A technique for securing digital audio files based on rotation and XOR operations

Anand Ballabh Joshi (anandiitd.joshi@gmail.com)
University of Lucknow https://orcid.org/0000-0003-0227-4032
Abdul Gaffar
University of Lucknow

Research Article

Keywords: Digital audio files, Rotation-XOR (RX) operations, Audio encryption, decryption

Posted Date: May 22nd, 2023

DOI: https://doi.org/10.21203/rs.3.rs-1749435/v1

Version of Record: A version of this preprint was published at Soft Computing on October 31st, 2023. See the published version at https://doi.org/10.1007/s00500-023-09349-5.

A technique for securing digital audio files based on rotation and XOR operations

Anand B. Joshi[0000−0003−0227−4032][1*†] and Abdul Gaffar[0000−0002−3388−9067†]

Department of Mathematics and Astronomy, University of Lucknow, University Road, Lucknow – 226 007, UP, India.

*Corresponding author(s). E-mail(s): anandiitd.joshi@gmail.com;
Contributing authors: abdulgaffar.lu@gmail.com;
†These authors contributed equally to this work.

Abstract

In this paper, we propose a WORD-oriented technique for encrypting (and decrypting) digital audio files based on rotation and XOR operations. The key-concepts of the designed encryption algorithm are the RX (Rotation-XOR) operations, i.e., the plain audio samples are first left-rotated by the sum-of-digits of the previous audio samples, and then XOR-ed with the previous audio samples. The designed encryption algorithm encodes a digital audio file into a random (noise) like audio file, from human visual as well as statistical points of view. Several encryption and decryption evaluation metrics, such as spectrogram, adjacent sample correlation coefficient, number of sample change rate, peak signal-to-noise ratio, etc., are applied on several digital audio files of varying sizes in order to empirically assess the performance and efficiency of the proposed technique. The results of these metrics validate the robustness of the designed technique.

Keywords: Digital audio files, Rotation-XOR (RX) operations, Audio encryption, decryption

1 Introduction

Everyday millions (perhaps billions) of messages in the form of texts, audio, images, and videos, are communicated on the Internet, which is an open (unsecure) network. So, there must be robust technique(s) in order to communicate secretly. In the context of secure communication, encryption is the best choice, which encodes a secret message into a form which is unrecognizable, except by the intended one. Broadly, there are two types of encryption schemes: symmetric-key encryption and asymmetric-key encryption. The symmetric-key encryption, also known as (a.k.a.) *private-key* encryption, uses the same secret key for encoding and decoding a message. The foremost application of the private-key encryption is to provide *confidentiality*. On the other hand, asymmetric-key encryption, a.k.a. *public-key* encryption, uses different keys for encoding and decoding a message. In particular, public-key is used for encoding, while private (secret) key is used for decoding a message. The foremost applications of the public-key encryption are authentication and non-repudiation, besides confidentiality.

Since symmetric-key encryption methods are much faster and efficient for attaining confidentiality as compared to asymmetric-key encryption methods, therefore, we adopt symmetric-key encryption method in the proposed technique. Note that, the Rotation-XOR (RX) operations utilized in the proposed technique are primitive operations, which are efficiently and directly supported by the most of the computer processors. These operations aid in the possible improvement of speed of the designed technique.

The rest of the paper has been put in the following order: Sec. 2 provides related works; Sec. 3 gives preliminaries; Sec. 4 describes the encryption and decryption algorithms of the proposed technique; Sec. 5 describes the implementation and experimental results; Sec. 6 discusses security analyses of the proposed technique; Sec. 7 gives comparison of the proposed technique with the recent state-of-the-art techniques; and Sec. 8 concludes the paper, followed by the references.

2 Related works

Abouelkheir and El-Sherbiny [1] in 2022 proposed a technique for the security of digital audio files based on a modified RSA (Rivest, Shamir, and Adleman) algorithm. The authors modified the RSA algorithm via using dynamic keys—for enhancing security of the proposed technique, and five numbers (two primes and three random numbers)—for enhancing speed of the proposed technique. Several metrics have been utilized in order to validate the aims of the designed scheme. Although the scheme performs well in terms of encryption, but in terms of decryption, it is not a good scheme. It performs lossy decryption, i.e., the decrypted audio files are not exactly identical to the original audio files.

Shah et al. [2] in 2021 proposed a technique for the secure communication of digital audio files based on finite fields. The authors generated a sequence of pseudo-random numbers via an elliptic curve, which is used to scramble the

samples of the plain audio files. Further, the scrambled audio samples are substituted via the newly constructed S-boxes, to ensure the confusion-diffusion properties [3] required for a secure encryption algorithm. Faragallah and El-Sayed [4] in 2021 proposed an encryption scheme for securing the audio files based on XOR (eXclusive OR) operation and Hartley Transform (HT). First of all, plain audio file is reshaped into a two-dimensional (2D) data blocks, and then it is XOR-ed with a gray scale image (treated as a secret key). The obtained XOR-ed blocks are then transposed via a chaotic map, followed by optical encryption using HT. Naskar et al. [5] in 2021 suggested an encryption scheme for audio files based on the distinct key blocks together with the Piece-Wise Linear Chaotic Map (PWLCM) and Elementary Cellular Automata (ECA). The scheme encrypts a plain audio file in three stages: cyclic shift, substitution, and scrambling. The cyclic shifting is utilized for reducing the correlation between the samples of each audio block. The shifted audio data-blocks are substituted (modified) via PWLCM , and finally, modified blocks are scrambled via ECA for better diffusion. Shah et al. [6] in 2021 proposed a method for encrypting digital audio files based on a 3D chaotic map. This map is used for substituting as well as permuting the samples of the audio files. Stoyanov and Ivanova [7] in 2021 designed an algorithm for securing audio files using an Ikeda map (a chaotic map). The map is utilized to generate pseudo-random bytes, which are XOR-ed with the samples of the plain audio files, producing the encrypted audio files. Aziz et al. [8] in 2021 proposed an audio encryption algorithm based on PSN (Permutation-Substitution Network) [3]. The permutation is performed via the application of Mordell elliptic curves, while substitution is performed via a symmetric group on eight symbols, i.e., S_8. The authors also utilized a chaotic map to further enhance the security of the audio files.

Abdelfatah [9] in 2020 proposed an algorithm for securing audio files in three phases utilizing three secret keys. First phase is the self-adaptive scrambling of the plain audio files via the first secret key. Second phase is the dynamic DNA (DeoxyriboNucleic Acid) encoding of the scrambled audio-data via the second secret key. The last phase is the cipher feedback mode via the third secret key, which aids in achieving better confusion and diffusion properties. Al-Kateeb and Moha [10] in 2020 proposed an audio encryption algorithm based on Discrete Wavelet Transform (DWT) and hand geometry. Hand geometry is utilized for fetching biometric information, to be used in the encryption algorithm.

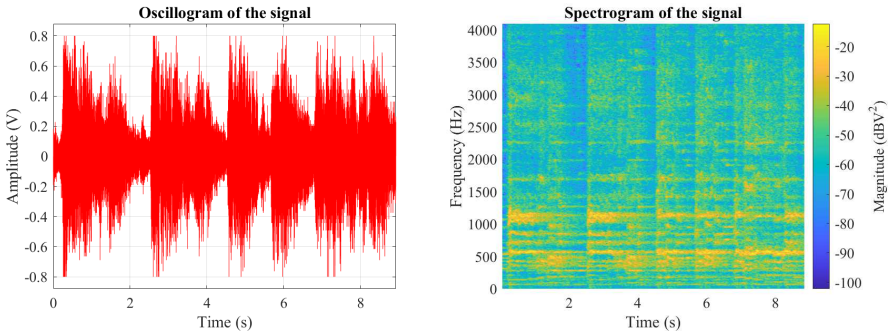
Wang and Su [11] in 2019 proposed an audio encryption approach using a PWLCM and DNA encoding, in order to attain the required confusion and diffusion properties. Kordov [12] in 2019 designed a scheme for the security of audio files based on the PSN using a chaotic circle map and modified rotation equations. Shah et al. [13] in 2019 suggested an audio encryption scheme based on PSN, wherein permutation is performed via the Henon map (chaotic map), while substitution is performed via the Mobius transformation.

Sasikaladevi et al. [14] in 2018 proposed an encryption scheme for encrypting audio files based on DWT and elliptic curves encryption. Sathiyamurthi and Ramkrishnan [15] in 2017 designed an encryption algorithm for encrypting audio files based on four chaotic maps: logistic map, tent map, quadratic map, and Bernoulli's map. Lima and Neto [16] in 2016 presented an approach for enciphering the digital audio files based on cosine number transform over a finite field.

3 Preliminaries

3.1 Digital audio

A digital audio, say, P is a l-by-c matrix, consisting of elements called *samples*, where l and c denote the number of samples and the number of channels in P, respectively. If $c = 1$, then P is said to be a *single* (or mono) channel audio file, and if $c = 2$, then P is said to be a *dual* (or sterio) channel audio file. Note that, the samples in P are floating-point values, i.e., real values. Figure 1 shows the oscillogram (a graph between amplitude and time) and spectrogram (a graph between frequency and time) of the audio file 'handel.wav', which is of size 73113×1, i.e., a single-channel audio file containing 73113 samples. For other details of the audio file 'handel.wav', namely, sample rate (in Hz—Hertz), duration (in sec—seconds), bits per sample, bit rate (in kbps—1000 bits per second), and size (in KB—1024 Bytes), see Table 1.



(a) Oscillogram (b) Spectrogram

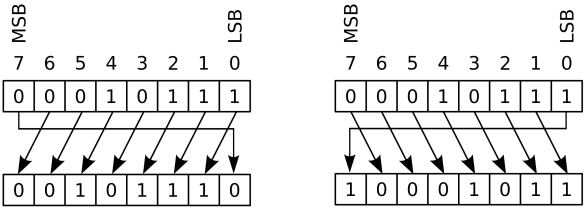
Fig. 1: Oscillogram and spectrogram of the audio file 'handel.wav'.

3.2 Rotation operation

By rotation operation, we mean "circular shift" or "bit-wise" rotation. It is of two types:

1. **Left rotation.** It is denoted by '$\lll$'. By $x \lll y$, it is meant that x is *left* rotated by y bits. For example, if $x = 0001\ 0111$ and $y = 1$, then $x \lll y$ gives 0010 1110. Fig. 2a demonstrates the concept, wherein MSB is the Most Significant Bit and LSB is the Least Significant Bit.

2. **Right rotation.** It is denoted by '$\ggg$'. By $x \ggg y$, it is meant that x is *right* rotated by y bits. For example, if $x = 0001\ 0111$ and $y = 1$, then $x \ggg y$ gives 1000 1011. Fig. 2b demonstrates the concept.



(a) Left rotation (b) Right rotation

Fig. 2: (a) Left rotation of $x = 0001\ 0111$ by 1-bit and (b) right rotation of x by 1-bit.

3.3 XOR operation

It is one of the simplest operations in a computer's processor. It is a bit-wise operation that takes two strings of bits of equal length, and performs XOR (denoted by $\oplus$) operation as: if two bits are same, the result is 0; and if not same, the result is 1. Its actually addition modulo 2.

For example, if $a = 1010\ 1011$ and $b = 0101\ 1100$, then $a \oplus b = 1111\ 0111$.

4 Description of the proposed encryption and decryption algorithms

4.1 Preprocessing on the audio file

Input. An audio file P of size $l \times 1$.

1. Convert the audio samples of P from floating point values (real values) to binary (matrix) via single-precision floating point (32-bit)[1].
2. Convert the binary (matrix) to non-negative integers (bytes) array, i.e., P is of size $1 \times l$. Note that, here samples of P are in bytes (0 to $2^8 - 1$).
3. Now, if l is a multiple of 4, then no padding is required, else pad $(4 - r)$ elements 'post' with zeros to P, where r is a remainder on dividing l by 4.

[1] See [17, 18]

4. Convert the bytes of P into WORDS, where WORD is a collection of 4 bytes, and rename the audio file P as P_w.

Output. The audio file P_w of size $1 \times m$, where m denotes number of WORDS in P_w.

4.2 Reverse preprocessing on the audio file

Input. The audio file P_w of size $1 \times m$, where m being number of WORDS in P_w.

1. Convert the WORDS of the audio file P_w into bytes (0 to $2^8 - 1$), and now, the size of P_w is $1 \times 4m$. Rename P_w as P.
2. Remove 'last' zero (padded) bytes, if any, from P, and let the size of P becomes $1 \times l$ bytes.
3. Convert the bytes (non-negative integers—0 to $2^8 - 1$) into binary (matrix).
4. Convert the binary (matrix) into floating-point values via single-precision floating point (32-bit).
5. Take transpose of P, so that the size of P becomes $l \times 1$.

Output. The audio file P of size $l \times 1$.

4.3 Preprocessing on secret key

Input. Secret key $K = \{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}, k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}, k_{17}, k_{18}, k_{19}, k_{20}, k_{21}, k_{22}, k_{23}, k_{24}, k_{25}, k_{26}, k_{27}, k_{28}, k_{29}, k_{30}, k_{31}, k_{32}\}$ of 32 bytes.

1. Split the secret key K into two equal parts, say, K_1 and K_2 as: $K_1 = \{k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}, k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}\}$ and $K_2 = \{k_{17}, k_{18}, k_{19}, k_{20}, k_{21}, k_{22}, k_{23}, k_{24}, k_{25}, k_{26}, k_{27}, k_{28}, k_{29}, k_{30}, k_{31}, k_{32}\}$.
2. Convert the key-bytes of K_1 and K_2 into WORDS as: $K_{1w} = \{q_{1w}, q_{2w}, q_{3w}, q_{4w}\}$ and $K_{2w} = \{r_{1w}, r_{2w}, r_{3w}, r_{4w}\}$, where $q_{1w} = k_1k_2k_3k_4$, $q_{2w} = k_5k_6k_7k_8$, $q_{3w} = k_9k_{10}k_{11}k_{12}$, and $q_{4w} = k_{13}k_{14}k_{15}k_{16}\}$; $r_{1w} = k_{17}k_{18}k_{19}k_{20}$, $r_{2w} = k_{21}k_{22}k_{23}k_{24}$, $r_{3w} = k_{25}k_{26}k_{27}k_{28}$, and $r_{4w} = k_{29}k_{30}k_{31}k_{32}\}$.
3. ***Expansion of*** K_{1w}**.**

▶ Expand K_{1w} to the size m as:

(a) For $i = 1, 2, 3, 4$; $T_1[i] = K_{1w}[i]$, i.e., $T_1[1] = q_{1w}$, $T_1[2] = q_{2w}$, $T_1[3] = q_{3w}$, and $T_1[4] = q_{4w}$.

(b) Calculate $T_1[5]$ as:

$$T_1[5] = \text{mod}(\lceil mean(T_1[i]) \rceil, 2^{32}), \qquad i = 1, 2, 3, 4.$$

where '*mean*' denotes the average function and 'mod' denotes the modulus function.

(c) Calculate $T_1[i]$, for $i = 6, 7, \ldots, m$, as:

$$T_1[i] = \text{mod}(T_1[i-1] + T_1[i-2], 2^{32}), \qquad i = 6, 7, \ldots, m.$$

4. ***Expansion of*** K_{2w}**.**

   ▶ Expand K_{2w} to the size m as:
   (a) For $i = 1, 2, 3, 4$; $T_2[i] = K_{2w}[i]$, i.e., $T_2[1] = r_{1w}$, $T_2[2] = r_{2w}$, $T_2[3] = r_{3w}$, and $T_2[4] = r_{4w}$.
   (b) Calculate $T_2[5]$ as:

$$T_2[5] = \text{mod}(\lceil mean(T_2[i]) \rceil, 2^{32}), \qquad i = 1, 2, 3, 4.$$

   where symbols have their usual meanings.
   (c) Calculate $T_2[i]$, for $i = 6, 7, \ldots, m$, as:

$$T_2[i] = \text{mod}(T_2[i-1] + T_2[i-2], 2^{32}), \qquad i = 6, 7, \ldots, m.$$

5. ***Generation of a third key.***

   ▶ Generate a third key K_{3w} from K_{1w} and K_{2w} as:

$$K_{3w} = \text{mod}(K_{1w} \cdot K_{2w}, 2^{32})$$

   where '$\cdot$' denotes component-wise multiplication.

Output. The expanded keys T_1 and T_2 of size m, and the generated key K_{3w} of size 4.

4.4 Encryption algorithm

Input. An audio file P of size $l \times 1$ and the secret key K of 32-byte.

1. Apply preprocessing on the audio file P (see Sec. 4.1), and let the obtained file be P_w of size $1 \times m$.
2. Apply preprocessing on secret key K (see Sec. 4.3) to obtain the expanded keys T_1 & T_2 of size m, and the generated key K_{3w} of size 4 (in WORDS).
3. ***Initial round substitution.*** XOR P_w with T_1, i.e.,

$$B[i] = P_w[i] \oplus T_1[i], \qquad i = 1, 2, \ldots, m.$$

4. ***First round substitution.***
   (a) Let $B = \{b_1, b_2, \ldots, b_m\}$, then do the following:

$$\begin{aligned} &\text{for } i = 1 \text{ to } m \\ &\quad b_{i-1} = c_{i-1} \\ &\quad c_i = [b_i \lll \sigma(b_{i-1})] \oplus b_{i-1} \\ &\text{end for} \end{aligned}$$

   where $c_0 = b_m$; 'σ' in $\sigma(b_{i-1})$ denotes sum-of-digits function, and $\sigma(b_{i-1})$ denotes sum-of-digits of b_{i-1}; and '$\lll$' denotes left rotation operator.

(b) Let $C = \{c_1, c_2, \ldots, c_m\}$, then do the following:

$$C[i] = C[i] \oplus K_{3w}[i], \qquad i = 1, 2, 3, \text{ and}$$

$$C[m] = C[m] \oplus K_{3w}[4].$$

5. ***Second round substitution.***
   (a) Do the following:

$$\begin{array}{l} \text{for } j = 1 \text{ to } m \\ \quad c_{j-1} = d_{j-1} \\ \quad d_j = [c_j \lll \sigma(c_{j-1})] \oplus c_{j-1} \\ \text{end for} \end{array}$$

   where $d_0 = c_m$, and rest symbols have their usual meanings.
   (b) Let $D = \{d_1, d_2, \ldots, d_m\}$, then do the following:

$$E[j] = D[j] \oplus T_2[j], \qquad j = 1, 2, \ldots, m.$$

6. Apply reverse preprocessing on the audio file E of size $1 \times m$ (see Sec. 4.2), and let the obtained audio file be F of size $l \times 1$.

Output. The encrypted audio file F of size $l \times 1$.

4.5 Decryption algorithm

Input. The encrypted audio file F of size $l \times 1$ and the secret key K (32-byte).

1. Apply the preprocessing on the audio file F (see Sec. 4.1) to obtain an audio file E of size $1 \times m$, m being number of WORDS in E.
2. ***Second round substitution.***
   (a) XOR the audio file E with T_2, i.e.:

$$D[j] = E[j] \oplus T_2[j], \qquad j = 1, 2, \ldots, m.$$

   (b) Let $D = \{d_1, d_2, \ldots, d_m\}$, then do the following:

$$\begin{array}{l} \text{for } j = m \text{ to } 1 \\ \quad c_j = [d_j \oplus d_{j-1}] \ggg \sigma(d_{i-1}) \\ \text{end for} \end{array}$$

   where '$j = m$ to 1' means $j = m, m-1, \ldots, 2, 1$; $d_0 = d_m$; and '$\ggg$' denotes right rotation.
3. ***First round substitution.***

(a) Let $C = \{c_1, c_2, \ldots, c_m\}$, then do the following:

$$C[i] = C[i] \oplus K_{3w}[i], \qquad i = 1, 2, 3, \text{ and}$$

$$C[m] = C[m] \oplus K_{3w}[4].$$

(b) Do the following:

$$\begin{aligned} &\text{for } i = m \text{ to } 1 \\ &\quad b_i = [c_i \oplus c_{i-1}] \ggg \sigma(c_{i-1}) \\ &\text{end for} \end{aligned}$$

where $c_0 = C_m$, and rest symbols have their usual meanings.

4. ***Initial round substitution.*** Let $B = \{b_1, b_2, \ldots, b_m\}$, then do the following:
$$P_w[i] = B[i] \oplus T_1[i], \qquad i = 1, 2, \ldots, m.$$
5. Apply the reverse preprocessing on the audio file P_w (see Sec. 4.2) of size $1 \times m$, to obtain the audio file P of size $l \times 1$.

Output. The decrypted (original) audio file P of size $l \times 1$.

5 Implementation and experimental results

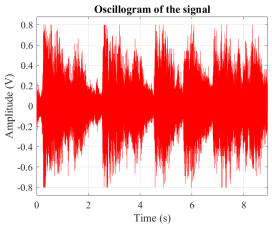
The proposed technique is implemented on *MATLAB (R2021a)* software under the Windows 10 operating system. To evaluate the performance (encryption and decryption qualities) of the proposed technique, a number of test audio files of different sample lengths are taken from the MATLAB IPT (Image Processing Toolbox)[2], except the audio file 'zeros.wav', which is created in MATLAB software. The details of these audio files are provided in Table 1. Also, the oscillograms (Osc. for oscillogram—in short) of the original, encrypted, and decrypted audio files are shown in Fig. 3.

Table 1: Description of the test audio files

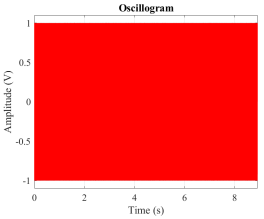
File Name (.wav)	Channels	Sample Rate (in Hz)	Total Samples (length)	Duration (in sec)	Bits/Sample	Bit rate (in kbps)	Size (in KB)
handel	1	8192	73113	8.9249	16	131.0720	142.7988
gong	1	8192	42028	5.1304	16	131.0720	82.0859
zeros	1	8192	19120	2.3340	16	131.0720	37.3438
splat	1	8192	10001	1.2208	16	131.0720	25.1562

From Fig. 3, we observe that the oscillograms of the encrypted audio files are uniform, unlike to those of the corresponding original audio files. Also, the oscillograms of the decrypted audio files are identical to those of the corresponding original files. Thus, our proposed technique performs a robust encryption. Also, since the audio files are successfully decrypted without any data-loss, so the designed technique performs lossless decryption.

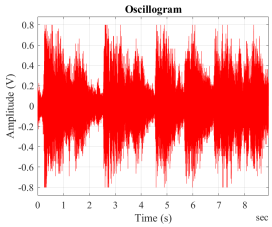
[2]Available in, C:\Program Files\Polyspace\R2021a\toolbox\images\imdata.



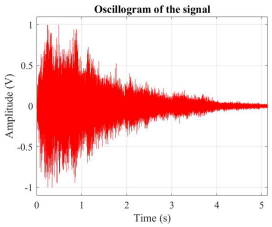
(a) Osc. of original handel file.



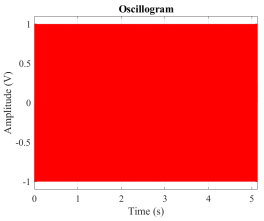
(b) Osc. of encrypted handel file.



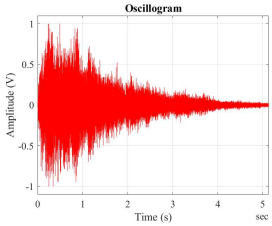
(c) Osc. of decrypted handel file.



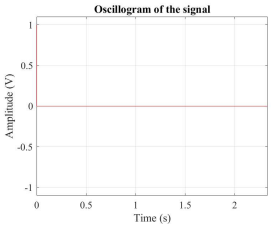
(d) Osc. of original gong file.



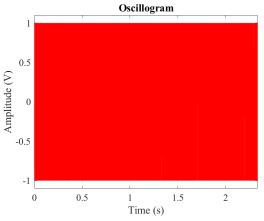
(e) Osc. of encrypted gong file.



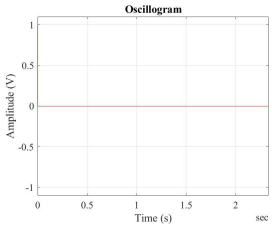
(f) Osc. of the decrypted gong file.



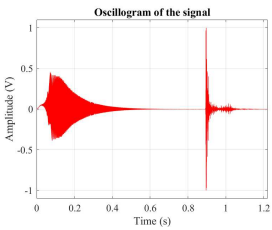
(g) Osc. of original zeros file.



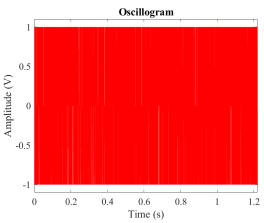
(h) Osc. of encrypted zeros file.



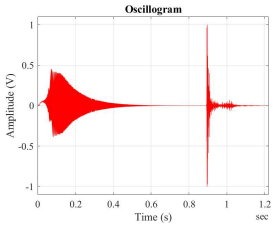
(i) Osc. of decrypted zeros file.



(j) Osc. of original splat file.



(k) Osc. of encrypted splat file.



(l) Osc. of decrypted splat file.

Fig. 3: Experimental results: Figs. (a), (d), (g), and (j) show the oscillograms of original audio files; Figs. (b), (e), (h), and (k) show the oscillograms of the corresponding encrypted audio files; and Figs. (c), (f), (i), and (l) show the oscillograms of the corresponding decrypted audio files.

6 Security analyses

6.1 Key space analysis

The space of all potential combinations of a key constitutes a key space of any encryption/decryption algorithm. Key space should be very large so that attacks, such as brute-force [19], known/chosen plaintext [20], etc., could become unsuccessful. Our proposed technique is based on a secret key of 32 bytes (256 bits), which produces a key space of 2^{256}, and as of today, it is believed to be unbreakable. We also compare our key space with the key space of the existing methods. The results are provided in the Table 2, whence we infer that our proposed technique has very large key space as compared to the existing methods.

Table 2: Comparison of the key space

Our Method	[15]	[21]	[23]	[22]
2^{256}	2^{149}	2^{144}	2^{128}	2^{128}

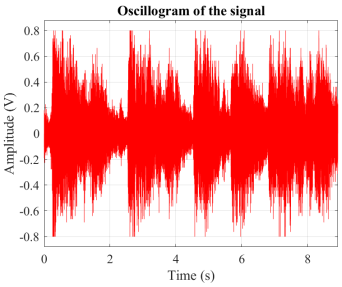
6.2 Key sensitivity analysis

This test is utilized to judge the confusion property [3] of any encryption/decryption algorithm. According to Shannon [3], a secure cryptographic algorithm must have the confusion property to thwart statistical attacks. It is the property of confusion that hides the relationship between the encrypted data and the secret key. The key sensitivity test is utilized to judge this confusion property. The sensitivity of the secret key is assessed in two aspects:

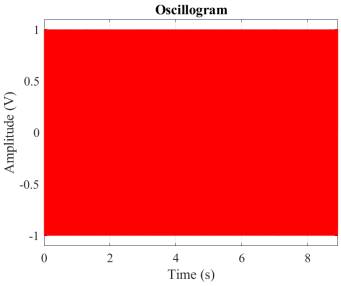
1. **Encryption.** It is used to measure the dissimilarity between the two encrypted audio files E_1 and E_2 with respect to (w.r.t.) the same plain audio file P using two different encryption keys λ_1 and λ_2, where λ_1 and λ_2 are obtained from the original secret key K by altering merely LSB corresponding to the last and the first byte of K, respectively.

2. **Decryption.** It is used to measure the dissimilarity between the two decrypted audio files D_1 and D_2 w.r.t. the same encrypted audio file E, encrypted via secret key K, using the decryption keys λ_1 and λ_2, respectively. Note that, both encryption/decryption keys λ_1 and λ_2 differ from each other as well as from the secret key K merely by 1-bit.

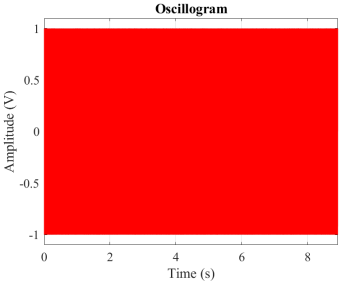
The results of key sensitivity analysis w.r.t. the encryption (*enc*—in short) and decryption (*dec*—in short) aspects are shown in Figs. 4 and 5, respectively, whence we infer that the proposed technique has very high bit-level sensitivity, and thus, ensures the property of confusion.



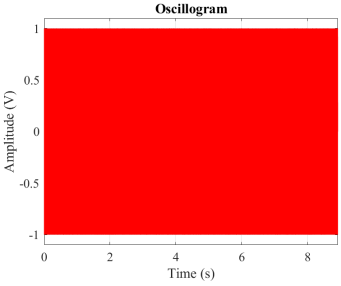
(a) Oscillogram of the original handel file (P).



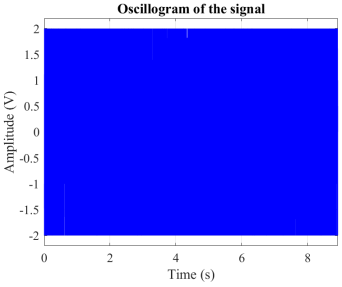
(b) Oscillogram of the encrypted handel file E, where $E = enc(P, K)$.



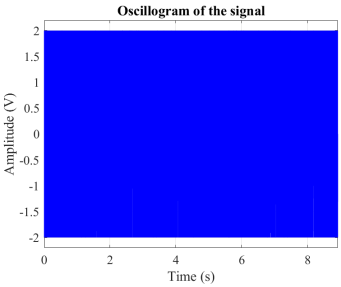
(c) Oscillogram of E_1, where $E_1 = enc(P, \lambda_1)$.



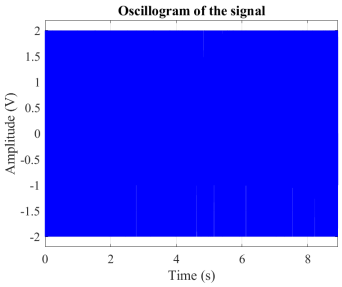
(d) Oscillogram of E_2, where $E_2 = enc(P, \lambda_2)$.



(e) Oscillogram of $|E_1 - E|$, where $|E_1 - E|$ is the absolute difference between E_1 and E.

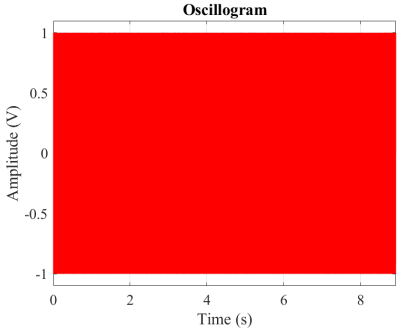


(f) Oscillogram of $|E_2 - E|$.

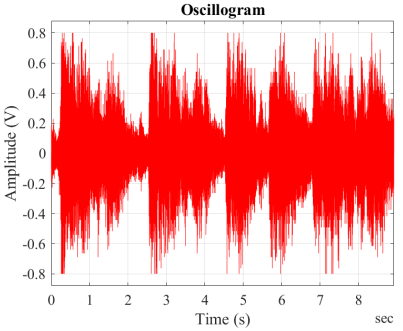


(g) Oscillogram of $|E_2 - E_1|$.

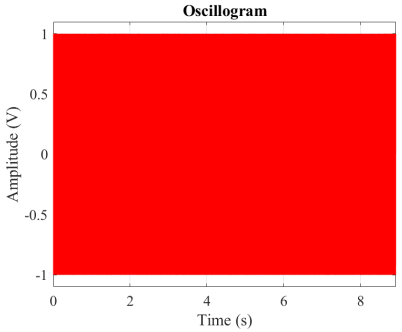
Fig. 4: Key sensitivity analysis w.r.t. encryption.



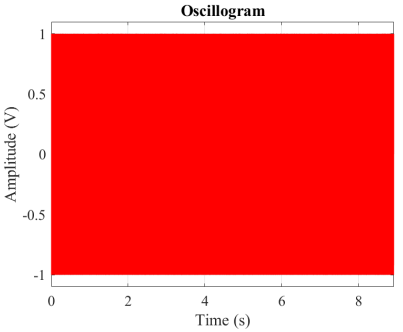
(a) Oscillogram of the encrypted handel file E, where $E = enc(P, K)$.



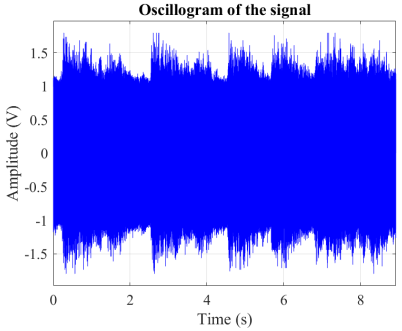
(b) Oscillogram of the decrypted handel file (D) using correct secret key K.



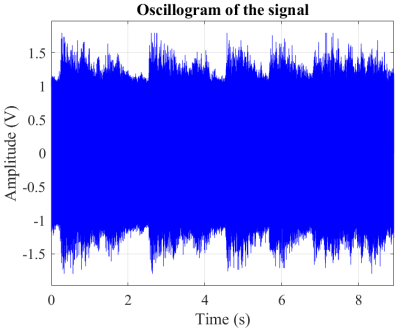
(c) Oscillogram of D_1, where $D_1 = dec(E, \lambda_1)$.



(d) Oscillogram of D_2, where $D_2 = dec(E, \lambda_2)$.



(e) Oscillogram of $|D_1 - D|$.



(f) Oscillogram of $|D_2 - D|$.

Fig. 5: Key sensitivity analysis w.r.t. decryption.

6.3 Encryption evaluation metrics

Since any single metric can not evaluate any encryption algorithm (or any encrypted audio file) fully, so we utilize several metrics, namely, spectrogram, adjacent sample correlation coefficient, signal-to-noise ratio, root mean square, crest factor, and number of sample change rate.

6.3.1 Spectrogram analysis

The spectrogram [24] is a graph of an audio file between frequency and time. X-axis represents time in seconds, Y-axis represents frequency in Hertz, and the co-ordinate values represent energy values. The spectrograms of the original and the corresponding encrypted audio files are shown in Fig. 6. From Fig. 6, we observe that the spectrograms of the encrypted audio files have uniform darker color (yellow color), i.e., have stronger energy, unlike to those of original audio files, which have (non-uniform) lighter color (mostly non-yellow), i.e., have weaker energy. Thus, the encrypted audio files are random-like audio files, which do not provide any relevant information regarding the original audio files.

6.3.2 Adjacent Sample Correlation Coefficient (ASCC) analysis

ASCC test [25] is the frequently used measure to assess the concreteness of the novel techniques constructed for audio encryption, and in particular, to test the random distribution of samples in the encrypted audio file. Here, we have taken two thousand pairs of samples, which are chosen at random to estimate ASCC along vertical direction. Note that, ASCC along horizontal and diagonal directions can not be calculated since a single-channel audio file is merely a column vector, not a matrix.

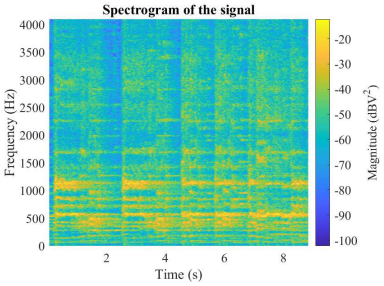
Let I be an audio file of size $l \times 1$. Then, the correlation coefficient of adjacent samples of I is given by Eq. 1:

$$\rho_{XY} = \frac{Cov(X, Y)}{\sqrt{Var(X) \cdot Var(Y)}}, \tag{1}$$

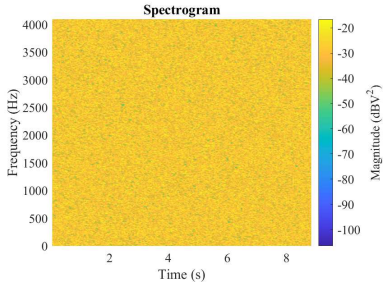
where $Cov(X, Y)$ denotes the covariance between the column vectors X and Y, while $Var(X)$ denotes the variance of column vector X. The X and Y are computed as follows:

$$\left.\begin{array}{l} X = I(1 : l-1,\ :) \\ Y = I(2 : l,\ :) \end{array}\right\}.$$

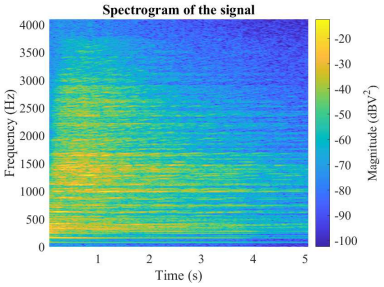
Since the neighboring samples in the original audio file are strongly correlated so, the value of correlation coefficient ρ_{XY} tends to 1, and in the case of an encrypted audio file, value of ρ_{XY} tends towards 0, cause the samples in the encrypted audio file are weakly correlated. The ASCC values of the plain and the cipher audio files along the vertical direction are shown in Table 3. For



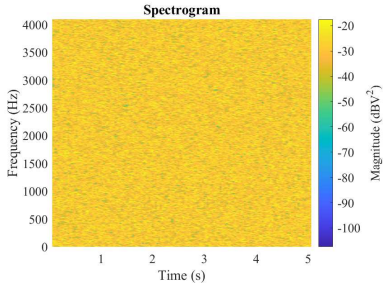
(a) Spectrogram of original handel file.



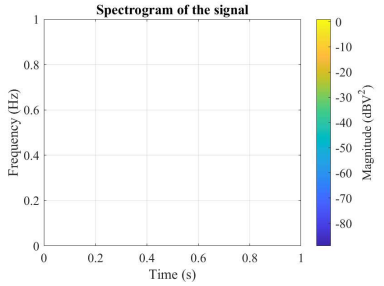
(b) Spectrogram of encrypted handel file.



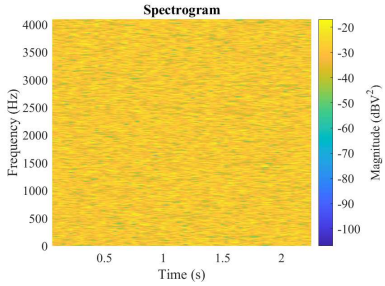
(c) Spectrogram of original gong file.



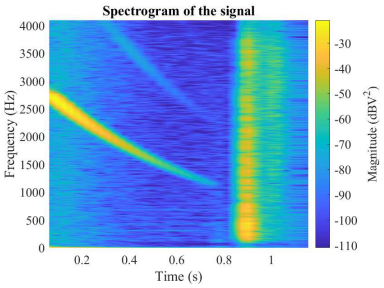
(d) Spectrogram of encrypted gong file.



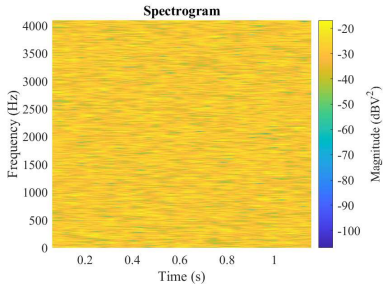
(e) Spectrogram of original zeros file.



(f) Spectrogram of encrypted zeros file.



(g) Spectrogram of original splat file.



(h) Spectrogram of encrypted splat file.

Fig. 6: Spectrograms of the original and the corresponding encrypted audio files.

Table 3: ASCC values of the plain and the cipher audio files

Method	File Name (.wav)	Duration (in sec)	Size (in KB)	ASCC	
				Plain	*Cipher*
Proposed	handel	8.9249	142.7988	0.7554	− 0.0065
	gong	5.1304	82.0859	0.6413	0.0067
	zeros	2.3340	37.3438	NaN	− 0.1041
	splat	1.2208	25.1562	− 0.0958	0.0075
Shah (2021) [2]	Music sound	—	—	0.9847	− 0.0081
Faragallah (2021) [4]	Alarm	—	—	—	—
Stoyanov (2021) [7]	usb-headset-weird	2.3200	24.4141	—	—
Shah (2021) [6]	Female sound	—	31.25	0.9933	− 0.0019
Abdelfatah (2020) [9]	Audio-2	1.7600	296.875	—	− 0.0003

quick observation, the correlation graphs are also provided, which are shown in Fig. 7. From the Fig. 7, we infer that the correlations graphs of the encrypted audio files are uniform, unlike to those of original audio files.

We also compare the obtained ASCC values with the most recent methods, and the comparison is provided in the Table 3. Note that, symbol '—' in the Table 3 means "not available", i.e., the data is not available in the literature.

6.3.3 Signal-to-Noise Ratio (SNR) analysis

The SNR [26] is also a metric used to analyze an encrypted audio file. The more negative SNR implies better encryption quality. It is measured in decibel (dB) units. The SNR of an audio file, say, I can be calculated via Eq. 2:

$$SNR = \frac{\mu}{\psi} , \tag{2}$$

where μ (mean) and ψ (standard deviation) are given by Eqs. 3 and 4, respectively;

$$\mu = \frac{\sum_{j=1}^{l} u_j}{l} , \tag{3}$$

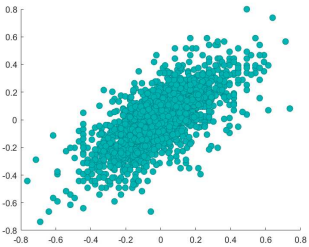
$$\psi = \sqrt{\frac{\sum_{j=1}^{l} (u_j - \mu)^2}{l}} , \tag{4}$$

where 'u_j' denotes the samples of the audio (plain/cipher) file I and 'l' denotes the number of samples in the audio file. The SNR values of the plain and the cipher audio files are provided in Table 4.

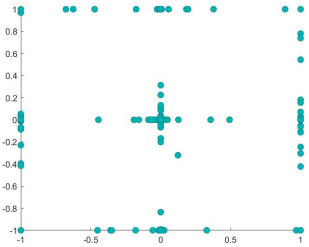
6.3.4 RMS (Root Mean Square) analysis

The RMS [29] is used to calculate the average amplitude value of any (plain/-cipher) audio file. For an original audio file, it should be close to zero, while for an encrypted audio file, it should be closed to one. It can be calculated using Eq. 5:

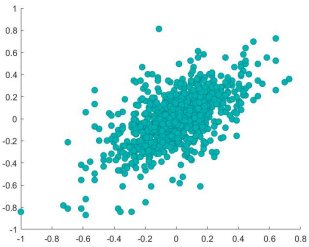
$$RMS = \sqrt{\frac{1}{l} \sum_{j=1}^{l} {u_j}^2} , \tag{5}$$



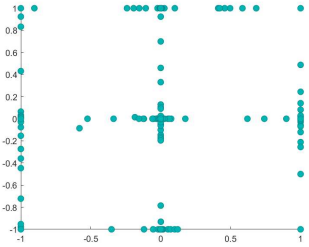
(a) ASCC graph of the original handel file.



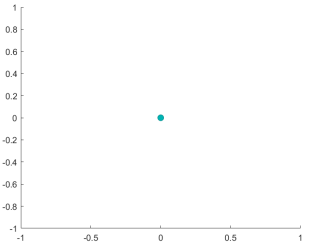
(b) ASCC graph of the encrypted handel file.



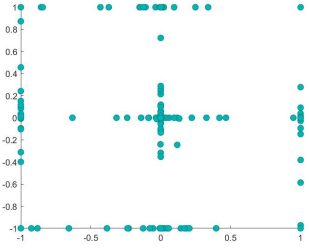
(c) ASCC graph of the original gong file.



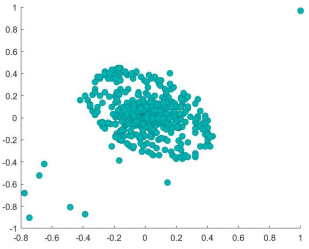
(d) ASCC graph of the encrypted gong file.



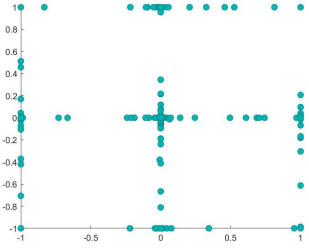
(e) ASCC graph of the original zeros file.



(f) ASCC graph of the encrypted zeros file.



(g) ASCC graph of the original splat file.



(h) ASCC graph of the encrypted splat file.

Fig. 7: ASCC graphs of the original and the encrypted audio files along vertical direction.

where symbols have their usual meanings.

The RMS values for the plain and the cipher audio files are provided in Table 4, whence we notice that the RMS values for the cipher audio files are closed to the ideal value.

6.3.5 Crest Factor (CF) analysis

The CF [30], a.k.a. peak-to-average ratio, is another metric to analyze an audio file. It is measured in dB units. For an encrypted audio file, the crest factor should be closed to 3 dB. It can be calculated using Eq. 6:

$$CF = \frac{u_p}{RMS} \ , \tag{6}$$

where u_p (peak value) is the maximum absolute value of an audio file, and RMS (average value) is the root mean square value, given by Eq. 5.

The CF values for the plain and the cipher audio files are provided in Table 4, whence we notice that the CF values for the cipher audio files are very closed to the ideal value (3 dB).

6.3.6 Number of Sample Change Rate (NSCR) test

The NSCR [31] is used to test the resistance of differential attack [32], or judging the Shannon's diffusion property [3]. The NSCR scores between the encrypted audio files E_1 and E_2 can be calculated via Eq. 7:

$$NSCR = \sum_{s=1}^{l} \frac{\beta(s,\,1)}{l} \times 100\% \ , \tag{7}$$

where $\beta(s,\,1)$ is given by Eq. 8:

$$\beta(s,\,1) = \begin{cases} 0, & \text{if } E_1(s,\,1) = E_2(s,\,1) \\ 1, & \text{if } E_1(s,\,1) \neq E_2(s,\,1) \end{cases} \tag{8}$$

where $E_1(s,\,1)$ and $E_2(s,\,1)$ are the samples of the encrypted audio files prior to and after alteration of only one-sample of the original audio file.

We have calculated the NSCR scores by changing only one sample of the test audio files at different positions (from beginning—$(1,\,1)^{\text{th}}$ sample as well as from the last—$(l,\,1)^{\text{th}}$ sample), l being the total number of samples in an audio file. The obtained NSCR scores are shown in the Table 5. Note that, if the calculated/reported NSCR score is greater than the theoretical NSCR value, which is 99.5527 at 0.01 significance level and 99.5693% at 0.05 level [31], then the NSCR test is *passed*. The proposed technique passes the NSCR test for all the audio files, and thus, ensures the property of diffusion, and also, outperforms the methods listed in the Table 5, which are vulnerable to the differential attack.

Table 4: SNR, RMS, and CF values of the plain and the cipher audio files

Method	File Name (.wav)	Duration (in sec)	Size (in KB)	SNR (dB)		RMS		CF (dB)	
				Plain	***Cipher***	***Plain***	***Cipher***	***Plain***	***Cipher***
Proposed	handel	8.9249	142.7988	− 16.1304	− 27.5990	0.1962	0.7089	12.2074	2.9892
	gong	5.1304	82.0859	− 12.3680	− 25.5598	0.1538	0.7087	16.2633	2.9904
	zeros	2.3340	37.3438	− 34.0016	− 23.9284	0.0072	0.7092	42.8149	2.9842
	splat	1.2208	25.1562	− 20.4148	− 21.2276	0.0959	0.7119	20.3654	2.9527
Shah (2021) [2]	Music sound	—	—	—	—	—	0.6423	—	4.7610
Naskar (2021) [5]	Audio-4	—	162	—	− 8.6030	—	—	—	—
Aziz (2021) [8]	Audio 1	—	—	—	—	—	0.0034	—	49.3
Abdelfatah (2020) [9]	Audio-2	1.7600	296.875	—	− 28.1400	—	0.6027	—	4.3973
Al-Kateeb (2020) [10]	A5	—	9.6436	—	− 18.2154	—	—	—	—
Kordov (2019) [12]	File 3	1.60	16.8457	—	− 8.7189	—	—	—	—
Naskar (2019) [27]	—	—	—	—	—	—	0.6000	—	4.8
Belmeguenai (2017) [28]	—	10.6300	—	—	—	—	—	—	
Sathiyamurthi (2017) [15]	Audio-2	8.0	—	—	32.5781	—	—	—	—

Table 5: NSCR scores of the encrypted images

Method	File Name (.wav)	Duration (in sec)	Size (in KB)	Position Altered	NSCR Score (in %)
Proposed	handel	8.9249	142.7988	(1, 1) (l, 1)	100 100
	gong	5.1304	82.0859	(2, 1) ($l-2$, 1)	100 100
	zeros	2.3340	37.3438	(1, 1) (l, 1)	100 100
	splat	1.2208	25.1562	(1, 1) (l, 1)	100 100
Shah (2021) [2]	Bells sound	—	—	—	99.9884
Faragallah (2021) [4]	Alarm	—	—	—	99.7500
Naskar (2021) [5]	Audio-4	—	162	—	99.9958
Shah (2021) [6]	Female sound	—	31.25	—	99.9958
Stoyanov (2021) [7]	usb-headset-weird	2.3200	24.4141	—	99.9940
Aziz (2021) [8]	Audio 1	—	—	—	99.5316
Abdelfatah (2020) [9]	Audio-2	1.7600	296.875	—	99.9700
Naskar (2019) [27]	—	—	—	—	99.9989
Shah (2019) [13]	Go ahead	—	139.6484	—	99.9973
Kordov (2019) [12]	File 3	1.60	16.8457	—	99.9972
Farsana (2019) [35]	—	—	—	—	99.9989
Sasikaladevi (2018) [14]	—	—	—	—	98.0000
Faragallah (2017) [34]	—	—	—	—	99.7649
Ghasemzadeh (2017) [21]	—	—	—	—	99.9978
Habib (2017) [36]	—	—	—	—	99.6521
Sathiyamurthi (2017) [15]	—	—	—	—	99.9996
Lima (2016) [16]	—	—	—	—	99.9992
Farsana (2016) [33]	—	—	—	—	99.3700
Liu (2016) [22]	—	—	—	—	99.9952

Table 6: MSE and PSNR values between the decrypted and the original audio files

Method	File Name (.wav)	Duration (in sec)	Size (in KB)	MSE	PSNR
Proposed	handel	8.9249	142.7988	0	∞
	gong	5.1304	82.0859	0	∞
	zeros	2.3340	37.3438	0	∞
	splat	1.2208	25.1562	0	∞
[1]	Sen_4	1.1901	41.0156	3.3161×10^{-11}	—

6.4 Decryption evaluation metrics

To evaluate the decryption algorithm, i.e., the decrypted audio files, we utilize two important metrics: mean square error and peak-signal-to-noise ratio.

6.4.1 Mean Square Error (MSE) analysis

The MSE [37] is used to judge the decryption quality of any decrypted audio file. MSE value can be any non-negative integer. Lower the MSE, better is the decryption quality, in particular, value 0 denotes perfect decryption, i.e., the original and the decrypted audio files are exactly identical—lossless decryption. The MSE can be calculated via Eq. 9:

$$MSE = \sum_{j=1}^{l} \frac{(P_j - D_j)}{l} \ , \tag{9}$$

where P_j and D_j denote the j^{th} samples of the original and the decrypted audio files, respectively, while other symbols have their usual meanings.

The values of MSE between the original and the decrypted audio files are provided in Table 6. From the table, we observe that the MSE values are 0 (zero), endorsing that the decrypted audio files are perfectly identical to the original audio files.

6.4.2 Peak-Signal-to-Noise Ratio (PSNR) analysis

The PSNR [38] metric is also used to measure the quality the decrypted audio file. PSNR can be any positive real number, and is measured in dB units. Higher the value of PSNR, better is the decryption quality. In particular, PSNR value equals to ∞ implies perfect decryption, i.e., lossless decryption. It can be calculated using Eq. 10:

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

where 'h' denotes largest possible value of an audio file and 'MSE' is defined by Eq. 9.

The values of PSNR between the original and the decrypted audio files are provided in Table 6. From the table, we observe that the PSNR values are equal to ∞, endorsing that the decrypted audio files are perfectly identical to the original audio files. In other words, the decryption algorithm performs lossless decryption.

7 Comparison with the existing techniques

The proposed technique is compared with the recent state-of-the-art techniques based on commonly available metrics, namely, key space, ASCC, SNR, RMS, CF, and NSCR. The comparison of the proposed approach with the recent approaches based on key space is provided in Table 2; based on ASSC is provided in the Table 3; based on SNR, RMS, and CF metrics is provided in the Table 4; and based on the metric NSCR is provided in the Table 5. From the Tables 2, 3, 4, and 5, we infer that our proposed technique performs well in terms of respective compared metrics.

8 Conclusion

In this paper, we proposed a technique for securing digital audio files based on the WORD-oriented RX operations. Several performance evaluation metrics, i.e., encryption and decryption evaluation metrics, have been utilized on a number of audio files of varying sizes from the standard database, in order to empirically assess the efficiency and robustness of the designed approach. The results of these performance evaluation metrics validate the goals of the proposed approach. Moreover, a thorough comparison with the recent state-of-the-art techniques, based on several metrics, have also been made.

Author contributions

Both the authors contributed equally to this manuscript.

Compliance with ethical standards

- **Ethical approval.** This manuscript does not contain any studies with human participants and/or animals.
- **Funding.** This work was partially supported by the UGC (University Grants Commission), India, under grant no. [415024].
- **Conflict of interest.** The authors declare that there is no conflict of interest regarding the publication of this manuscript.
- **Informed consent.** Informed consent was obtained from all individual participants included in the study.

References

[1] Abouelkheir E and Sherbiny SE (2022) Enhancement of speech encryption/decryption process using RSA algorithm variants. Human-centric Computing and Information Sciences 12(6). https://doi.org/10.22967/HCIS.2022.12.006.

[2] Shah D, Shah T, Hazzazi MM, Haider MI, Aljaedia, and Hussain I (2021) An efficient audio encryption scheme based on finite fields. IEEE Access 9:144385–144394. https://doi.org/10.1109/ACCESS.2021.3119515.

[3] Shannon CE (1949) Communication theory of secrecy systems. The Bell System Technical Journal 28(4):656–715. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x.

[4] Faragallah OS and El-Sayed HS (2021) Secure opto-audio cryptosystem using XOR-ing mask and Hartley transform. IEEE Access 9:25437–25449. https://doi.org/10.1109/ACCESS.2021.3055738.

[5] Naskar PK, Bhattacharyya S, and Chaudhuri A (2021) An audio encryption based on distinct key blocks along with PWLCM and ECA. Nonlinear Dyn 103:2019–2042. https://doi.org/10.1007/s11071-020-06164-7.

[6] Shah D, Shah T, Ahamad I., Haider MI, and Khalid I (2021) A three-dimensional chaotic map and their applications to digital audio security. Multimed Tools Appl 80:22251–22273. https://doi.org/10.1007/s11042-021-10697-3.

[7] Stoyanov B and Ivanova T (2021) Novel implementation of audio encryption using pseudorandom byte generator. Applied Sciences 11(21):10190. https://doi.org/10.3390/app112110190.

[8] Aziz H, Gilani SMM, Hussain I, Janjua AK, and Khurram S (2021) A noise-tolerant audio encryption framework designed by the application of S8 symmetric group and chaotic systems. Mathematical Problems in Engineering 2021:5554707. https://doi.org/10.1155/2021/5554707.

[9] Abdelfatah RI (2020) Audio encryption scheme using self-adaptive bit scrambling and two multi chaotic-based dynamic DNA computations. IEEE Access8:69894–69907. https://doi.org/10.1109/ACCESS.2020.2987197.

[10] Al-kateeb ZN and Mohammed SJ (2020) A novel approach for audio file encryption using hand geometry. Multimed Tools Appl 79:19615–19628. https://doi.org/10.1007/s11042-020-08869-8.

[11] Wang X and Su Y (2020) An audio encryption algorithm based on DNA coding and chaotic system. IEEE Access 8:9260–9270. https://doi.org/10.

1109/ACCESS.2019.2963329.

[12] Kordov K (2019) A novel audio encryption algorithm with permutation-substitution architecture. Electronics 8:530. https://doi.org/10.3390/electronics8050530.

[13] Shah D, Shah T, and Jamal SS (2020) Digital audio signals encryption by Mobius transformation and Hnon map. Multimedia Systems 26:235–245. https://doi.org/10.1007/s00530-019-00640-w.

[14] Sasikaladevi N, Geetha K, and Srinivas KNV (2018). A multi-tier security system (SAIL) for protecting audio signals from malicious exploits. Int J Speech Tech 21(2):319–332. https://doi.org/10.1007/s10772-018-9510-0.

[15] Sathiyamurthi P and Ramakrishnan S (2017). Speech encryption using chaotic shift keying for secured speech communication. J Audio Speech Music Proc. 20. https://doi.org/10.1186/s13636-017-0118-0.

[16] Lima JB and Neto EFS (2016) Audio encryption based on the cosine number transform. Multimedia Tools Appl. 75(14):8403–8418. https://doi.org/10.1007/s11042-015-2755-6.

[17] Available at, https://in.mathworks.com/help/matlab/matlab_prog/floating-point-numbers.html (Accessed Jun. 09, 2022).

[18] Available at, https://en.wikipedia.org/wiki/Single-precision_floating-point_format (Accessed Jun. 09, 2022).

[19] ECRYPT II yearly report on algorithms and keysizes, N. Smart (ed.) (BRIS), 2011–12. https://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf (Accessed Jun. 09, 2022).

[20] Stinson DR (2006) Cryptography: Theory and Practice, London: Chapman and Hall CRC.

[21] Ghasemzadeh A and Esmaeili E (2017) A novel method in audio message encryption based on a mixture of chaos function. Int. J. Speech Technol. 20(4):829–837. https://doi.org/10.1007/s10772-017-9452-y.

[22] Liu H, Kadir A, and Li Y (2016) Audio encryption scheme by confusion and diffusion based on multi-scroll chaotic system and one-time keys. Optik 127(19):7431–7438. https://doi.org/10.1016/j.ijleo.2016.05.073.

[23] Augustine N, George SN, and Pattathil DP (2015) An audio encryption technique through compressive sensing and Arnold transform. Int. J. Trust Manage. Comput. Commun. 3(1):74–92. https://doi.org/10.1504/IJTMCC.2015.072467.

[24] Available at, https://in.mathworks.com/help/signal/ref/spectrogram.html (Accessed Jun. 09, 2022).

[25] Fisher RA and Yates F (1958) Statistical Methods for Research Workers, 13th edn., New York, USA: Hafner.

[26] Available at, https://in.mathworks.com/help/signal/ref/snr.html (Accessed Jun. 09, 2022).

[27] Naskar PK, Paul S, Nandy D, and Chaudhuri A (2019) DNA encoding and channel shuffling for secured encryption of audio data. Multimedia Tools Appl. 78(17):25019–25042. https://doi.org/10.1007/s11042-019-7696-z.

[28] Belmeguenai A, Ahmida Z, Ouchtati S, and Dejmii R (2017) A novel approach based on stream cipher for selective speech encryption. Int. J. Speech Technol. 20:685–698. https://doi.org/10.1007/s10772-017-9439-8.

[29] Available at, https://in.mathworks.com/help/matlab/ref/rms.html (Accessed Jun. 09, 2022).

[30] Available at, https://in.mathworks.com/help/predmaint/ug/signal-features.html (Accessed Jun. 09, 2022).

[31] Wu Y, Noonan JP, and Agaian S (2011) NPCR and UACI randomness tests for image encryption. Journal of Selected Areas in Telecommunications, pp. 31–38.

[32] Biham E, Shamir A (1993) Differential Cryptanalysis of the Data Encryption Standard (DES), Springer-Verlag.

[33] Farsana F and Gopakumar K (2016) A novel approach for speech encryption: Zaslavsky map as pseudo random number generator. Procedia Comput Sci 93:816–823. https://doi.org/10.1016/j.procs.2016.07.302.

[34] Faragallah OS (2018) Secure audio cryptosystem using hashed image LSB watermarking and encryption. Wireless Personal Communications 98:2009–2023. https://doi.org/10.1007/s11277-017-4960-2.

[35] Farsana FJ, Devi VR, and Gopakumar K (2019) An audio encryption scheme based on fast walsh hadamard transform and mixed chaotic keystreams. Appl. Comput. Inform. https://doi.org/10.1016/j.aci.2019.10.001.

[36] Habib Z, Khan JS, Ahmad J, Khan MA, and Khan FA (2017) Secure speech communication algorithm via DCT and TD-ERCS chaotic map. 4th International Conference on Electrical and Electronic Engineering (ICEEE), IEEE, pp. 246–250. https://doi.org/10.1109/ICEEE2.2017.7935827.

[37] Available at, https://en.wikipedia.org/wiki/Mean_squared_error (Accessed Jun. 09, 2022).

[38] Available at, https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio (Accessed Jun. 09, 2022).