Securing Digital Audio Files Using Rotation and XOR Operations

Abdul Gaffar

1 Introduction

Every day 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 the symmetric-key encryption methods are much faster and more efficient, for attaining confidentiality, as compared to the asymmetric-key encryption methods, therefore, we adopt the symmetric-key encryption method in the proposed technique. Note that the Rotation-XOR (RX) operations, used in the proposed technique, are the primitive operations, which are efficiently and directly supported by 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: Sect. 2 provides related works; Sect. 3 gives preliminaries; Sect. 4 describes the encryption and decryption algorithms of the proposed technique; Sect. 5 describes the implementation and

A. Gaffar (✉)
Department of Mathematics and Statistics, Integral University, Lucknow 226 026, UP, India
e-mail: abdulgaffar.lu@gmail.com

B. K. Roy et al. (eds.), *Cryptology and Network Security with Machine Learning*, Algorithms for Intelligent Systems, https://doi.org/10.1007/978-981-99-2229-1_38

experimental results; Sect. 6 discusses security analyses of the proposed technique; Sect. 7 gives comparison of the proposed technique with the recent state-of-the-art techniques; and Sect. 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 the 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 the XOR (eXclusive OR) operation and the Hartley Transform (HT). First of all, a plain audio file is reshaped into a two-dimensional (2D) data block, and then it is XOR-ed with a grayscale image (treated as a secret key). The obtained XOR-ed blocks are then transposed via a chaotic map, followed by an 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 the Elementary Cellular Automata (ECA). The scheme encrypts a plain audio file in three stages: cyclic shift, substitution, and scrambling. The cyclic shift is used 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.

Abdelfatah [6] in 2020 proposed an algorithm for securing audio files in three phases utilizing three secret keys. The first phase is the self-adaptive scrambling of the plain audio files via the first secret key. The 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.

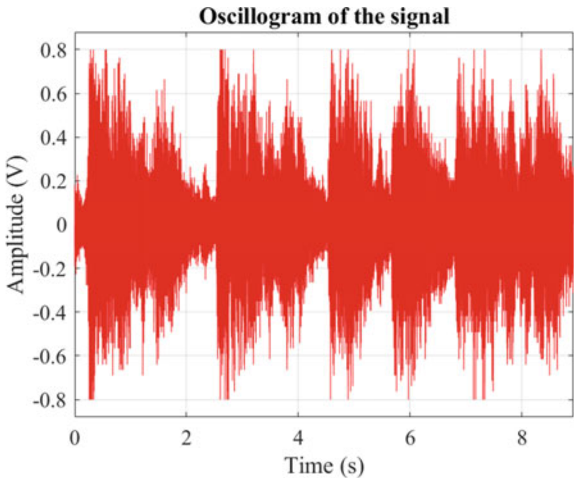


Fig. 1 Oscillogram of the audio file "handel.wav"

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 the floating-point values, i.e., real values. Figure 1 shows the oscillogram (a graph between amplitude 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.

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. Figure 2a demonstrates the concept, wherein MSB is the Most Significant Bit and LSB is the Least Significant Bit.

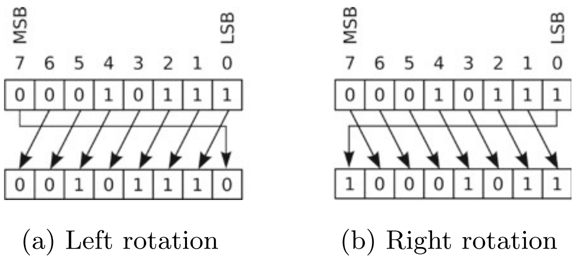


Fig. 2 **a** Left rotation of $x = 0001\ 0111$ by 1-bit and **b** right rotation of x by 1-bit

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. Figure 2b demonstrates the concept.

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 the XOR (denoted by $\oplus$) operation as: if two bits are same, the result is 0; and if not same, the result is 1. It's 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 the 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–$2^8 - 1$).
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.
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.

[1] See [7, 8].

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

1. Convert the WORDS of the audio file P_w into bytes (0–2^8– 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–$2^8 - 1$) into a binary (matrix).
4. Convert the binary (matrix) into the floating-point values via the single-precision floating point (32-bit).
5. Take the 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, $\lceil\cdot\rceil$ denotes the *ceiling* 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 "·" 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 Sect. 4.1), and let the obtained file be P_w of size $1 \times m$.
2. Apply preprocessing on the secret key K (see Sect. 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:

```
for $i = 1$ to $m$
  $b_{i-1} = c_{i-1}$
  $c_i = [b_i \lll \sigma(b_{i-1})] \oplus b_{i-1}$
end for
```

where $b_0 = c_0 = b_m$; "$\lll$" denotes the *left rotation* operator; and "σ" in $\sigma(b_{i-1})$ denotes *sum-of-digits* function, and $\sigma(b_{i-1})$ denotes sum-of-digits of b_{i-1}. For instance, if $b_{i-1} = 123$, then $\sigma(123) = 1 + 2 + 3 = 6$.

(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:

```
for $j = 1$ to $m$
  $c_{j-1} = d_{j-1}$
  $d_j = [c_j \lll \sigma(c_{j-1})] \oplus c_{j-1}$
end for
```

where $d_0 = c_m$ and the 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 the reverse preprocessing on the audio file E of size $1 \times m$ (see Sect. 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 Sect. 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:

```
for $j = m$ to 1
  $c_j = [d_j \oplus d_{j-1}] \ggg \sigma(d_{i-1})$
end for
```

where "$j = m$ to 1" means $j = m, m-1, \ldots, 2, 1$; $d_0 = d_m$; and "$\ggg$" denotes the right rotation.

3. ***First round substitution***.

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

$$C[i] = C[i] \oplus K_{3w}[i], \quad i = 1, 2, 3, \text{and}$$
$$C[m] = C[m] \oplus K_{3w}[4].$$

(b) Do the following:

```
for $i = m$ to 1
  $b_i = [c_i \oplus c_{i-1}] \ggg \sigma(c_{i-1})$
end for
```

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

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

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

5. Apply the reverse preprocessing on the audio file P_w (see Sect. 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, two test audio files of different sample lengths are taken from the *MATLAB* IPT (Image Processing Toolbox).[2] The details of these audio files are provided in Table 1. Also, the oscillograms of the original, encrypted, and the decrypted audio files are shown in Fig. 3.

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

Table 1 Description of the test audio files

File name (.wav)	Channels	Sample rate (in Hz)	Total samples (length)	Duration (in s)	Bits/Sample	Bit rate (in kbps)	Size (in KB)
Handel	1	8192	73113	8.9249	16	131.0720	142.7988
Splat	1	8192	10001	1.2208	16	131.0720	25.1562

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. Keyspace should be very large so that attacks, such as the brute-force [9], known/chosen plaintext [10], 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.

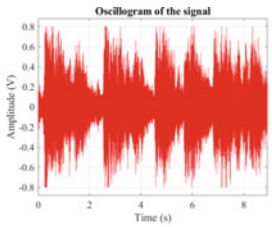
6.2 Encryption Evaluation Metrics

Since any single metric cannot evaluate any encryption algorithm (or any encrypted audio file) fully, so we employ two important metrics, namely, the oscillogram and the number of sample change rates.

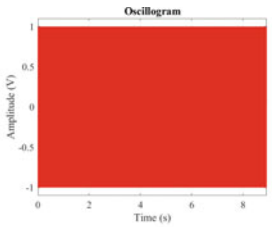
6.2.1 Oscillogram Analysis

The oscillogram is a 2D graph of an audio file (or signal) between the amplitude and the time, generated by the *oscilloscope*, a.k.a. *oscillograph* [11, Chap. 5]. It represents the change in amplitude of an audio file over the time. X-axis represents the time in seconds, while Y-axis represents the amplitude in volts. The oscillograms of the original, encrypted, and the decrypted audio files are shown in Fig. 3.

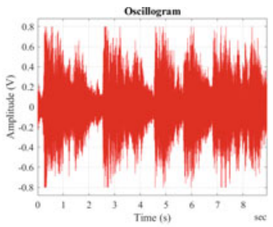
From Fig. 3, we observe that the oscillograms of the encrypted audio files are uniform, unlike those of the corresponding original audio files. Also, the oscillograms of the decrypted audio files are identical to those of the corresponding original files. Thus, our proposed technique performs a robust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



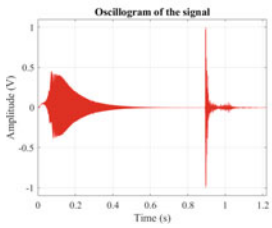
(a) Oscillogram of original handel file.



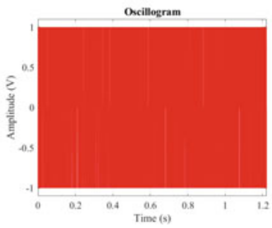
(b) Oscillogram of encrypted handel file.



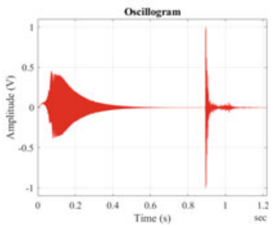
(c) Oscillogram of decrypted handel file.



(d) Oscillogram of original splat file.



(e) Oscillogram of encrypted splat file.



(f) Oscillogram of decrypted splat file.

Fig. 3 Experimental results: Figs. **a** and **d** show the oscillograms of the original audio files; Figs. **b** and **e** show the oscillograms of the corresponding encrypted audio files; and Figs. **c** and **f** show the oscillograms of the corresponding decrypted audio files

6.2.2 Number of Sample Change Rate (NSCR) Test

The NSCR [13] is used to test the resistance of the differential attack [14], 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. 1:

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

where $\beta(s,1)$ is given by Eq. 2:

$$\beta(s,1) = \begin{cases} 0, & \text{if } E_1(s,1) = E_2(s,1) \\ 1, & \text{if } E_1(s,1) \neq E_2(s,1) \end{cases} \tag{2}$$

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)th sample as well as from the last—(l, 1)th sample), l being the total number of samples in an audio file. The obtained NSCR scores are shown in Table 2. Note that, if the calculated/reported

Table 2 NSCR scores of the encrypted images

Method	File Name (.wav)	Duration (in s)	Size (in KB)	Position altered	NSCR Score (in %)
Proposed	Handel	8.9249	142.7988	(1, 1) (l, 1)	100 100
	Splat	1.2208	25.1562	(1, 1) (l, 1)	100 100
Shah [2]	Bells sound	–	–	–	99.9884
Faragallah [4]	Alarm	–	–	–	99.7500
Naskar [5]	Audio-4	–	162	–	99.9958
Abdelfatah [6]	Audio-2	1.7600	296.875	–	99.9700

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 [13], 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 Table 2, which are vulnerable to the differential attack.

6.3 *Decryption Evaluation Metric*

To evaluate the decryption algorithm, i.e., the decrypted audio files, we use an important metric: the mean square error.

6.3.1 Mean Square Error (MSE) Analysis

The MSE [15] is used to judge the decryption quality of any decrypted audio file. The MSE value can be any non-negative integer. *Lower* the MSE, *better* is the decryption quality, in particular, value 0 denotes the perfect decryption, i.e., the original and the decrypted audio files are exactly identical—lossless decryption. The MSE can be calculated via Eq. 3:

$$MSE = \sum_{j=1}^{l} \frac{(P_j - D_j)}{l}\ , \tag{3}$$

where P_j and D_j denote the jth samples of the original and the decrypted audio files, respectively, while the other symbols have their usual meanings.

The values of the MSE between the original and the decrypted audio files are provided in Table 3. 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.

Table 3 The MSE values between the decrypted and the original audio files

Method	File name (.wav)	Duration (in s)	Size (in KB)	MSE
Proposed	Handel splat	8.9249 1.2208	142.7988 25.1562	0 0
Abouelkheir [1]	Sen_4	1.1901	41.0156	3.3161×10^{-11}

6.4 *Key Sensitivity Analysis*

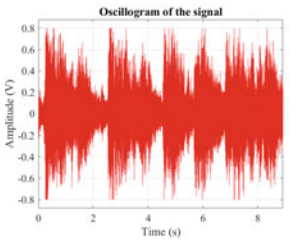
This test is utilized to judge the *confusion* property [3] of any encryption/decryption algorithm. According to Shannon [3], a secure cryptographic algorithm must have the confusion property to thwart statistical attacks. It is the property of confusion that hides the relationship between the encrypted 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 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 the LSB corresponding to the last and the first bytes 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 the encryption/decryption keys λ_1 and λ_2 differ from each other as well as from the secret key K merely by 1-bit.

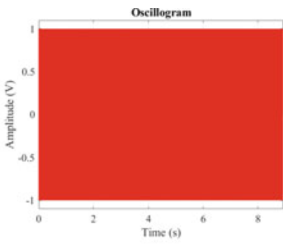
The results of the key sensitivity analysis w.r.t. the encryption (*enc*—in short) and decryption (*dec*—in short) aspects are shown in Figs. 4 and 5, respectively, whence we infer that the proposed technique has a very high bit-level sensitivity, and thus, ensures the property of confusion.

7 Comparison with the Existing Techniques

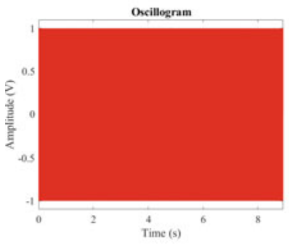
The proposed technique is compared with the recent state-of-the-art techniques based on the commonly available metrics, namely, the NSCR and the MSE. The comparisons of the proposed approach with the recent approaches based on the NSCR and the MSE metrics are provided in Tables 2 and 3, respectively. From these tables, we infer that our proposed technique performs well in terms of the respective compared metrics.



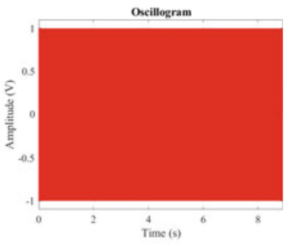
(a) Oscillogram of the original handel file (P).



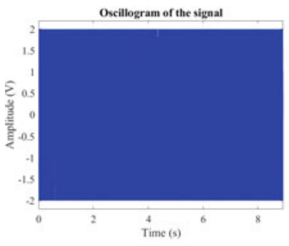
(b) Oscillogram of the encrypted handel file E, where $E = enc(P, K)$.



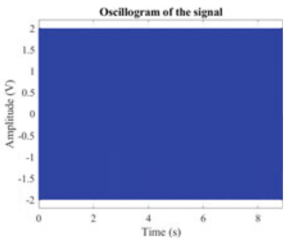
(c) Oscillogram of E_1, where $E_1 = enc(P, \lambda_1)$.



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



(e) Oscillogram of $|E_1 - E|$, where $|\cdot|$ denotes the absolute difference.

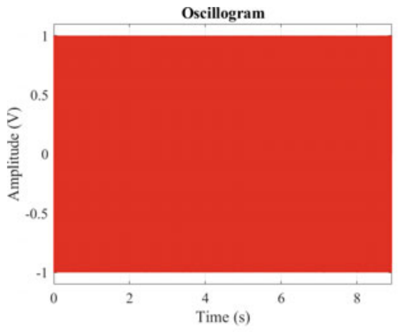


(f) Oscillogram of $|E_2 - E|$.

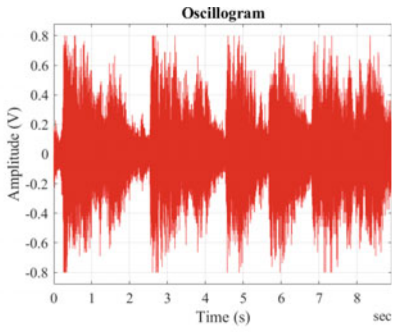
Fig. 4 Key sensitivity analysis w.r.t. the encryption

8 Conclusion

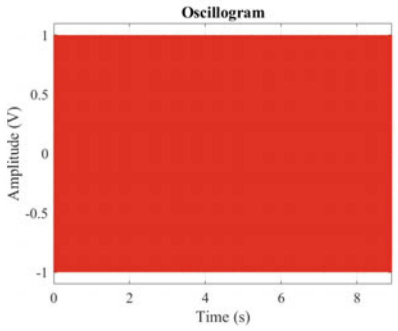
In this paper, we proposed a technique for securing digital audio files based on the WORD-oriented RX operations. Several performance evaluation metrics, i.e., encryption and decryption evaluation metrics, have been used on the audio files of varying sizes from the standard database, in order to empirically 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.



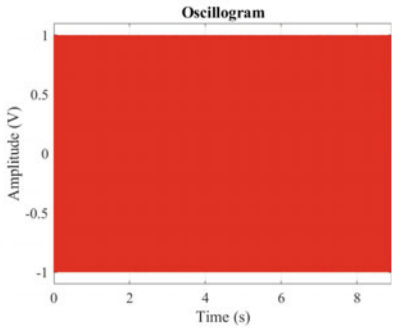
(a) Oscillogram of the encrypted handel file E, where $E = enc(P,\, K)$.



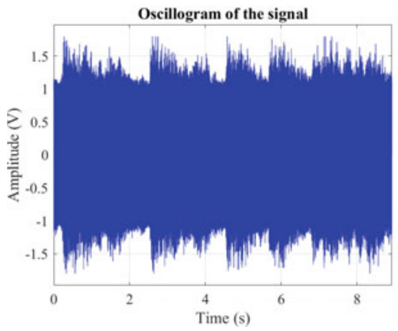
(b) Oscillogram of the decrypted handel file (D) using correct secret key K.



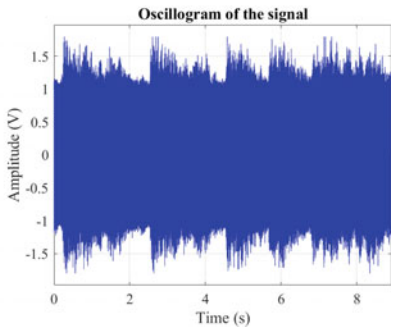
(c) Oscillogram of D_1, where $D_1 = dec(E,\, \lambda_1)$.



(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. the decryption

References

1. Abouelkheir E, Sherbiny SE (2022) Enhancement of speech encryption/decryption process using RSA algorithm variants. Hum-Centric Comput Inf Sci 12(6). https://doi.org/10.22967/HCIS.2022.12.006
2. Shah D, Shah T, Hazzazi MM, Haider MI, Aljaedia, 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. Bell Syst Tech J 28(4):656–715. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x
4. Faragallah OS, 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, 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. Abdelfatah RI (2020) Audio encryption scheme using self-adaptive bit scrambling and two multi chaotic-based dynamic DNA computations. IEEE Access 8:69894–69907. https://doi.org/10.1109/ACCESS.2020.2987197
7. Available at https://in.mathworks.com/help/matlab/matlab_prog/floating-point-numbers.html. Accessed 05 Nov 2022
8. Available at https://en.wikipedia.org/wiki/Single-precision_floating-point_format. Accessed 05 Nov 2022
9. ECRYPT II yearly report on algorithms, keysizes NS (eds) (BRIS) 2011–2012. https://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf. Accessed 05 Nov 2022
10. Stinson DR (2006) Cryptography: theory and practice. Chapman and Hall CRC, UK
11. Kularatna N (2002) Digital and analogue instrumentation: testing and measurement. IET, UK. https://doi.org/10.1049/PBEL011E
12. 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
13. Wu Y, Noonan JP, Agaian S (2011) NPCR and UACI randomness tests for image encryption. J Sel Areas Telecommun 31–38
14. Biham E, Shamir A (1993) differential cryptanalysis of the data encryption standard (DES). Springer, US
15. Hossein PN (2014) Introduction to probability, statistics, and random processes. Kappa Research LLC, USA