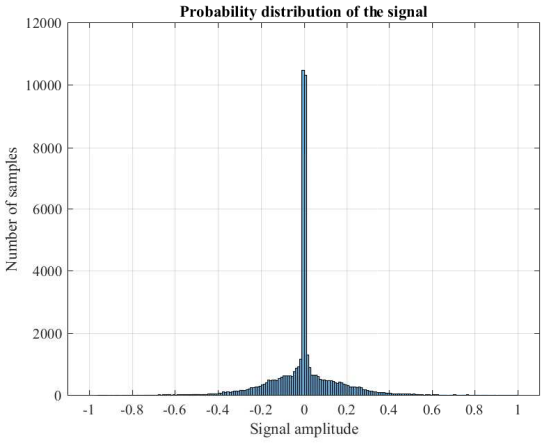
(**c**) Plot of decrypted file

Figure 6. Spectrogram plotting.

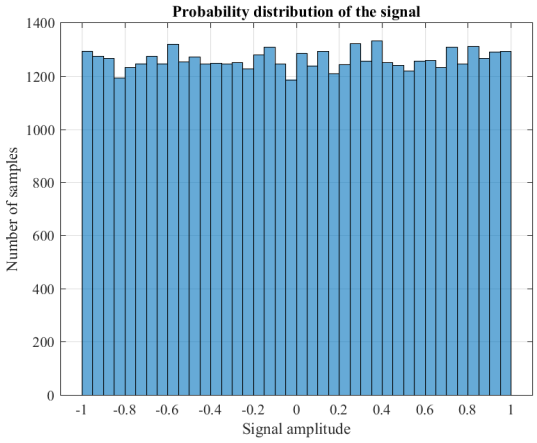
4.3. Histogram Analysis

The histograms are common tool to measure the distribution of values. Evaluating audio signal with histogram diagrams is excellent method to determine the distribution of the samples values in the audio files.

Figure 7a shows the histogram of one of the tested plain files and Figure 7b shows the histogram of the corresponding encrypted file. The distribution of the values in Figure 7b are very close and uniform, indicating strong encryption. The close values, also indicates resistance against attacks.



(**a**) Histogram of plain file



(**b**) Histogram of encrypted file

Figure 7. Histograms of plain file and encrypted file.

4.4. Correlation Analysis

Measuring correlation coefficient between two audio files express the dependency between their corresponding sample values. This is another statistical evaluation for testing the quality encryption algorithms. Calculating correlation coefficient determines the level of correlation between two files and the correlation coefficient is always in range $[-1, 1]$. Values between $|1–0.7|$ is considered as strong correlation (samples from the plain files are similar to samples from the encrypted file), correlation between $|0.7–0.3|$ is considered as medium correlation and values between $|0.3–0|$ is considered as weak correlation.

Correlation coefficient can be calculated as follows:

$$r_{xy} = \frac{cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}, \tag{3}$$

where

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2,$$

$$D(y) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \overline{y})^2,$$

$$cov(x,y) = \sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y}),$$

N is the total number of samples, x_i and y_i are the sample values of the plain and encrypted files, $\overline{x}$ and $\overline{y}$ are the mean values of samples, and finally $cov(x,y)$ is covariance between both files.

Table 4 shows the obtained result values from our tests.

Table 4. Correlation between plain and encrypted audio files.

File	File Size	File Length (s)	Correlation Coefficient
File1.wav	41.1 kb	0.47 s	−0.004794
File2.wav	98.6 kb	1.14 s	0.001699
File3.wav	138 kb	1.60 s	−0.00038781
File4.wav	277 kb	3.21 s	−0.00088715
File5.wav	544 kb	6.32 s	−0.0011166
File6.wav	1.08 mb	12.91 s	0.00053012
File7.wav	2.33 mb	13.85 s	0.00047104
Ref. [3]	-	7 s	0.0119
Ref. [4]	-	-	0.0263

The results in Table 4 indicates values close to zero which means there is no dependence between the two files. The results also mean high quality of the encryption.

4.5. Number of Sample Change Rate

Number of sample change rate (NSCR) is robustness test for establishing the quality of encryption algorithms. The purpose of the test is to compare the corresponding sample values of the original and encrypted audio files and to show the difference in percents. NSCR can be calculated as follows:

$$NSCR = \frac{\sum_{i=1}^{N} D_i}{N} \times 100\%, \quad (4)$$

where

$$D_i = \begin{cases} 1, x_i \neq y_i \\ 0, Otherwise \end{cases}$$

In Equation (4), N is the total number of samples, x_i and y_i are the corresponding sample values of the plain and encrypted files.

Table 5 shows the obtained result values from our tests.

The results clearly demonstrate the complete difference between the original and encrypted files, indicating high security level of the proposed audio encryption scheme.

Table 5. Number of sample change rate.

File	File Size	File Length (s)	NSCR
File1.wav	41.1 kb	0.47 s	100%
File2.wav	98.6 kb	1.14 s	99.996%
File3.wav	138 kb	1.60 s	99.9972%
File4.wav	277 kb	3.21 s	99.9993%
File5.wav	544 kb	6.32 s	99.9996%
File6.wav	1.08 mb	12.91 s	99.9986%
File7.wav	2.33 mb	13.85 s	99.9984%
Ref. [3]	-	-	99.9996%
Ref. [22]	-	-	99.9992%

4.6. Signal to Noise Ratio

Signal to Noise Ratio (SNR) is widely user to determine the quality of the signals [2,23]. Values grater than 0 dB indicates the clear signal is more than the noise. For this test we need both plain and encrypted audio files and SNR is calculated as follows:

$$SNR = 10log_{10}\frac{\sum_{i=1}^{N} x_i^2}{\sum_{i=1}^{N}[x_i - y_i]} \text{ (dB)}, \tag{5}$$

where x_i and y_i are corresponding sample values from audio files, and N is number of samples.

The results from our SNR tests are shown in the next Table 6.

Table 6. Signal to noise ratio.

File	File Size	File Length (s)	SNR
File1.wav	41.1 kb	0.47 s	−16.0483 dB
File2.wav	98.6 kb	1.14 s	−10.6478 dB
File3.wav	138 kb	1.60 s	−8.7189 dB
File4.wav	277 kb	3.21 s	−10.6345 dB
File5.wav	544 kb	6.32 s	−15.3072 dB
File6.wav	1.08 mb	12.91 s	−14.3761 dB
File7.wav	2.33 mb	13.85 s	−16.0483 dB
Ref. [2]	-	4 s	−12.1727 dB
Ref. [2]	-	7 s	−12.5180 dB

All the obtained values for SNR are negative which means the encrypted files are very noisy and the encryption method completely destroys the clear signal from plain audio files.

4.7. Peak Signal to Noise Ratio

Peak Signal to Noise Ratio (PSNR) is different approach to measure the power of clean signal against the power of noise. PSNR is more applicable for image encryption algorithms, but can be used for testing the quality of the proposed encryption scheme in this paper. PSNR is calculated as follows:

$$PSNR = 10\log_{10}\frac{MAX^2}{MSE}\text{dB}, \tag{6}$$

where MAX is the maximum possible value of audio stream (In our case the maximum value is 65,535) and MSE is the mean square error between the plain and encrypted file. MSE is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2, \tag{7}$$

where N is the total number of samples, x_i and y_i are the corresponding sample values of the plain and encrypted files.

Table 7 contains the results of our tests.

Table 7. Peak signal to noise ratio.

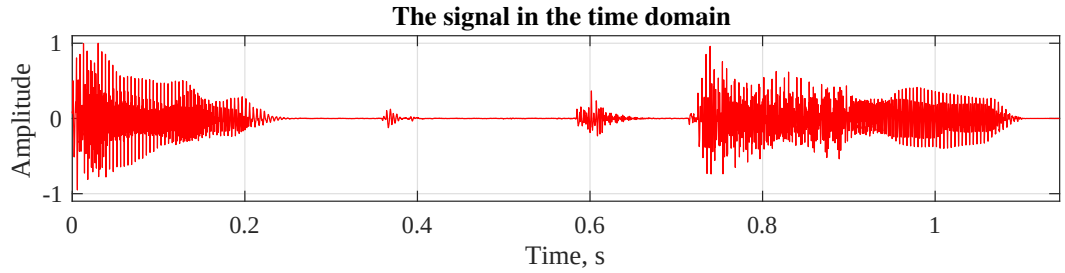
File	File Size	File Length (s)	PSNR
File1.wav	41.1 kb	0.47 s	1.4524 dB
File2.wav	98.6 kb	1.14 s	4.3909 dB
File3.wav	138 kb	1.60 s	2.3292 dB
File4.wav	277 kb	3.21 s	−1.1625 dB
File5.wav	544 kb	6.32 s	−0.53082 dB
File6.wav	1.08 mb	12.91 s	1.5163 dB
File7.wav	2.33 mb	13.85 s	1.4524 dB
Ref. [1]	12.17 kb	-	4.5299 dB
Ref. [4]	-	-	4.373 dB

All the obtained values for PSNR are close to zero (or below) indicating very high level of noise in the encrypted audio files.

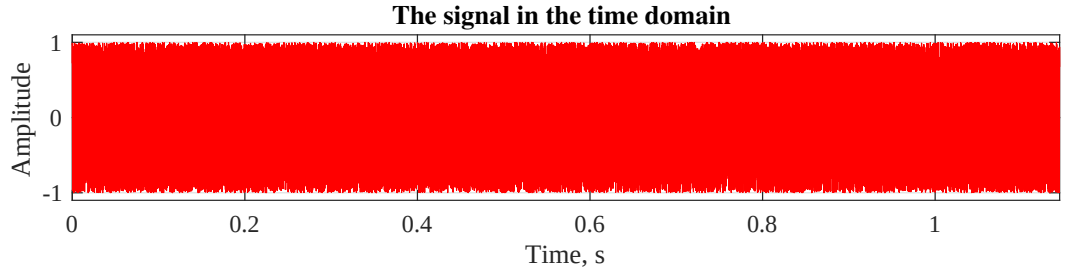
4.8. Encryption/Decryption Key Sensitivity

In Section 2.4.5 we performed key sensitivity test of the PRG used for proposed the audio encryption algorithm in this paper. Analyzing the general behavior of the encryption method we used very similar keys to encrypt and to restore encrypted audio file. The decryption key is obtained by changing a single digit from one of the variables constructing the key space. Key 1 and Key 2 are described in Section 2.4.5.

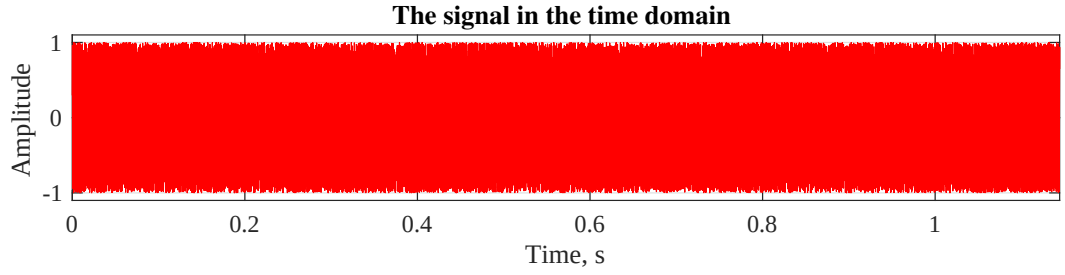
Both Figures 8 and 9 demonstrate that the decryption is unsuccessful even with very similar secret key. Changing a single digit of the key leads to fail decryption. The experiment is proof of high key sensitivity concerning the proposed audio encryption algorithm.



(**a**) Waveform of plain file

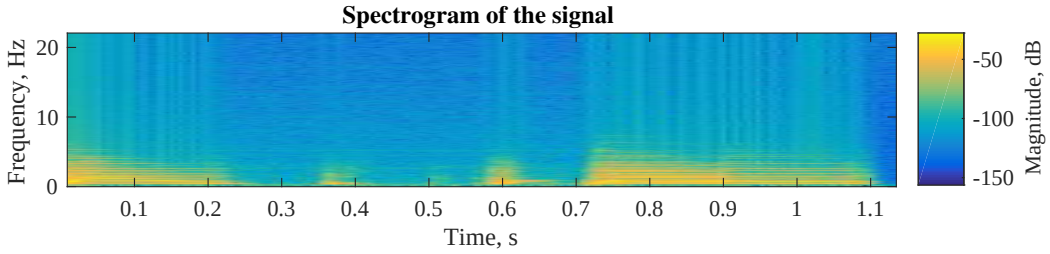


(**b**) Waveform of encrypted file using secret key K1

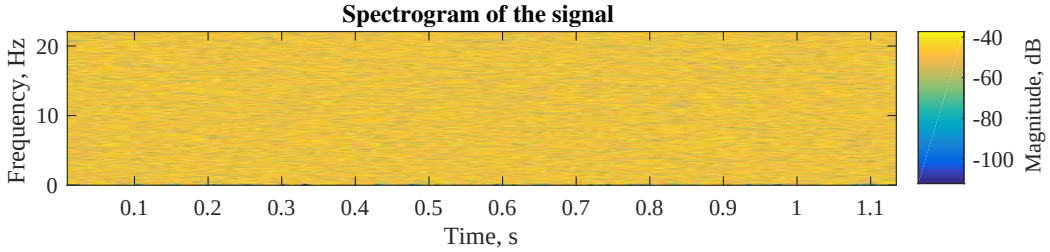


(**c**) Waveform of decrypted file using secret key K1

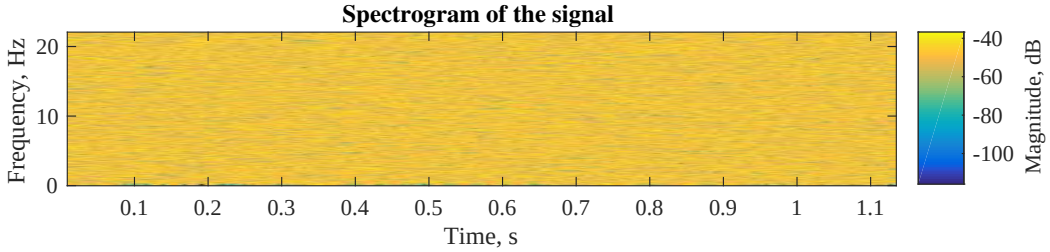
Figure 8. Waveform plotting-key sensitivity.



(**a**) Spectrogram of plain file



(**b**) Spectrogram of encrypted file using secret key K1



(**c**) Spectrogram of decrypted file using secret key K2

Figure 9. Spectrogram plotting-key sensitivity.

4.9. Speed Performance

To measure the necessary encryption time we used audio files with different size with hardware configuration—2.40 GHz Intel® Core™ i7-3630QM Dell Inspiron, 8 GB RAM, Windows 7. Table 8 contains the results of our tests.

Table 8. Speed performance test.

File	File Size	Bytes per Sample	Encryption Time (s)
File1.wav	41.1 kb	2	0.130 s
File2.wav	98.6 kb	2	0.245 s
File3.wav	138 kb	2	0.333 s
File4.wav	277 kb	2	0.694 s
File5.wav	544 kb	2	1.324 s
File6.wav	1.08 mb	2	2.688 s
File7.wav	2.33 mb	2	5.767 s

5. Conclusions

This paper evaluates a new design for audio files encryption algorithm. The proposed cryptographic algorithm relies on permutation-substitution architecture realized by using chaotic circle map and modified rotation equations. Extended cryptographic analysis is performed for testing the proposed method for security. The waveform plots and the spectrograms of the tested audio files demonstrate the changes in encrypted files compared to plain files. The correlation analysis and NSCR tests confirm the high quality of encryption, demonstrating the sample values are completely different

in corresponding files. The measured SNR and PSNR values show high levels of noise in the encrypted files, indicating the original signal is destroyed in the encryption process. Key space analysis shows the necessary level of security against brute-force attacks and key sensitivity analysis shows that even minimal change of the secret key leads to unsuccessful decryption. Considering the obtained results during the cryptographic analysis, we can conclude that the proposed algorithm has the necessary cryptographic security for audio files encryption.

Funding: This research received no external funding.

Acknowledgments: The author is grateful to the anonymous referees for valuable and helpful comments. This work is supported by the Scientific research fund of Konstantin Preslavski University of Shumen under the grant No. RD-08-71/29.01.2019 and by European Regional Development Fund and the Operational Program "Science and Education for Smart Growth" under contract UNITe No. BG05M2OP001-1.001-0004-C01 (2018–2023).

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
MSE	Mean Square Error
NSCR	Number of sample change rate
PRG	Pseudo-random generator
PSNR	Peak Signal to Noise Ratio
SNR	Signal to Noise Ratio

References

1. Liu, H.; Kadir, A.; Li, Y. Audio encryption scheme by confusion and diffusion based on multi-scroll chaotic system and one-time keys. *Optik* **2016**, *127*, 7431–7438. [CrossRef]
2. Hato, E.; Shihab, D. Lorenz and Rossler Chaotic System for Speech Signal Encryption. *Int. J. Comput. Appl.* **2015**, *128*, 09758887. [CrossRef]
3. Sathiyamurthi, P.; Ramakrishnan, S. Speech encryption using chaotic shift keying for secured speech communication. *EURASIP J. Audio Speech Music Process.* **2017**, *2017*, 20. [CrossRef]
4. Tamimi, A.A.; Abdalla, A.M. An Audio Shuffle-Encryption Algorithm. In Proceedings of the World Congress on Engineering and Computer Science, San Francisco, CA, USA, 22–24 October 2014; Volume 1.
5. Preishuber, M.; Hütter, T.; Katzenbeisser, S.; Uhl, A. Depreciating motivation and empirical security analysis of chaos-based image and video encryption. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2137–2150. [CrossRef]
6. Kordov, K.M. Modified Chebyshev Map Based Pseudo-random Bit Generator. *AIP Conf. Proc.* **2014**, *1629*, 432–436.
7. Kordov, K. Signature Attractor Based Pseudorandom Generation Algorithm. *Adv. Stud. Theor. Phys.* **2015**, *9*, 287–293. [CrossRef]
8. Stoyanov, B. Pseudo-random bit generation algorithm based on Chebyshev polynomial and Tinkerbell map. *Appl. Math. Sci.* **2014**, *8*, 6205–6210. [CrossRef]
9. Kordov, K. Modified pseudo-random bit generation scheme based on two circle maps and XOR function. *Appl. Math. Sci.* **2015**, *9*, 129–135. [CrossRef]
10. Stoyanov, B.P. Using Circle Map in Pseudorandom Bit Generation. *AIP Conf. Proc.* **2014**, *1629*, 460–463.
11. Skiadas, C.H. Mathematical models of Chaos. In *Chaos Applications in Telecommunications*; Stavroulakis, P., Ed.; CRC Press: Boca Raton, FL, USA, 2006; pp. 383–413.
12. Skiadas, C.H.; Skiadas, C. *Chaotic Modelling and Simulation: Analysis of Chaotic Models, Attractors and Forms*; CRC Press: Boca Raton, FL, USA, 2008.
13. Stoyanov, B.P.; Zhelezov, S.K.; Kordov, K.M. Least significant bit image steganography algorithm based on chaotic rotation equations. *C. R. Acad. Bulgare Sci.* **2016**, *69*, 845–850.
14. Parvees, M.M.; Samath, J.A.; Bose, B.P. Cryptographically Secure Diffusion Sequences—An Attempt to Prove Sequences Are Random. In *Advances in Big Data and Cloud Computing*; Springer: Singapore, 2019; pp. 433–442.

15. Tutueva, A.V.; Butusov, D.N.; Pesterev, D.O.; Belkin, D.A.; Ryzhov, N.G. Novel normalization technique for chaotic Pseudo-random number generators based on semi-implicit ODE solvers. In Proceedings of the 2017 International Conference "Quality Management, Transport and Information Security, Information Technologies" (IT&QM&IS), St. Petersburg, Russia, 24–30 September 2017; pp. 292–295.
16. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Application*; NIST Special Publication 800-22; NIST: Gaithersburg, MD, USA, 2001.
17. Marsaglia, G. *The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness*; Florida State University: Tallahassee, FL, USA, 1995.
18. Walker, J. ENT: A Pseudorandom Number Sequence Test Program. Available online: http://www.fourmilab.ch/random/ (accessed on 7 April 2019).
19. IEEE Computer Society. *754-2008—IEEE Standard for Floating-Point Arithmetic, Revision of ANSI/IEEE Std 754-1985*; IEEE: New York, NY, USA, 2018. [CrossRef]
20. Alvarez, G.; Li, S. Some Basic Cryptographic Requirements for Chaos-Based Cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
21. Kordov, K.; Bonchev, L. Using circle map for audio encryption algorithm. *Math. Softw. Eng.* **2017**, *3*, 183–189.
22. Lima, J.B.; da Silva Neto, E.F. Audio encryption based on the cosine number transform. *Multimedia Tools Appl.* **2016**, *75*, 8403–8418. [CrossRef]
23. Kabakchiev, H.; Behar, V.; Garvanov, I.; Kabakchieva, D.; Kabakchiev, A.; Rohling, H.; Bentum, M.; Fernandes, J. Comparison of Two Algorithms for Signal Detection in Pulsarbased FSR. In Proceedings of the 2018 19th International Radar Symposium (IRS), Bonn, Germany, 20–22 June 2018; pp. 1–9.