Mathematical and Software Engineering, Vol 10, No 1-2 (2024), 1–12.
Varεpsilon Ltd, https://varepsilon.com/index.php/mse/issue/archive
DOI 10.5281/zenodo.14650067

Audio encryption algorithm employing data-parallel approach

Dimitar Dobrev and Tsvetelina Ivanova

Department Computer Informatics, Faculty of Mathematics and Informatics,
Konstantin Preslavski University of Shumen, Bulgaria

Abstract

In this paper we present a parallel algorithm for encrypting audio files. The parallel algorithm is designed using a data-parallel approach and uses a pseudo-random byte generator based on Ikeda and Tinkerbell - mathematical functions of chaos theory. Analysis of the encrypted audio and image files shows that our proposed scheme performs comparably to other existing algorithms.

Keywords: audio encryption, chaotic functions, Ikeda map, Tinkerbell map, parallel data processing, security analysis

1 Introduction

With the rapid growth of information sharing, ensuring data security has become a crucial challenge. Today, it has become increasingly easier for people to send different types of multimedia messages, such as audio files. Audio messages can contain sensitive information ranging from personal conversations to confidential business conversations, which demands secure, reliable, and resistant to attacks or vulnerability protection methods against unauthorized access, since fast-phased technology development also means many more methods to attack multimedia data.

Compared to images and text, audio files are often larger, and traditional encryption methods such as AES [1] and RSA [2] may not always be suitable for large audio data due to their computational complexity and the amount of processing power required. As a result, handling audio data by balancing efficiency and security has become a huge challenge. Given these considerations, exploring different encrypting techniques that maintain strong security and efficiency, is important. Chaotic functions from chaos theory are known for their unpredictability and sensitivity to initial conditions and are a promising approach for encryption algorithms. In Ref. [3] a mixture of chaotic functions is used for a novel audio data encryption algorithm. Another encryption scheme was presented in [4], where the chaotic Ikeda function is used.

Parallel computation is an approach that makes encryption algorithms faster. In [5], [6], [7], [8], [9], parallel encryption schemes based on logistic map are presented. In this study, our focus is on the encryption of wave audio files, employing a parallel approach. The

wave file consists of two parts, the header part and the sample part. By using a pseudo-random byte generator, we generate a pseudo-random byte sequence. Logical bitwise XOR operation is performed on this sequence and on the sample part of the wave file to encrypt the audio file. To speed up this process, we suggest splitting the samples into several parts and allowing each part to be processed on a different thread simultaneously.

The main contribution of this study can be summarized as follows:

- A pseudo-random byte generation scheme based on a mixture of chaotic functions is applied in parallel data encryption algorithm;
- The parallel data encryption algorithm is modified to encrypt audio files;
- We examined the encryption scheme and the results are comparable with other encryption algorithms.
- We assume that the parallel data encryption scheme is applicable not only to audio files but to other multimedia files such as images and video.

2 Pseudo-random Byte Generator Based on Ikeda and Tinkerbell Chaotic Functions

A novel pseudo-random byte generator using Ikeda and Tinkerbell chaotic functions has been introduced in [10] by one of the authors (DD) in collaboration with Borislav Stoyanov. They conducted extensive statistical analyses and proved that this generator produces highly unpredictable byte sequences.

2.1 Ikeda and Tinkerbell Chaotic Functions Description

Ikeda map [11], [12] is a two-dimensional chaotic function described as follows:

$$z_{n+1} = 1 + u \cdot (z_n \cdot \cos t_n - w_n \cdot \sin t_n) \tag{1}$$

$$w_{n+1} = 1 + u \cdot (z_n \cdot \sin t_n - w_n \cdot \cos t_n) \tag{2}$$

where u is a parameter and

$$t_n = 0.4 - \frac{6}{1 + z_n^2 + w_n^2} \tag{3}$$

The Tinkerbell map [13] is a two-dimensional discrete-time dynamical system given by the equations:

$$x_{n+1} = x_n^2 - y_n^2 + a \cdot x_n + b \cdot y_n \tag{4}$$

$$y_{n+1} = 2 \cdot x_n \cdot y_n - c \cdot x_n + d \cdot y_n \tag{5}$$

where a, b, c and d are constants.

2.2 Pseudo random byte generation algorithm

The proposed generator consists of the following steps:

1. Four initial values $z_0, w_0, x_0, y_0 \in \mathbb{R}$ are determined for the Ikeda and Tinkerbell chaotic functions, respectively.
2. z_k, w_k, x_k, and y_k are calculated using equations (1), (2), (3), (4) and (5) where k is a constant.
3. For each $i > k$, four real numbers are generated: z_i, w_i, x_i, and y_i.
4. Four non-negative integer numbers p_i, q_i, r_i, and s_i are obtained as follows:

$$\begin{aligned} p_i &= |\lfloor z_i \cdot 10^9 \rfloor \mod 256| \\ q_i &= |\lfloor w_i \cdot 10^9 \rfloor \mod 256| \\ r_i &= |\lfloor x_i \cdot 10^9 \rfloor \mod 256| \\ s_i &= |\lfloor y_i \cdot 10^9 \rfloor \mod 256| \end{aligned}$$

5. A single byte is obtained by performing an XOR operation on p_i, q_i, r_i, and s_i.
6. Steps 3 to 5 are repeated until the desired output length is reached.

3 Parallel encryption algorithm

The described pseudo-random byte generation algorithm (from Section 2.2) has been utilized in a parallel data encryption algorithm [10]. This algorithm is slightly modified to match wave files and consists of the following steps:

1. The initial values z_{0j}, w_{0j}, x_{0j}, and y_{0j}, for $1 \leq j \leq p$, of p pseudo-random generators (described in Section 2.2) are determined, where p is a constant.
2. The values of z_{kj}, w_{kj}, x_{kj}, and y_{kj} are calculated, where k is a constant, and $1 \leq j \leq p$, using the chaotic equations (described in Section 2.1), each on an available thread.
3. The plain audio file is divided into $b = \left\lceil \frac{l}{B} \right\rceil$ disjoint blocks, where l is the length of the plain file in samples and B is the size of each block (B being a constant such that $p|B$). The block is filled with zeros if $B > l$.
4. For each $1 \leq i \leq b$, read the i^{th} block from the plain audio file. The i^{th} block consists of B non-negative integer numbers with values $< 2^{16}$.
5. The i^{th} block is divided into p disjoint sub-blocks, each of size $\frac{B}{p}$.
6. For each sub-block s_j $(1 \leq j \leq p)$, if a thread is available, $2 \cdot \frac{B}{p}$ random bytes $byte_u$ $(1 \leq u \leq 2 \cdot \frac{B}{p})$ are generated using the j^{th} pseudo-random generator. These bytes are transformed into $\frac{B}{p}$ non-negative integer numbers $t_v = 2^8 \cdot byte_{2 \cdot v-1} + byte_{2 \cdot v}$

$(1 \leq v \leq \frac{B}{p}$ and $t_v < 2^{16})$, to match the number and values of each sub-block. Perform an XOR operation between the elements of s_j and t_v. Repeat step 6 until the i^{th} block is processed.

7. Steps 4 to 6 are repeated until all the blocks are processed.
8. Processed blocks are merged in one output file.

4 Metrics

The metrics (constants and initial values) of the pseudo-random byte generator and the metrics used in the two algorithms are as follows:

- $p = 8$, $k = 5724$, $B = 3 \cdot 2^{24}$.
- Ikeda map properties: $u = 9.7941$
  - $z_{01} = 0.354355462887$, $w_{01} = -0.452345435887$
  - $z_{02} = -0.534534532987$, $w_{02} = -0.423423423987$
  - $z_{03} = -0.143223462887$, $w_{03} = 0.543532432887$
  - $z_{04} = -0.435345332987$, $w_{04} = -0.765432832987$
  - $z_{05} = -0.423432342887$, $w_{05} = 0.343224142887$
  - $z_{06} = -0.423424535987$, $w_{06} = -0.435425433437$
  - $z_{07} = -0.765767562887$, $w_{07} = -0.435433543537$
  - $z_{08} = 0.436534532987$, $w_{08} = 0.235435435437$
- Tinkerbell map properties:
  - $a = 0.9$, $b = -0.64$, $c = 2$, $d = 0.5$
  - Initial values:
    - $x_{01} = -0.12546$, $y_{01} = 0.47236$
    - $x_{02} = -0.13546$, $y_{02} = 0.17236$
    - $x_{03} = -0.16747$, $y_{03} = 0.43242$
    - $x_{04} = -0.12342$, $y_{04} = -0.98739$
    - $x_{05} = -0.19992$, $y_{05} = 0.53452$
    - $x_{06} = -0.16753$, $y_{06} = -0.42235$
    - $x_{07} = -0.16898$, $y_{07} = -0.78878$
    - $x_{08} = -0.12334$, $y_{08} = -0.97654$

5 Results and Discussion

The algorithm described in Section 3 is implemented using OpenMP in the C++ programming language. All experiments are conducted on a laptop with an Intel(R) Core(TM) i7-8550U CPU @ 1.80 GHz (2.00 GHz), 4 cores, 8 threads, hyper-threading activated; RAM: 8 GB; cache: 8 MB.

Three audio files are encrypted and analyzed. These audio files are obtained from the website `https://freesound.org/` and are as follows:
`689113__gaijinbuzz__psalterium16.wav` with a size of 385984 bytes and a duration of

4.376 seconds,
`352011__luisaguacate__20160812_1025181.wav` with a size of 1318912 bytes and a duration of 29.907 seconds, and
`728410__darklitstudio__scary-horror-sound-effectambience-scratchy-shock.wav` with a size of 8988062 bytes and a duration of 101.905 seconds. We will call these files wav1, wav2 and wav3 respectively.

5.1 Key Space Analysis

The set of $z_{0j}, w_{0j}, x_{0j}, y_{0j} \in \mathbb{R}, 1 \leq j \leq p$ is the key space. The cardinality of this set is 2^{4wp}, where w is the length of the machine word. Following the IEEE standard for floating point numbers [14], the 64-bit double-precision number has about 10^{-15} computational precision, therefore $w \approx 49$. If, for example, $p = 8$, then $2^{4 \cdot 49 \cdot 8} = 2^{1568}$ seeds are possible. Assuming that some of the constants can also be considered as part of the key, the number of initial seeds can be even greater.

5.2 Waveform plotting

Waveform plotting is a two-dimensional graphic that represents an audio file's amplitude distribution over time [15]. Figures 1, 3, and 5 show waveform plotting of the plain files wav1, wav2, and wav3, while figures 2, 4, and 6 are graphics of the encrypted files. The graphics of the plain and their corresponding encrypted files are completely different which shows the quality of the proposed encryption scheme.

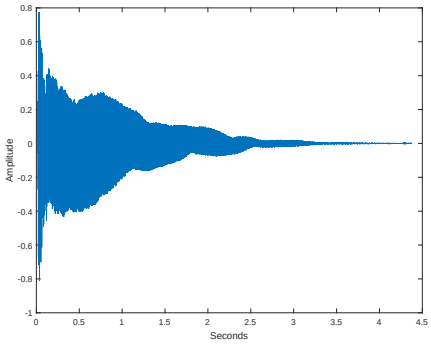


Figure 1: Plain signal (wav1).

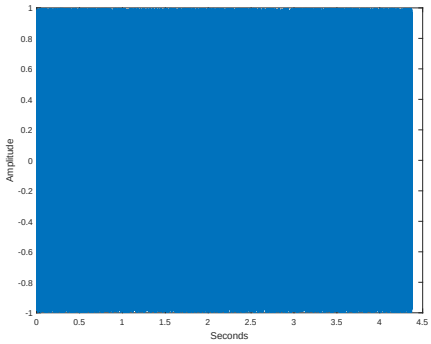


Figure 2: Encrypted signal (wav1).

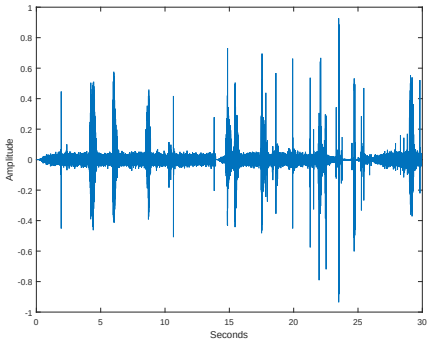


Figure 3: Plain signal (wav2).

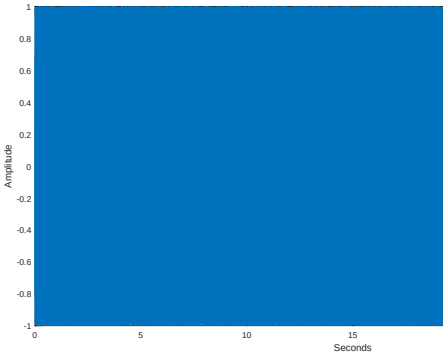


Figure 4: Encrypted signal (wav2).

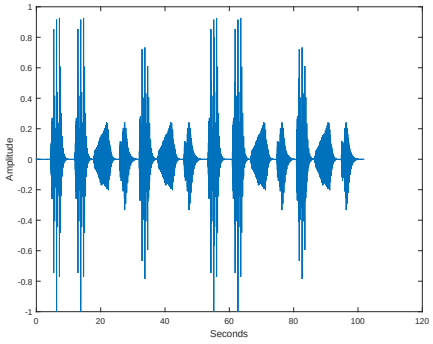


Figure 5: Plain signal (wav3).

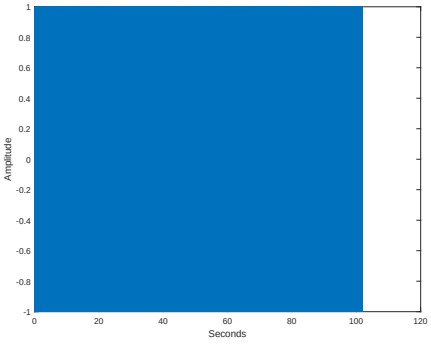


Figure 6: Encrypted signal (wav3).

5.3 Correlation Analysis

The correlation coefficient is a mathematical measure of the relationship between two sets of values [16]. In our case, these sets are the samples from both plain and encrypted audio files, which are non-negative integers ranging from 0 to $2^{16} - 1$. The correlation coefficient is a real number in the range $[-1.0, 1.0]$. The closer the value of the correlation coefficient is to 0.0, the weaker the linear relationship between the two sets of values.

Let a and a' represent the plain and encrypted audio files, respectively. The correlation coefficient $r_{a,a'}$ can be calculated as follows:

$$r_{a,a'} = \frac{cov(a, a')}{\sqrt{D(a)} \cdot \sqrt{D(a')}},$$

where:

$$D(a) = \frac{1}{n} \sum_{i=1}^{n} (a_i - \overline{a})^2,$$

$$D(a') = \frac{1}{n} \sum_{i=1}^{n} (a'_i - \overline{a'})^2,$$

$$cov(a, a') = \sum_{i=1}^{n} (a_i - \overline{a}) \cdot (a'_i - \overline{a'}),$$

n is the total number of samples, a_i and a'_i are the sample values, $\overline{a}$ and $\overline{a'}$ are the mean values, and $cov(a, a')$ is the covariance between the plain and encrypted audio files. The correlation coefficients are:

- wav1: -0.0283148
- wav2: 0.0479316
- wav3: -0.0131906

These test results are comparable with [4], [19], [20], and [21].

5.4 Number of Sample Change Rate (NSCR)

The Number of Sample Change Rate (NSCR) is a test used to compare the corresponding sample values of a plain file and its encrypted version, determining the percentage of different values between them. It helps evaluate the effectiveness of encryption by measuring how much the encrypted file differs from the original.

NSCR is calculated using the following formula:

$$NSCR = \frac{\sum_{i=1}^{n} D_i}{n} \cdot 100\%,$$

where n is the total number of samples (or frames) in the file, and:

$$D_i = \begin{cases} 1, & \text{if } a_i \neq a'_i \\ 0, & \text{if } a_i = a'_i \end{cases}$$

Here, a_i represents the value of the i^{th} sample in the plain file, and a'_i is the value of the corresponding sample in the encrypted file. The results are as follows:

- wav1: 99.9979%

- wav2: 99.9981%
- wav3: 99.9984%

Similar test results were made in [4], [19], [20], and [21].

5.5 Signal-to-Noise Ratio

The signal-to-noise ratio (SNR) is a test that measures the degree of corruption of the signal by a noise [17]. SNR is measured in dB and can be calculated as follows:

$$SNR = 10 \cdot \log_{10} \left(\frac{\sum_{i=1}^{n} a_i^2}{\sum_{i=1}^{n} (a_i - a_i')} \right) dB,$$

where a_i and a_i' are the i^{th} samples of the plain and encrypted audio files, and n is the total number of frames. The results are:

- wav1: -14.7431 dB
- wav2: -21.5998 dB
- wav3: -24.95658 dB

5.6 Peak Signal-to-Noise Ratio

The Peak Signal-to-Noise Ratio (PSNR) is a test that measures the quality of a signal after encryption [18]. It compares the maximum possible value of the signal to the level of noise or distortion present. For an audio file, the maximum possible value is $2^{16} - 1 = 65535$.

PSNR can be calculated using the following formula:

$$PSNR = 10 \cdot \log_{10} \left(\frac{R^2}{MSE} \right),$$

where R is the maximum possible value of the signal (65535 for 16-bit audio), and MSE is the Mean Squared Error between the original and encrypted audio files and

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (a_i - a_i')^2,$$

n is the total number of samples, a_i is the i^{th} sample of the original audio file, and a_i' is the i^{th} sample of the encrypted audio file. The results are as follows:

- wav1: 4.6253 dB
- wav2: 4.7453 dB
- wav3: 4.7581 dB

These results are similar to [4], [19], [20], and [21].

5.7 Time Complexity and Speed Performance

In theoretical computer science, time complexity describes an algorithm's time to run. The time complexity is usually measured by the number of elementary operations performed by the algorithm, assuming each operation takes a unit of time. The O (big O) notation describes the upper bound on the number of elementary operations an algorithm performs as the size of the input grows.

If we assume that the length of the machine word $w = O(1)$, the time complexity of the described algorithm is:

$$O\left(\frac{n}{min(p,p')}\right),$$

where n is the total number of frames in the plain audio file, p is a constant, described in section 2.3.1, and p' is the number of available cores of the machine, used for this algorithm. Our algorithm speed is 0.18 seconds for file wav1 and 1.32 seconds for wav2. We measured encryption time on wav3 using 1, 2, 4, and 8 threads - 4.08, 2.49, 1.73, and 1.42 seconds respectively.

5.8 Key sensitivity

Key sensitivity analysis is a test on how the encryption/decryption is sensitive to the values of the key set. Figures 7, 9, and 11 represent the wave plotting of the encrypted files wav1, wav2, and wav3. Figures 8, 10, and 12 are wave plotting of the files, obtained by trying to decrypt these files with a slightly modified key set. Wave plotting on these files shows that modifying the key set with a slight margin results in a very different outcome.

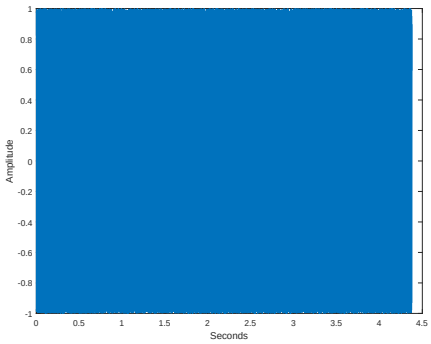


Figure 7: Encrypted signal (wav1).

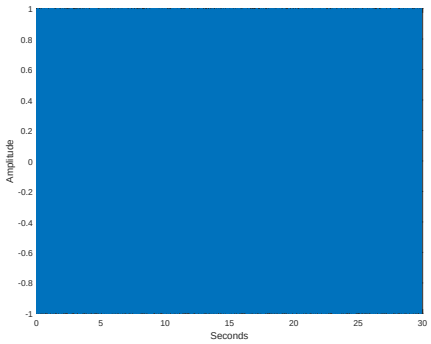


Figure 8: Unsuccessful decryption (wav1).

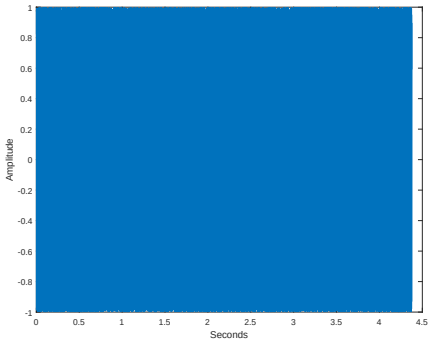


Figure 9: Encrypted signal (wav2).

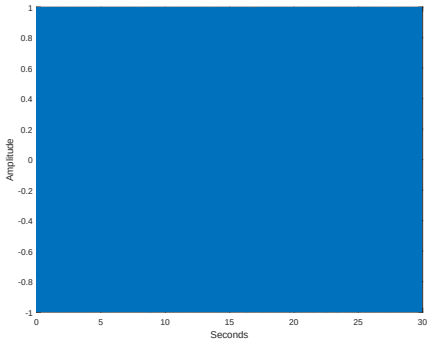


Figure 10: Unsuccessful decryption (wav2).

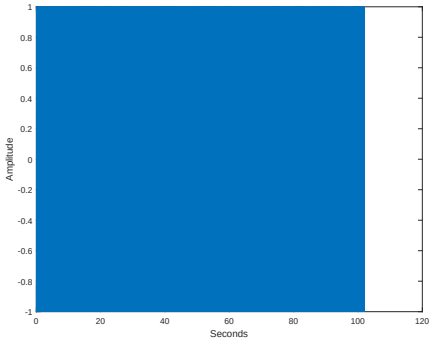


Figure 11: Encrypted signal (wav3).

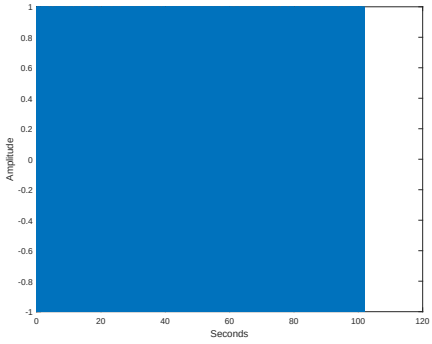


Figure 12: Unsuccessful decryption (wav3).

6 Conclusion

In this paper we used an existing data encryption algorithm employing a parallel approach to encrypt wave audio files. Key space analysis shows a high level of security. Waveform plotting of both plain and encrypted files demonstrates that encrypted files are completely different than plain. The number of samples change rate and correlation analysis shows little to non-existent relation between plain and encrypted files. A high noise level on encrypted files is proven by the peak signal-to-noise ratio test. The speed performance test demonstrates considerable speedup in time when running this algorithm on multiple threads. The algorithm is also very sensitive to the initial conditions - the initial key values. The tests that were conducted have shown that this encryption scheme has at least the same quality as others.

Acknowledgement

This work is funded by project RD-08-116/31.01.2024 from Shumen University.

References

[1] Rijmen, V.; Daemen, J. Advanced encryption standard (AES). In Federal Information Processing Standards Publications 197; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001; Available online: https://csrc.nist.gov/publications/ detail/fips/197/final (accessed on 1 September 2021).

[2] Rivest, R.L.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM 1978, 21, 120–126.

[3] Ghasemzadeh, A.; Esmaeili, E. A novel method in audio message encryption based on a mixture of chaos function. Int. J. Speech Technol. 2017, 20, 829–837.

[4] Stoyanov, B., Ivanova, T. (2021). Novel implementation of audio encryption using pseudorandom byte generator. applied sciences, 11(21), 10190.

[5] J. Liu, D. Song, Y. Xu, A parallel encryption algorithm for dual-core processor based on chaotic map, in: S. S. Mahmoud, Z. Zeng, Y. Li (Eds.), Fourth International Conference on Machine Vision (ICMV 2011): Computer Vision and Image Analysis; Pattern Recognition and Basic Technologies, volume 8350, International Society for Optics and Photonics, SPIE, 2012, pp. 73 – 79. doi:10.1117/12.920226.

[6] W. Wang, X. Wang, D. Song, A parallel chaotic cryptosystem for dual-core processor, in: The 2nd International Conference on Information Science and Engineering, 2010, pp.920–923. doi:10.1109/ICISE.2010.5689747.

[7] J. Liu, H. Zhang, D. Song, G. Sun, W. Bi, M. K. Buza, A parallel encryption algorithm of the logistic map for multicore with openmp, in: Ifost, volume 2, 2013, pp. 47–50. doi:10.1109/IFOST.2013.6616857.

[8] D. Burak, Parallelization of the block encryption algorithm based on logistic map, Przegląd Elektrotechniczny 88 (2012) 198–200.

[9] M. J. Rostami, A. Shahba, S. Saryazdi, H. Nezamabadi-pour, A novel parallel image encryption with chaotic windows based on logistic map, Computers & Electrical Engineering 62 (2017) 384–400. doi:10.1016/j.compeleceng.2017.04.004.

[10] Stoyanov, B., & Dobrev, D. (2022). Parallel Data Encryption Based on Tinkerbell and Ikeda Chaotic Functions. CEUR-WS 3191, 36-44.

[11] K. Ikeda, Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system, Optics Communications 30 (1979) 257–261. doi:10.1016/0030-4018(79)90090-7.

[12] K. Ikeda, H. Daido, O. Akimoto, Optical turbulence: Chaotic behavior of transmitted light from a ring cavity, Physical Review Letters 45 (1980) 709–712. doi:10.1103/PhysRevLett.45.709.

[13] K. T. Alligood, T. D. Sauer, J. A. Yorke, D. Chillingworth, Chaos: an introduction to dynamical systems, Springer, New York, 1996.

[14] IEEE Computer Society. IEEE Standard for Binary Floating-Point Arithmetic. ANSI/IEEE Std. 754; IEEE: New York, NY, USA, 1985.

[15] Jozwiak, M.; Monnet, X.; Teboul, J.L. Pressure waveform analysis. Anesth. Analg. 2018, 126, 1930–1933.

[16] Taylor, R. Interpretation of the correlation coefficient: a basic review. J. Diagnost. Med. Sonogr. 1990, 6, 35–39.

[17] Johnson, D.H. Signal-to-noise ratio. Scholarpedia 2006, 1, 2088.

[18] Korhonen, J.; You, J. Peak signal-to-noise ratio revisited: Is simple beautiful? In Proceedings of the 2012 Fourth International Workshop on Quality of Multimedia Experience, Melbourne, Australia, 5–7 July 2012; pp. 37–38

[19] Sathiyamurthi, P.; Ramakrishnan, S. Speech encryption using chaotic shift keying for secured speech communication. EURASIP J. Audio Speech Music Process. 2017, 2017, 1–11.

[20] Kordov, K. A Novel Audio Encryption Algorithm with Permutation-Substitution Architecture. Electronics 2019, 8, 530.

[21] Farsana, F.; Devi, V.; Gopakumar, K. An audio encryption scheme based on Fast Walsh Hadamard Transform and mixed chaotic keystreams. Appl. Comput. Inform. 2019.

Copyright © 2024 Dimitar Dobrev and Tsvetelina Ivanova. This open access article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.