Enhancing LSB Using Binary Message Size Encoding for High Capacity, Transparent and Secure Audio Steganography—An Innovative Approach

MAHMOUD M. MAHMOUD AND HUWAIDA T. ELSHOUSH
Department of Computer Science, Faculty of Mathematical Sciences and Informatics, University of Khartoum, Khartoum 11111, Sudan
Corresponding author: Huwaida T. Elshoush (htelshoush@uofk.edu)

ABSTRACT We propose a novel LSB-BMSE method that enhances LSB audio steganography. It uses an innovative mechanism, Binaries of Message Size Encoding (BMSE), to embed a secret message after hiding its size in random samples. First, the secret message is compressed using Huffman coding, then encrypted by AES-128. The audio cover is split into number of blocks depending on secret message size. A secure key $BMSE$, output from the BMSE mechanism, is used to embed the secret message in random blocks and bytes adaptively according to its size. It is implemented using MATLAB and standard parameters: Perceptual Evaluation of Speech Quality and NIST Statistical Test Suite were used to measure imperceptibility between cover & stego audios and randomness of BMSE mechanism respectively. The fidelity was tested using Mean Square Error, Peak Signal to Noise Ratio and Signal-to-Noise Ratio. Comprehensive experiments on widely used metrics demonstrate that LSB-BMSE significantly surpasses existing methods in terms of hiding capacity and imperceptibility. Moreover, LSB-BMSE shows resistance to brute force attacks and statistical analysis. Although it was robust towards re-sampling attacks, nevertheless was not vigorous towards noise nor LSB attacks. The key $BMSE$ complies with Kerckhoff’s principle and exhibits avalanche criteria. The tested proposed LSB-BMSE proclaimed its effectiveness.

INDEX TERMS Adaptive steganography, audio steganography, hiding information, least significant bit (LSB).

I. INTRODUCTION
To increase the security of secret messages in an open system environment, it is very important to hide two properties of the message, its context and existence in order to cover it from unauthorized recipients. Steganography is a science of invisible or secret communication [1]–[3]. Audio steganography is the art and science of hiding digital data such as text messages and binary files into audio files such as WAV, MIDI, AVI, MPEG and MP3 files [3]–[5]. The reason for considering audio as cover media is the representation of amplitudes in real number format which causes very small distortions after embedding the bits of target data.

An efficacious steganography scheme should meet the three requirements, namely capacity, transparency and robustness. Capacity is the maximum amount of secret information that can be embedded in a file [6]. Imperceptibility indicates that a steganographic system is perfectly secure if the statistics of the cover and the stego files are identical. The last requirement is robustness which denotes the resistance of stego file to various attacks and its capability to retrieve secret message with a minimum error [7]–[10]. These parameters are conflicting as the increase in hiding capacity leads to degradation in the robustness of secret message and transparency of stego file and vice versa. Thus the trade-off among them is difficult [7], [8]. Hence, the main aim of steganography is to increase the steganographic capacity and enhance the security and the imperceptibility while maintaining the robustness [11]–[13].

Depending on the hiding mechanism, audio steganography may be categorized into temporal domain [7], [8], [14], transform domain [9], [15]–[22] and compressed [23], [24].
Transfer domain methods have good robustness but low capacity and relatively complex calculations, in contrast to time-domain methods that have high transparency and capacity but low robustness.

LSB algorithm is the most common, simplest and easiest technique to implement [4], [25]–[30]. The low complexity is a major advantage while easy detection is an obvious disadvantage [5], [31]. The structural limitations of the traditional LSB method motivate for an alternate effective and enhanced approach. An audio steganography technique is desired that should have the capacity comparable to the traditional LSB technique but maintaining a high value of robustness and imperceptibility. The embedding sequence is expected to be as less predictable as possible to make the proposed method more secure.

In statistical techniques, messages are embedded by changing many properties of the cover, which involves splitting the cover into blocks and then embedding one message bit in each block. Hence, the cover block is modified only when the size of secret message bit is one otherwise no modification is necessarily [6], [32]. The proposed method uses this technique.

Adaptive techniques studies the statistical features of the cover before embedding the secret data. They interact with the cover object in an intelligent way in order to either maximize its hiding capacity, or minimize the distortion caused by hiding [17], [33], [34].

The proposed method aims at maximizing both hiding capacity and imperceptibility, while being secure. It is an adaptive statistical technique in the sense that the cover is split into blocks according to the size of the cover audio and secret message. If the secret message is small, the number of blocks will be large and hence the bits of the secret message will be scattered randomly (one secret bit in each block) in the audio cover file. Therefore, keeping the distortion to the minimum and hence enhancing the imperceptibility of the proposed method and also its hiding capacity.

In this study, a novel enhanced LSB audio steganography using BMSE is proposed. The LSB-BMSE method exhibits high capacity and transparency of the stego file while achieving robustness of the secret message. It uses a novel mechanism of Binary Message Size Encoding (BMSE) that hides the size of the message in random bytes, thus preserving high payload and high security. Similarly, depending on the BMSE mechanism, a secure key is generated to boost the secrecy of the embedding process by hiding in random bytes and blocks in the cover audio file. Using the statistical technique, the audio cover is split into blocks, and the embedding is done in bytes of random blocks adaptively according to the secret message size. First, the Huffman compression algorithm is used to reduce the size of the confidential message. Furthermore, the message is encrypted using AES before hid using the proposed LSB-BMSE algorithm. Thus, a multilayer security is achieved by AES, and moreover by embedding the secret message in a nondeterministic way using the BMSE mechanism.

Huffman coding is a lossless data compression technique, which is based on the frequency of occurrence of a data clause. The concept is to use a minimum number of bits to encode the data that happens more frequently [35]. The lossless compression of the Huffman encoding improves the capacity of the overall method [36].

A preliminary version of this work appears in [37], where the idea of the algorithm is presented. In this paper, we further tackle with the challenge of keeping a high hiding capacity while preserving the fidelity and security. The contributions of this work are hereby summarized:

- We proposed a novel effective secure LSB-BMSE method that alleviate the difficulty of increased hiding capacity while maintaining high imperceptibility. Additionally, it is robust against re-sampling attacks.
- We developed an innovative BMSE mechanism that hides the secret message size in random blocks after splitting the audio cover into blocks. Moreover, its output key$_{BMSE}$ is used to hide secret message arbitrarily in random blocks and bytes.
- The proposed BMSE method acts as a pseudo-Random Number Generator (PRGN).
- Our proposed method achieved effectual performance over the state-of-the-art schemes.

The paper is structured as follows: section II surveys the LSB enhanced approaches for audio steganography in temporal domain. Section III elucidate the proposed method. The experimental results are discussed in section IV. Finally, section V concludes the paper.

II. RELATED WORK

Although traditional LSB is one of the simplest and easiest approaches, it has some weaknesses. The secret data is embedded in a deterministic way, which gives the attacker the chance to extract or even damage the hidden data without apparent impact in the perceptual quality of the stego audio. Thus, LSB method is prone to intentional attacks as data are embedded in LSBs only. Moreover, unintentional attacks like noise disturbances are the cause of data loss [38]. Thus, it is also highly sensitive towards attacks such as addition of noise, LSB removal, compression, amplification, re-sampling, etc. For these reasons, various approaches to modify the traditional LSB have been proposed by the researchers to reduce these limitations. Ergo, this section presents a state of the art of the LSB enhanced approaches for audio steganography in temporal domain.

To reduce the effect of noise attacks, researchers [14], [39]–[41] embed the secret bits in the higher depth of cover audio (4th to 8th bit locations of sample), but the price is rather paid by the decrease on imperceptibility between cover and stego audios. M.A.Ahmad et al. [39] avoid embedding in silent intervals and low amplitudes, and rather embed in higher bits at the 8th LSB to reduce the effect of noise attacks. Their idea improves the capacity and robustness without affecting the perceptual transparency but still the embedding capacity is reduced compared to the traditional LSB.
S. S. Bharti et al. [14], in 2019, noticed that the 4 least significant bits are more affected by attacks like LSB attack, re-sampling attack etc. Thus they embed secret audio by separate processing of the amplitudes (in 1st to 4th LSB, one at a time) and signs (in 5th to 8th LSB, four at a time). Their proposed approach is robust towards LSB attack, re-sampling attack and also more resistive towards noise attack, and moreover achieving high embedding capacity. [40] designed a high bit rate LSB audio watermarking method that reduces embedding distortion of the host audio with increased capacity of secret text, thus resulting in increased robustness against noise addition, but limited by perceptual transparency. In [41], a stego key and AES encryption are applied to increase the security of the stego audio. Their scheme is robust and able to withstand steganography unintentional attack which is compression till 17% at maximum payload of 953 bps in the 11th LSB.

Same as [41], other researchers [42]–[44] also combined steganography with encryption to enhance security. Researchers [43] hide secret data in both audio and video with the help of Diffie Hellman key exchange to enhance the security and also to generate the key based index for LSB insertion. [44] embed text encrypted with RC4 encryption cipher in audio steganography, but their method has a limitation as it is only supported by .wav audio.

Some other researchers [4], [7], [8], [38], [45] use chaotic map to embed in a nondeterministic way using different approaches. For instance, [4] proposed a chaos based audio steganography and cryptography method. The secret message is encrypted first by one-time pad algorithm, where the key is generated by Piecewise Linear Chaotic Map (PWLCM) chaotic map. In the steganography phase, the indices of a second generated sequence of PWLCM is encrypted first by one-time pad algorithm, where the key is known to everyone. Thus, the motivation of this work is to develop an enhanced LSB audio steganography method using trusted third party random key indexing method which provides extra layer of security by generating a secondary key and message retrieval password. Their approach provides high Bit Error Rate (BER) and SNR dB values when tested on 32-bit and 16-bit audios.

Working on a different dimension, [48] increased the hiding capacity by presenting an audio data hiding algorithm that uses Modulus function based 8 in order to suit the octal secret digits with the sample remainder values. The difference between the secret digits and the sample remainder values is subtracted from the value of the cover sample. Based on uint8, uint16, and uint24, this algorithm achieves high embedding capacity of three bits per sample.

Most of the previous studies related to the security enhancement of the steganography process were based on mechanisms that lacked adaptivity and randomness. In other words, it is easy to trace these mechanisms to reach to the secret message or part of it. Moreover, noise attacks, re-sampling attacks, and robustness are the major problems existing in the literature for audio steganography.

To overcome the shortcomings of the aforementioned methods, we propose a key based adaptive and nondeterministic steganography technique. It complies with Kerckhoff’s principle of cryptography [49], [50], which says an exemplary method should be secure even if all the details of that method except the key is known to everyone. Thus, the motivation of this work is to develop an enhanced LSB audio steganography approach for hiding secret text with high capacity without reducing the perceptual transparency, in addition to being secure.

III. THE PROPOSED METHOD

The traditional LSB uses the last consecutive eight bytes to conceal the size of secret message. It hides one bit in each byte, hence uses 8 bits to encode the message size. Those bytes positions are agreed on by the sender and receiver. So before the receiver retrieves the hidden secret message, he/she extracts the 8 bits in the last 8 bytes and then convert them into decimal to know the size of the hidden secret message. If the secret message size is large, the traditional
LSB may use the last 16 bytes and so on. This is a limitation in the traditional LSB, as the audio cover file may have more capacity. Besides being deterministic and prone to statistical attacks, noise addition, re-sampling and LSB attacks, the traditional LSB technique has two more weaknesses:

- When the position of that 8 bytes is known that makes the extraction of a secret message very easy, and this is because the size of the secret message can be easily identified.
- The maximum number that can be hidden in 8 bytes is \(2^8 = 256\), which means if the message size is greater than 256 the hiding operation will need more bytes to hide, and this will become a limitation.

To build a good mechanism to enhance the LSB algorithm, there are some criteria that must be considered:

- **Randomness**: the automatism must be randomized as much as possible to make the hiding process non-traceable to their behavior, i.e. should not be based on known behavior.
- **Variability**: the random should be based on variable and when those variables are many, the secrecy is increased.
- **Complexity**: means if the key is not known, it is very difficult to extract the correct message.

First, in the novel proposed method, the secret message is compressed using Huffman encoding, then encrypted using AES-128. In the foreground, LSB algorithm is in front of attack. So it was enhanced and strengthened against penetration, thus bolstering the security of LSB using BMSE mechanism to hide the size of the message in random samples and thereafter the message itself in random blocks and bytes to ensure that the secret message has a high degree of randomness, complexity and hence protection.

The novel proposed LSB-BMSE method solved the two above problems of traditional LSB, precisely as follows:

- The message size is not hidden as in the traditional LSB in a sequential manner. Actually, the binary message size encoding is calculated and these BMSE bits are hidden in the cover file in a very secure random way.
- For even large message sizes, the BMSE bits representation is not large compared to the traditional LSB representation of the message size as depicted in figure 1. Hence, the proposed novel LSB-BMSE method provides higher capacity.

The following subsections details the embedding and extraction processes using the novel LSB-BMSE method.

**A. THE EMBEDDING PROCESS USING THE NOVEL ENHANCED LSB-BMSE ALGORITHM**

1) **PREPROCESSING**

Compaction using Huffman algorithm is added as a basic operation to minimize the size of the secret message \(Msg\). This is because whenever the secret message is small, the effect of hiding on the cover file will be small. Then, if the existence of the secret message is identified, there must be some mechanism to prevent hackers from accessing it. Hence, it is encrypted with AES-128. For simplicity, the plaintext message and the compressed encrypted message are both referred to in this research as \(Msg\).

2) **BMSE CALCULATION**

First, the BMSE of the \(Msg\) is calculated using the following novel method as detailed in algorithm 1, *Function: CalculatingBMSE(MsgSize)*.

To explain more clearly, an example of encoding binaries of messages of size 7 bytes is depicted in figure 1. First, the 7 bytes is converted into binary which yields \((111)_{10}\). As it is represented in 3 bits, i.e. its length is 3, hence \((3)_{10} = (0000 0011)_{2}\). Note that this length (3 bits) has to be represented in 8 bits. Then, the size of the \(Msg\) itself (which is 7) is written in the successive 3 bits. Therefore, the encoding of the binaries of message size 7 bytes (BMSE of 7 bytes) = 0000 0011 111. Hence, when extracting any \(Msg\), the first 8 bits when converted to decimal indicates the number of bits and after those 8 bits is the size of \(Msg\).

*Function: CalculatingBMSE(MsgSize)\* returns \(N_{BMSE}\) and the BMSE of the \(Msg\), \(N_{BMSE}\), which is the number of bits of BMSE, is needed to hide these BMSE bits in the audio cover file using algorithm 1.

**Algorithm 1: The BMSE Algorithm**

**Input:** \(MsgSize\) // \(MsgSize\) is the secret message \(Msg\) size in bytes

**Output:** \(N_{BMSE}, BMSE\) // \(N_{BMSE}\) is the number of bits of BMSE // BMSE is the Binaries of Message Size Encoding of \(Msg\)

1) **Function** CalculatingBMSE \((MsgSize)\) :

2) \(c = \text{BinaryOf}(MsgSize)\) // Convert \(MsgSize\) to bits

3) \(l = |c|\) // \(l\) is the length of \(c\)

4) BMSE = BinaryOf(l) || BinaryOf(MsgSize) // || is concatenation. Note that BinaryOf(l) has to be in 8 bits

5) \(N_{BMSE} = |BMSE|\)

6) **Return** \(N_{BMSE}, BMSE\)

7) **End Function**

3) **EMBEDDING THE BMSE BITS ALGORITHM**

After calculating the BMSE of the \(Msg\), *Function: EmbedExtractBMSE(AF, Key, N, Status, BMSE)* in algorithm 2, with Status==Embed, is executed to embed the BMSE bits of
Algorithm 2: The Embedding/Extracting BMSE Algorithm

Input: \( AF, Key, N, Status, \{BMSE\} \)  
// \( AF \) is the audio file  
// Key is the hiding key 128 bits; written as \( Key_1, Key_2, \ldots, Key_{128} \)  
// \( N \) = number of BMSE bits to be hidden or extracted, thus how many times loop is repeated  
// Status has two values either embed or extract  
// BMSE is an optional parameter included only when the Status is embed  

Output: \( AF_{\text{leftover}}, Key_{BMSE}, \{S - \text{array}\}, \{K - \text{array}\}, \{\text{MsgSize}\} \)  
// \( AF_{\text{leftover}} \) = leftover blocks from \( AF \)  
// \( Key_{BMSE} \) is a modified key after hiding or extracting BMSE  
// \( S \)-array, where BMSE bits are embedded, is an optional parameter output only when Status is embed  
// \( K \)-array, which is an auxiliary array, is an optional parameter output only when the Status is embed  
// \( \text{MsgSize} \) (Msg size in bytes) is an optional parameter output only when Status is extract

Function EmbedExtractBMSE(\( AF, Key, N, Status, \{BMSE\} \)):

1. \( AF = AF_1, AF_2, \ldots, AF_{\text{LengthOfAF}} \)  
   Initialize S-array and K-array to 0
2. \( \text{for } i = 1 \text{ to } N \) do
3.      \( CO = \text{BinaryOf}(\text{LengthOf}(AF)) \) // Convert no. of bytes of \( AF \) to binary & find its length
4.      if (S-array is empty) OR (last two bits of BinaryOf(K) = "11") then
5.         \( A = Key_1, Key_2, \ldots, Key_{CO} \) // A equals the first \( CO \) bits of the key
6.      else if last two bits of BinaryOf(K) = "00" or "01" then
7.         \( A = \text{Middle CO bits from Key} \)
8.      else if last two bits of BinaryOf(K) = "10" then
9.         \( A = \text{Last CO bits from Key} \)
10.     \( G = \text{DecimalOf}(A) \) // read byte no. \( S \) from \( AF \) and store in \( S \)-array,
11.    if Status == Embed then
12.       \( \text{Embed BMSE} \) in S-array, \( i \) // Embedding bit BMSE(\( M_i \)) in \( S \)-array, using LSB
13.    else
14.       \( \text{Extract BMSE}_i \) from S-array, \( i \) // Extracting bit BMSE(\( M_i \)) from S-array,
15.    end
16.    if G is even then
17.       \( \text{K-array}_i = AF_{S+2} \) // read byte \( AF_{S+2} \) and store in K-array,
18.    else
19.       \( \text{K-array}_i = AF_{S-1} \) // read the preceding byte \( AF_{S-1} \) and store in K-array,
20.    end
21.    \( K = \text{ByteValueOf}(\text{K-array}, K) \) \( \ll \) \( \text{AF} \) bytes two places and arrange \( AF_1, AF_2, \ldots, AF_{\text{LengthOf}(AF)} \)
22.    \( \text{LengthOf}(AF) = \text{LengthOf}(AF) - 2 \) // One byte is used for S-array, and another for K-array, \( K \) has \( n \) bits
23.    if \( \text{LengthOf}(\text{BinaryOf}(K)) < \text{LengthOf}(A) \) then
24.       \( L_A = K_1, K_2, \ldots, K_n, K_1, K_2, \ldots, K_m \) // Repeat \( K \) bits until \( L_A = \text{the length of } A (\geq m + n) \)
25.       \( L_A = K_1, K_2, \ldots, K_m \) // Start reading bits of \( L \) till length of \( A \)
26.    end
27.    \( Z = L_A \oplus A \) // \( L_A \) XOR it with A
28.    \( \text{MixRemainKey} = \text{MIX}(\text{RemainKey}_{left}, \text{RemainKey}_{right}) \) // Split remainingOf(key), after taken away A bits, into two halves and MIX their bits
29.    if BitsOf(MixRemainkey) > BitsOf(Z) then
30.       \( P = \text{Quotient}(\text{BitsOf(MixRemainkey)} \div \text{BitsOf(Z)}) \)
31.       Key = MIX(Z,MixRemainKey \( Z \)) // after every P bits of MixRemainKey, put 1 bit from Z
32.    else
33.       \( P = \text{Quotient}(\text{BitsOf(Z)} \div \text{BitsOf(MixRemainkey)}) \)
34.       Key = MIX(Z,MixRemainKey) // after every P bits of \( Z \), put 1 bit from MixRemainKey
35.    end
36.    Return \( AF_{\text{leftover}}, Key_{BMSE}, \{S - \text{array}\}, \{K - \text{array}\}, \{\text{MsgSize}\} \)  
   // optional outputs \( S \)-array and \( K \)-array are only when Status is embed, while \( \text{MsgSize} \) is when Status is extract

End Function
Algorithm 3: The Embedding/Extracting Secret Message Algorithm

**Input:** AF<sub>leftover</sub>, Key<sub>BMSE</sub>, Status, N, {Msg}, {S-array}, {K-array}

// AF<sub>leftover</sub> is the leftover blocks from audio file AF
// Key<sub>BMSE</sub> is the modified 128-hiding key after hiding or extracting the BMSE
// Status has two values either embed or extract
// N is the message size in bits thus how many times the loop is repeated
// Msg (=m<sub>1</sub>, m<sub>2</sub>,...,m<sub>NMsg</sub> in binary) an optional parameter included only when Status is embed
// S-array is an array where BMSE bits of secret message are embedded. It is an optional parameter included only when the Status is embed to form Stego file
// K-array is an auxiliary array used in embedding the BMSE. It is an optional parameter included only when the Status is embed to form Stego file

**Output:** Secret message Msg, Stego File

// If Status is extract, the output is Msg. Otherwise if Status is embed, then it returns AF<sub>leftover</sub>, S-array and K-array in their original positions in CF to form stego file

```plaintext
Function EmbedExtract (AF<sub>leftover</sub>, Key<sub>BMSE</sub>, Status, N, {Msg}, {S-array}, {K-array}) :

\[
B = \frac{\text{LengthOf}(AF_{leftover})}{N}
\]

// B = block size, which is the number of bytes of AF<sub>leftover</sub> divided by N
AF<sub>leftover</sub> = AF<sub>block1</sub>, AF<sub>block2</sub>,...,AF<sub>blockN</sub> // Split the AF<sub>leftover</sub> into N blocks and arrange them. Note:
number of blocks = number of bits in secret Msg; and size of each block = B bytes
Split the AF<sub>leftover</sub> blocks into B bytes // Each bit of Msg is hidden in (or extracted from) one byte of a block of AF<sub>leftover</sub>
V = |BinaryOf(B)| // V is how many bits from the key are considered in each step
Arrange Key<sub>BMSE</sub> from 1 to EndOf(Key<sub>BMSE</sub>) in V's array // BinaryOf(Key<sub>BMSE</sub>) (=bits of hiding secret Key<sub>BMSE</sub>) are split into V's bits & put in an array of V<sub>1</sub>, V<sub>2</sub>...., EndOf(Key<sub>BMSE</sub>)
x := 0
j := 0
k := 0
for i = 1 to N do

// Calculate block no. H to hide or extract bit m<sub>x</sub> of Msg
H = (B + 1) * x + j

if H > N then

if Status == Embed then

[ \ll m<sub>x</sub> x places ] // ‘\ll’ means shift the bits of the secret message x places
end
x := 0
j := j + 1
goto step 11
end

if k > EndOf(Key<sub>BMSE</sub>) then

k := 0 // If there are still more bits of the secret message to be hidden, then start the Key<sub>BMSE</sub> selection bits from the beginning
end
W = V<sub>k</sub> // is the binary value of array V<sub>k</sub> from key<sub>BMSE</sub>
D<sub>i</sub> = DecimalOf(W) // Decimal representation of binary W
F = D<sub>i</sub> mod B // Compute byte no. F inside block H to hide/extract m<sub>x</sub> of Msg
if Status == Embed then

Embed bit m<sub>x</sub> inside the block H, in byte F // Embedding inside a byte is done using LSB
else

Extract bit m<sub>x</sub> from block H, in byte F and save in array T<sub>i</sub> // m<sub>x</sub> will be read from SF & saved in array T<sub>i</sub>
end
x := x + 1
k := k + 1
end

if Status == Embed then

Return Stego file = AF<sub>leftover</sub>, S-array and K-array in their original positions in AF
else if Status == Extract then

Return Secret message Msg = T<sub>1</sub>, T<sub>2</sub>..... T<sub>NMsg</sub>
end

End Function
```
FIGURE 2. The novel proposed LSB-BMSE embedding algorithm flowchart.

the $Msg$. The loop is executed $N_{BMSE}$ times (=number of BMSE bits), which is calculated from algorithm 1, **Function: CalculatingBMSE($Msg$Size)**.

Two arrays representing bytes (or samples) of the audio cover file, are used in embedding the BMSE bits, namely S-array and K-array. The BMSE bits are actually hid in the S-array bytes in an LSB fashion, but the selection of the bytes is performed in a random way. Thus assuring the robustness of the novel method. Some operations on the length of the audio cover file and the key are carried out to determine the S-array byte value, and accordingly the K-array byte value is selected. These bytes value of the K-array, together with the previous key, determines the randomness of the embedding process by yielding the key to the next step of the loop. Each step of the for-loop uses the key of the previous step, and the final output key, $Key_{BMSE}$, is used in embedding the $Msg$ itself using algorithm 3, with $Status==Embed$.

The left part of the flowchart in figure 2 depicts the operations of embedding BMSE bits in an audio cover file.

**Algorithm 4: The Novel Proposed LSB-BMSE Embedding Algorithm**

**Input:** CF: audio cover file

Key: Hiding key 128 bits

$Msg$: the secret message

**Output:** SF: Stego file containing the compressed encrypted hidden secret message $Msg$

1. $Compress(Msg)$ // Compressing $Msg$ using Huffman Algorithm

2. $Encrypt(CompressedMsg)$ // Encrypting the compressed $Msg$ using AES algorithm.

Note: For simplicity, the plain message and the compressed encrypted message are both referred to as $Msg$

3. $CalculatingBMSE(MsgSize)$ // Algorithm 1

4. $EmbedExtractBMSE(CF, Key, N_{BMSE}, Embed, BMSE)$ // Algorithm 2

5. $EmbedExtract(AF_{leftover}, Key_{BMSE}, Embed, N_{Msg}, Msg, S-\text{array}, K-\text{array})$ // Algorithm 3

6. $Return SF$ // Stego file = $CF_{leftover}$, S-array and K-array in their original positions in CF

4) **EMBEDDING THE SECRET MESSAGE ALGORITHM**

Depending on $Key_{BMSE}$, a specific byte is selected in a random block to embed the secret message $Msg$, see the right part of the flowchart of figure 2. $Key_{BMSE}$ is output from algorithm 2 and used together with other parameters in calling Function $EmbedExtract(AF_{leftover}, Key_{BMSE}, Embed, N_{Msg}, Msg, S-\text{array}, K-\text{array})$ of algorithm 3.

$AF_{leftover}$ is the leftover bytes from the cover file after embedding the BMSE in S-array and eliminating the K-array from the audio cover file. The secret message $Msg$ is actually embed in the $AF_{leftover}$.

Algorithm 3 starts with dividing the leftover audio cover file into blocks, and then further into bytes to hid the $Msg$. The selection of the blocks, and hence the bytes inside the blocks are done in a random fashion. But inside the bytes, the $Msg$ is hid in an LSB fashion.

5) **GENERATING STEGO-AUDIO PROCESS**

After the hiding process is completed, all the previously extracted bytes of S-array (containing the BMSE bits), and the K-array, are returned together with the leftover audio cover file (where the $Msg$ is embed) in their correct positions to form the Stego-audio file. Thus, the hiding process is completed using the proposed novel LSB-BMSE method. Flowchart 2 depicts the whole embedding process.

Algorithm 4 presents the steps of the embedding process and the calling of the specified functions. Each time the main key is changed, a different $Key_{BMSE}$, block number and byte number is calculated. Thus this makes the embedding process a dynamic one, hence increasing the security of the enhanced LSB-BMSE method.
B. EXTRACTION PROCESS USING THE PROPOSED ENHANCED LSB-BMSE ALGORITHM

The following subsections detail the extraction process of the proposed novel LSB-BMSE method. See algorithm 5 and figure 3.

1) THE BMSE EXTRACTION ALGORITHM

Algorithm 5 takes as input the extraction key and the Stego-audio file. It calls Function: EmbedExtractBMSE(SF, Key, 8, Extract) with Status==Extract. First, the 8 bits of the BMSE are extracted from the Stego-audio file. These bits reveal how many successive bits hides the size of the secret message Msg. Hence, these bits are extracted from the S-array and converted into decimal to know the number of bits containing the size of the Msg. Thereafter, the same function, Function: EmbedExtractBMSE(SF, Key, Q, Extract) is called again with the leftover stego-audio file, Status==Extract and Q (=number of bits containing the size of the Msg), together with the returned key, KeyBMSE. Now the size of the Msg in bytes, namely MsgSize, can be retrieved form the output and used in extracting the Msg itself. The left part of figure 3 presents this step.

2) EXTRACTING THE SECRET MESSAGE ALGORITHM

The output from the previous step, MsgSize is the size of message in bytes. So Function: EmbedExtract(AF leftover, KeyBMSE, Extract, 8 × (MsgSize)) is called with 8 × (MsgSize) representing the number of bits of the Msg. Lastly, the retrieved Msg is decompressed using Huffman algorithm, then decrypted using AES-128 to extract the secret message Msg. The right part of figure 3 depicts the extraction of the secret message Msg.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents series of experiments using objective metrics for evaluating the performance and demonstrating the efficiency of the novel proposed method LSB-BMSE in relation to the transparency of the stego audio, the hiding
capacity and the statistical analysis tests in terms of histogram distribution. Furthermore, the robustness of the proposed method against attacks is tested. From the security aspect, the randomness and the avalanche criterion have been investigated. Moreover, comparisons to related schemes are also conducted.

A. PRELIMINARIES

The proposed novel method LSB-BMSE is implemented using MATLAB software version R2020b. All the experimental results were tested on a laptop with Windows 10, Core i5 processor with 2.5 GHz speed and 4 GB for RAM.

The same audio specifications of the cover audio files have been adopted to evaluate the performance of the proposed method and to compare its performance with the related work, Ahmed et al. (2010) [39], Ali et al. (2018) [7], and Bharti et al. (2019) [14], discussed in section II. For this reason, uncompressed audio files are used as cover audios, which were selected from the GTZAN dataset [51], [52]. The specifications of the audio files used in the experiments are listed below in Table 1.

| Specification                | Value      |
|------------------------------|------------|
| Bit per sample               | 16         |
| Number of samples            | 661500     |
| Channel                      | Mono       |
| Audio type                   | Music      |
| Duration in Seconds          | 1-30       |

B. IMPERCEPTIBILITY ANALYSIS

Imperceptibility or transparency means the closeness of property between the stego and reconstructed files and the original cover and secret files, respectively. This parameter also means minimum degradation and is inversely proportional to the hiding capacity. Specifically, high distortion and low transparency result in high hiding capacity [7], [8].

In terms of transparency, the following objective metrics: Perceptual Evaluation of Speech Quality (PESQ), Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR) and Signal-to-Noise Ratio (SNR) are used to gauge the performance of the proposed method [7]:

1) PERCEPTUAL TRANSPARENCY

PESQ is used to measure the similarity between two audio signals [53], [54]. Its score varies between 1 and 4.5, where ‘4.5’ indicates that both cover and stego audios are perceptually similar while ‘1’ indicates dissimilar, ergo the acceptance rate of PESQ should be ≥3.8 [14].

2) MEAN SQUARED ERROR (MSE)

MSE is the average square of the differences between the cover audio (input) and stego audio (output) signals. It can measure the distortion in the audio. Thus, when this value decreases to zero, the fidelity of the input and output signals becomes similar, and hence better performance of the algorithm. It is expressed in decibel (dB) and is defined as in equation 2 [7]:

\[ \text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (s_{1i} - s_{2i})^2 \]  \hspace{1cm} (2)

where \( s_{1i} \) and \( s_{2i} \) are the \( i^{th} \) samples of the input and output signals, and \( N \) is the number of signal samples.

3) PEAK SIGNAL TO NOISE RATIO (PSNR)

PSNR measures (in dB) the maximum signal to noise ratio of a given signal, and is given by equation 3 [7]:

\[ \text{PSNR} = 10 \log_{10} \left( \frac{(2^n - 1)^2}{\text{MSE}} \right) \]  \hspace{1cm} (3)

where \( n \) is the maximum number of bits used to represent each signal sample.

The lower the MSE value, the higher the steganography quality and vice versa. And the higher the PSNR’s value, the greater the quality of the concealment, which means better quality (less distortion). As can be seen in equation 3, it also depends on MSE (the lower the MSE rate, the higher the PSNR rate).

4) SIGNAL-TO-NOISE RATIO (SNR)

SNR measures the distortion in the fidelity between two signals, input and output. Thus it evaluates the quality of the output signal after embedding process in decibels (dB) [55]. International Federation of the Phonographic Industry (IFPI) states that the SNR value should be more than 20 dB to be acceptable. In fact, higher SNR means better invisibility of the algorithm. It is expressed as in equation 4 [7], [15]:

\[ \text{SNR} = 10 \log_{10} \left( \frac{\sum_{i=1}^{N} s_{1i}^2}{\sum_{i=1}^{N} (s_{1i} - s_{2i})^2} \right) \]  \hspace{1cm} (4)

where \( s_{1i} \) and \( s_{2i} \) are the \( i^{th} \) samples of the input and output signals, and \( N \) is the number of signal samples.

5) TRANSPARENCY TESTS

These experiments evaluate the transparency of the novel proposed method LSB-BMSE using objective tests with various
secret message sizes (hence block sizes) and five different cover audio from the GTZAN dataset, namely jazz, classical music, dialogue, female and vlobos [51], [52]. First, the different secret message sizes, 10KB, 20KB,...,100KB, are compressed using Huffman algorithm, then encrypted using AES-128 and finally embedded using the LSB-BMSE proposed method using the five different audios. There are several ways in representing digital sound samples in sound environments. The tests were performed in two ways to ensure the comprehensiveness and validity of the effectiveness of the proposed method; specifically the sound samples are first represented in the range $-1$ to $1$ and then in the range $0$ to $65535$. MSE, PSNR and SNR are used to justify the transparency accomplished by the proposed method. The results of the objective tests (for range 0 to 65535) are demonstrated in figures 4, 5 and 6, which show that the proposed LSB-BMSE method exhibits high fidelity. The SNR values range from 105.677dB for classical audio to 116.5612 dB for vlobos for the different secret message sizes and the different audio cover files. The PSNR gave a minimum of 3.5693E-02 dB for classical to 3.9788E-03 dB for dialogue. The transparency of the stego file generated by the proposed method is preserved since the distortion of the cover file is reduced by compressing the secret messages before embedding. Furthermore, random selection of blocks for embedding is used.

Ensue, there is no significant audible distortion produced from the embedding process that is capable to raise suspicion on the existence of the secret messages in the resulted stego signals. The PESQ score was 4.5 which attest to the efficacy of the proposed LSB-BMSE method. Using the female cover audio, our method achieved 4.5 PESQ score compared to 4.43 and 4.47 which were achieved by the conventional LSB and 4-LSB methods respectively.

Such substantial results are directly influenced by block size in the embedding process of the LSB-BMSE proposed method. The block size, which is based on equation 1 algorithm 3, is inversely proportional to secret message size. During the embedding process, when the block size is large this means small secret message size, resulting in less distortion to audio cover file and hence a higher transparency is achieved. Hence, SNR of the stego audio is directly proportional to the block size.

C. HIDING (EMBEDDING) CAPACITY (HC)

Another objective test is the hiding capacity of the proposed method LSB-BMSE, which is an essential metric for evaluating any steganography technique. Hiding capacity or payload means the percentage of the secret message size file to that of the cover file and measures by standard percentage, as calculated by equation 5 [7].

\[
\text{Hiding Capacity (HC)} = \frac{\text{Secret file size}}{\text{Cover file size}} \times 100 \quad (5)
\]

In other words, it is the largest size that can be hidden in a cover file. Specifically, in LSB algorithm, the embedding capacity is calculated as in equation 6:

\[
\text{Hiding Capacity for LSB} = \frac{\text{number of samples}}{8} \quad (6)
\]

Therefore, if a cover audio file has 8000 samples, the maximum embedding capacity using LSB method will be 8000/8, which is 1000 bytes, as presented in the 2nd column of table 2.

Table 2 compares the hiding capacity of the proposed method to the traditional LSB capacity, and achieving an increase of 72.94% in the case of 661500 samples. Thus based on these results, the compression effect has significantly improved the hiding capacity, ergo affirms the high capacity achieved by the proposed method.

1) EMBEDDING RATE

The embedding rate is the number of the secret message that can be embedded in a byte (or sample) of cover audio file. Thus, it is worth noting that the embedding rate of the proposed approach is directly proportional to the sampling rate of a cover audio file. It may also be calculated in bits per sample, which is the number of secret message bits that can be embedded in each cover audio sample. The number of cover audio sample is different between files depending on sample rate and audio duration. Thus, the cover audio capacity is one kilobyte per second (kbps) per 1 kilohertz (kHz). As an example, in a noiseless channel, the bit rate will be 8 Kbps.
TABLE 2. The hiding capacity of the proposed method.

| Number of samples (in bytes) | Hiding Capacity of traditional LSB (in bytes) | Hiding Capacity of proposed method compared to traditional LSB (in %) |
|-----------------------------|-----------------------------------------------|-------------------------------------------------------------|
| 7769                        | 971                                           | 1840 (158.6%)                                               |
| 12098                       | 1512                                          | 2490 (164.68%)                                             |
| 19912                       | 2489                                          | 4220 (169.55%)                                             |
| 47567                       | 5945                                          | 9133 (153.62%)                                             |
| 65405                       | 8175                                          | 13712 (167.73%)                                            |
| 70113                       | 8764                                          | 15049 (171.71%)                                            |
| 71352                       | 8919                                          | 15395 (172.61%)                                            |
| 89190                       | 11148                                         | 19193 (172.17%)                                            |
| 661500                      | 82 687                                        | 143000 (172.94%)                                           |

in an 8 kHz sampled sequence and 44.1 kbps in a 44.1 kHz sampled sequence.

For instance, in the traditional LSB method, 1 byte of secret message should be embedded in 8 bytes (or samples) of carrier data; thus the embedding rate is 12.5% or 0.125 bytes per sample [15].

The experiments demonstrated in figure 5 were carried out by embedding different message sizes on different cover audio files to investigate the correlation between bps and PSNR. The results showed that the PSNR will decrease if high number of bits per sample were embedded and vice versa. Thus by increasing bits per sample, i.e. the capacity is improved, nonetheless the PSNR indicating imperceptibility is decreased.

D. ROBUSTNESS ANALYSIS

Robustness reflects the ability to withstand attack. That is to say, it is the ability to recover a secret message without or with an acceptable distortion after an attack, such as AWGN attack. Hence, similarity between the recovered and original secret messages is usually used to measure the robustness.

The stego audio may be attacked intentionally to alter the secret message using one of the following approaches:

- **LSB Attack**: each of the LSB bits of a stego audio are arbitrarily modified from ‘0’ to ‘1’ or vice-versa.
- **Re-sampling Attack**: The attacker performs re-sampling, for example from 16,000 to 8,000 and then back to 16,000, and after that send the stego audio to the recipient. As a consequence, embedded secret bits at LSB bits may be altered [56].
- **AWGN attack**: White gaussian noise is added to the stego audio causing corruption.

Nonetheless, the stego audio could also be attacked unintentionally during transmission, when affected with channel noise.

1) BIT ERROR RATE (BER)

BER is a metric used to calculate the correctness of embedding; it shows the percentage of the message bits that was retrieved incorrectly, as expressed by equation 7 [57], [58]:

\[
BER = \frac{N_{Err}}{N_{Bits}}
\]

where \(N_{Err}\) is the incorrect retrieved bits, and \(N_{Bits}\) is the total number of bits embedded in the audio file.

From the transparency point of view, it describes the ratio of the modified bit number of the stego audio and the bit number of the original audio, hence better imperceptibility of the algorithm. It is calculated by equation 8 [15]:

\[
BER = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 
1, & y(i) \neq x(i) \\
0, & y(i) = x(i) 
\end{cases}
\]

where, \(x\) is the original audio, \(y\) is the stego audio, and \(N\) is the sampling point number of \(x\) and \(y\).

The BER is directly proportional to the message size, but inversely proportional to the block size. The lower the BER is, the stronger the robustness of the steganographic method will be. Figure 7 reflects the testing of BER for the proposed LSB-BMSE method, giving lowest value 0.000244 for classical with 1KB and highest value 0.002255 for vlobos with 10KB, thus demonstrating the robustness and the fidelity of the proposed method.

2) NORMALIZED CROSS-CORRELATION (NCC)

A most popular method used to measure the similarities is NCC. The formula for one dimensional message signal such as text is presented in equation 9:

\[
NC(M, M') = \frac{\sum_{k=1}^{QG} M(k)M'(k)}{\sqrt{\sum_{k=1}^{QG} M(k)^2} \sqrt{\sum_{k=1}^{QG} M'(k)^2}}
\]

where \(M\) and \(M'\) are the original and recovered secret message, respectively

\(QG\) is number of samples in each one of them.
3) ROBUSTNESS OF THE PROPOSED METHOD

Commonly, BER and NCC are used to evaluate robustness. The lower the BER towards zero, the vigorous the robustness of the steganographic method. Nonetheless, the greater the NCC is (near to 1), the stronger the robustness of the steganographic approach [59]. In the absence of an attack in the proposed method, the extracted secret is exactly the same as the original one, achieving NCC value of 1 for all tested audios. Likewise, the stego and the cover audios were indistinguishable yielding an NCC value of 1, when there is no attack. However, the NCC were less than 1 (~0.8950) for all audios when AWGN noise is added due to high distortion.

The proposed method is tested against the LSB attack, re-sampling and AWGN attack. In re-sampling attack the stego signal is down sampled to 8 KHz and then up-sampled to original sampling frequency of 16 KHz. In AWGN attack, a white gaussian noise of 0.001dB is added to the stego signal. The distortion in the revealed secret is least in case of re-sampling attack, giving a PESQ value of 4.5. However, the juxtaposition of the cover and stego audios gave favorable results of 4.5 for the PESQ for both LSB attack and the re-sampling attack affirming that the stego and cover are imperceptible and robust against these attacks.

Nonetheless, the nature of the proposed method deemed it difficult to withstand the robustness of constructing secret message under AWGN and LSB attacks.

The proposed method has innate resistance to noise and LSB attacks due to nature of the method. This is due to concerted factors: such as splitting an audio file into blocks and the BMSE mechanism of hiding secret message size and then using the random key to embed the secret message in the leftover blocks. Thus any attack affecting the samples/their number is inevitable.

E. SECURITY ANALYSIS

1) STRICT AVALANCHE CRITERION

This part of the research paper demonstrates the steps taken to experiment the avalanche criterion [60] to measure the strength of the proposed LSB-BMSE method. It is highly desirable to have an avalanche score in the range 0.5 ± ϵ. This means, if only one bit is changed in the input key, then every bit in the output key has a probability of 0.5 ± ϵ to change their value. There are two inputs, the hiding key and the secret message to be hidden. Furthermore, the embedding block positions also change accordingly. The avalanche criterion measures how much the secret message random block positions and the key itself will change if a small change is introduced to the input.

To calculate the score of the key avalanche, different secret message sizes were embedded using an initial key. Then a small change is introduced to the input.

The results of the avalanche test are calculated by determining the positions that were used to embed in the case of the original key and comparing them to the positions that were used in embedding in the case of the changed key. After the comparison, the percentage of change that occurred in the hidden locations between the two cases is calculated for each secret message size for three different bit change locations.

The avalanche score is calculated as follows:

\[
\text{Key avalanche score} = \frac{\text{number of changed bits}}{128}
\]  

(10)

The results of the embedding positions avalanche score, illustrated in table 3, range from 93.75 % to 99.22% which affirms the robustness of the proposed LSB-BMSE method. The key avalanche score also gave favorable results as shown in the last column of table 3.

| Secret message in bytes | Changed bit position in key | Percentage change in block positions | Key avalanche score |
|------------------------|-----------------------------|-------------------------------------|---------------------|
| 1000                   | First                       | 99.13                              | 48.44               |
|                        | Middle                      | 99.22                              | 50                  |
|                        | Last                        | 98.67                              | 44.35               |
| 2000                   | First                       | 97.71                              | 44.44               |
|                        | Middle                      | 97.39                              | 52.34               |
|                        | Last                        | 97.53                              | 51.56               |
| 3000                   | First                       | 96.05                              | 49.32               |
|                        | Middle                      | 96.31                              | 50                  |
|                        | Last                        | 96.12                              | 55.47               |
| 4000                   | First                       | 94.99                              | 49.22               |
|                        | Middle                      | 94.81                              | 50                  |
|                        | Last                        | 93.74                              | 55.47               |
| 5000                   | First                       | 93.75                              | 44.44               |
|                        | Middle                      | 94.15                              | 50.78               |
|                        | Last                        | 93.79                              | 50                  |

2) RESISTANCE TO BRUTE FORCE ATTACK

In fact, the security of the proposed LSB-BMSE method depends on the secret key rather than the privacy of the scheme as it complies with Kerckhoff’s principle.

The proposed method has two keys, namely key and 128-key input to AES, therefore the number of possible random numbers using these two keys is

\[
2^{128} \times 2^{128} = 2^{256} = 1.1579209 \times 10^{77}
\]

The attacker systematically tries all possible numbers till the right one is recognized. Thus, an ample time is needed to break this huge number. Moreover, the proposed method exhibits avalanche criteria as presented in the section IV-E1, and is proved highly sensitive even to small changes in the key, as well as the randomness of the novel BMSE mechanism, which is elucidated next.

3) THE RANDOMNESS OF THE BMSE MECHANISM

The Key is generated from the input key (16 bytes) in a random manner using the proposed BMSE method. This randomness property is tested using the Statistical Test Suite (STS) recommended by the NIST [61], which ensures that the output key is statistically indistinguishable from a random output. Thus the proposed BMSE method acts...
This suite has 15 different tests, where a probability (P-Value) is calculated for each. The method passes any particular test if the P-Value is in the range $0.01 \leq P \leq 1$. As per the guidelines of the NIST, the null hypothesis is that the sequence being tested is random. The tests have been carried on a significance level of 0.01, which means that the probability of rejecting the null hypothesis while it is true is 0.01 [61]. Figure 8 shows that the proposed BMSE method passed all tests successfully.

### F. Steganalysis Tests

1) **Histogram Attack**

With respect to the histogram attack, fifty experiments were conducted using different cover audios and different secret message sizes as demonstrated in figures 9 and 10. Histogram Error Rate (HER) using equation 11 is adopted to find the histogram error between the original cover and the stego audio produced by the proposed LSB-BMSE method. Figure 9 presents the HER value and the histogram of the original cover audio before and after embedding secret message size of 100KB using five different cover audios.

$$HER = \frac{\sum_{i=1}^{N} (His_c - His_s)^2}{\sum_{i=1}^{N} His_c^2}$$  \quad (11)$$

where $His_c$ and $His_s$ are histograms of cover and stego audios.

### G. Comparison with Related Schemes

In order to highlight the overall potential of the proposed method, it is juxtaposed with related schemes and compared in terms of hiding capacity (as shown in table 2 section IV-C), fidelity, PESQ, HER and robustness based on the published results from various schemes. Specifically, comparisons with other LSB schemes, using the same dataset, were performed with Ali et al. [7], Bharti et al. [14], Ahmed et al. [39], and Bazyar et al. [62].

Research [62] achieved an increase in the hiding capacity of 45.65% using jazz audio cover and an SNR 51.09 dB, albeit
our method attained an increase of 72.94% for the same audio and SNR of 101.72 dB on average.

Comparing the histogram error rate, our proposed method achieved HER of 0.00002244 which outperformed that of [7] which is 0.1278 for jazz cover, confirming that it is robust to histogram attacks.

Table 4 presents the PESQ scores for the proposed method compared to some selected related schemes, together with the randomness of the hiding process.

Table 5 compares the MSE, PSNR and SNR of the proposed method with research [7], as they used the same audio cover files. Clearly, our method’s results were beyond comparison. Blatantly, the proposed method has successfully preserved the stego audio fidelity at an approximate average of 99.98 dB for all tested audios, well above the acceptable 30 dB level affirming its superiority.

V. CONCLUSION AND FUTURE WORK

In this paper, a novel LSB-BMSE method for enhancing LSB audio steganography is presented. At first, the secret message is compressed using Huffman coding then encrypted by AES-128. Next, the audio cover is divided into blocks adaptively according to the secret message size. We designed the BMSE as an embodiment of the security boosting mechanism. It addresses the hiding of the secret message size in random blocks, then uses its output random key, key

\[ \text{BMSE} \]

, to embed the secret message in random blocks and bytes using a nondeterministic fashion. The avalanche criterion of key

\[ \text{BMSE} \]

and the percentage change in the block positions are investigated and the scores were 0.5 and 96.49% on average respectively. Above that, the key fully complies with Kerckhoff’s principle. Moreover, the innovative BMSE has passed all NIST STS randomness tests successfully.

Furthermore, the novel LSB-BMSE method was evaluated using PESQ, MSE, PSNR, and SNR imperceptibility tests and gave favorable results dominating prevailing schemes, with an average SNR of 99.98 dB, which is well above the acceptable 30 dB level. The standard PESQ scores 4.5 which affirms that the cover and stego audios were indistinguishable in the absence of attack. Above that, it achieved an increase of 72.94% in the hiding capacity compared to traditional LSB. Additionally, it proved its resistance against re-sampling attack, however it did not resist noise and LSB attacks due to its nature. The stego signal produced was analyzed against histogram distribution and attained HER as low as 2.8624E-07 for jazz audio. The quality of the revealed secret was tested using NCC and showed resemblance to the original secret message. The effective results of the novel LSB-BMSE method and the innovative BMSE mechanism gave reassurance of its efficacious and affirms its superiority to existing schemes.

APPENDIX A

IMPLEMENTATION OF THE PROPOSED METHOD

A. IMPLEMENTING THE BMSE ALGORITHM

First, we will elucidate the embedding of the BMSE using the proposed LSB-BMSE algorithm method. After encoding the BMSE, random blocks from the cover file are used to embed the BMSE using the traditional LSB technique, see algorithm 2. The following specifications are used:

- cover file (CF) size = 34 bytes (samples)
- Key 16-bits = “1011 0001 1010 1011”
- Secret message m = “0110 1101”; hence it is 1 byte and the BMSE = 0000 0001 1.

These sizes were chosen for simplicity. Moreover, the index calculation of CF is made to start from 1 because of the nature of the function, as it will not have a value of zero.
Round1: CF = 34 ⇒ BinaryOf(CF) = 100010
CO = length(CF Bits) = length(100010) = 6
Hence, A = first CO bits selected from the beginning of the key. Given key = 1011 0001 1010 1011 ⇒ A = 1011 00
G = DecimalOf(A) = 44.
S = G mod length(CF) = 44 mod 34 = 10. Store byte S in S-array. As shown in figure 11. And if G is an even number, then store next byte K in K-array. K = 190 ⇒ BinaryOf(K) = 1011 1110.

Start reading bytes of K till length of A and XOR them to get Z = (1011 11) ⊕ (1011 00) = 000011.

Divide the remaining key (01 1010 1011) into two parts and MIX them. Hence, left half of the key. Given key = MixRemainKey = 0011011111.

New key = MIX(MixRemainKey, Z)
P = Quotient(BitsOf(MixRemainKey), Z) ⇒ P = Quotient(1011 00)10/6 = 1

If BitsOf(MixRemainKey) > BitsOf(Z), then after every P bits of MixRemainKey, put 1 bit from Z, and put remaining bits at the end to get New Key = 0000101011010111.

FIGURE 11. Round 1: Embedding BMSE bits.

Round2: Now CF becomes = 32 bytes ⇒ BinaryOf(CF) = 100000
CO = length(CF Bits) = length(100000) = 6
Since the last 2 bits of last K is “10,” then select the last CO bits from the key and convert them into decimal (G):
Last K = 190 ⇒ K bits = 10111110, hence
Key = 0000101011010111 ⇒ A = 010111.

G = DecimalOf(A) = 23
S = G mod length(CF) = 23 mod 32 = 23. Extract byte number S from CF and store in S-array. If G is odd, then extract the preceding byte K and store in K-array.

Hence K = 145, ⇒ BinaryOf(K) = 10010001
Start reading bits of K till length of A and XOR them to obtain Z = (100100) ⊕ (011111) = 110111

Divide the remaining key (000010110111) into two parts and MIX them. Left half = 00001 & right half = 01011 ⇒ Mixing remaining key, K1 = 0001001111.

If (the bits of (key) > the bits of (z)), then P = Quotient(5/6 = 1)

To produce new key, MIX K1 with Z as follows: after every P bits of key K1, put one bit from Z, and if there are bits remaining from Z, place them at the end of new key.

New Key = 0101001010011111

Round3: CF = 30 ⇒ BinaryOf(CF) = (30)2 = 1110 ⇒ CO = 5

Then take 5 bits from new key as follows:
last K = 145 and K in bits = 1001001, K bits ends in “01,” then the 5 bits are selected from the middle. Let R1 = length(key) ⇒ CO (here 16-5 = 11).

If R1 even then R2 = R1/2 Selection = R2 + 1 to R2 + CO
If R1 odd then R2 = (R1 + 1)/2 (here = (11 + 1)/2 = 6)
Selection = R2 to R2 + CO-1 (=6 to (6 + 5-1) = 6 to 10) ⇒ A = 01001
G = DecimalOf(A) = 9
S = G mod length(CF) = 9 mod 30 = 9. As G is odd, then K = preceding byte number 8, hence K = 233

To calculate new key: BinaryOf(K) = 23310 = (11101001)2
K = 11101 & A = 01001 ⇒ Z = K2 XOR A = 10100
Last key (0110 0 10001 0 1011), thus after taking the A bits ⇒ remaining key = (01100010111).

If bits(remaining key) is odd, then left half = n/2 + 1 and right half = n/2. Hence, the mixing remaining key = 01100111010. As p = 11/5 = 2 ⇒ New key = 01110001111010100.
B. IMPLEMENTING THE EMBEDDING ALGORITHM

After processing algorithm 2 on the 25 bytes of the cover file, the following steps in figure 14 yields randomly the block numbers to embed the BMSE of the secret message in succession. Each step produces a new key to be input into the next step, and finally the last key produced, Key\_BMSE, is used in the embedding algorithm 3. Figure 15 shows the leftover blocks of the cover audio file and the S-array containing the BMSE bits.

Therefore, to embed the first bit “0,” having j initially 0, H = (B + 1) \times j + (2 + 1)(0) + 0 = 0, thus the first bit “0” is embedded in block number 0. To calculate the byte inside the block ⇒

\[ D_0 = \text{BinaryOf}(11) = 3 \Rightarrow F = 3 \mod B = 3 \mod 2 = 1 \]

Therefore, the first bit “0” is embedded in block number 0, inside byte number 1 using LSB technique. See algorithm 3 for the algorithm details. For the next round, D will take the value of the next 2 bits from the key. Thus, D = BinaryOf(10) = 2, and so on for the rest of the operations. We will suffice with this explanation of the proposed method implementation for the space limitation.

APPENDIX B

EXPERIMENTING THE AVALANCHE CRITERION

Table 7 gives the different input keys for experimenting the avalanche criteria. The original input key is first manipulated changing the first bit, and the avalanche score. Then a bit in the middle is changed and finally the last bit is changed.

TABLE 7. Different key inputs to avalanche criterion.

| Input key | Avalanche score |
|-----------|-----------------|
| Initial key | 0 |
| Changing 1st bit | 1 |
| Changing 2nd bit | 2 |
| Changing last bit | 3 |

Therefore, the original input key is first manipulated changing the first bit, and the avalanche score. Then a bit in the middle is changed and finally the last bit is changed.
Each time the output key is presented together with the avalanche score as presented in table 8 showing the details of the avalanche criteria on the input key.

### TABLE 8. Experimenting the avalanche criterion.

| Secret message | Output key for original input key | Message bit positions in input key | Output key after changing 1 bit in weighted key | Avalanche score |
|----------------|-----------------------------------|------------------------------------|-----------------------------------------------|-----------------|
| PIN            | Present                           | Non-present                        | Present                                       | 44.55           |
| 2400           | Partially Present                  | Partially present                  | Partially present                             | 53.64           |
| 6000           | Present                            | Present                            | Present                                       | 49.73           |
| 5000           | Partially Present                  | Partially present                  | Partially present                             | 59.47           |
| 10000          | Present                            | Present                            | Present                                       | 59.71           |
| 20000          | Partially Present                  | Partially present                  | Partially present                             | 59.71           |
| 50000          | Present                            | Present                            | Present                                       | 49.73           |

### ACKNOWLEDGMENT
(Mahmoud M. Mahmoud and Huwaida T. Elshoush contributed equally to this work.)

### REFERENCES

[1] A. Arora, M. P. Singh, P. Thakral, and N. Jarwal, “Image steganography using enhanced LSB substitution technique,” in Proc. 4th Int. Conf. Parallel Distrib. Grid Comput. (PDGC), 2016, pp. 386–389.

[2] R. Kaur, A. Thakur, H. S. Saini, and R. Kumar, “Enhanced steganographic method preserving base quality of information using LSB, parity and spread spectrum technique,” in Proc. 5th Int. Conf. Adv. Comput. Commun. Technol., Feb. 2015, pp. 148–152.

[3] A. Kumar and K. K. KM, “Enhanced LSB algorithm for steganographic communication,” J. Web Develop. Web Designing Int. J., vol. 12, no. 2, pp. 154–162, Jan. 2016.

[4] S. M. Alwahbani and H. T. Elshoush, “Chaos-based audio steganography and cryptography using LSB method with one-time pad,” in Proc. SAI Intell. Syst. Conf., Cham, Switzerland: Springer. Sep. 2016, pp. 755–768.

[5] T. K. Hazra, M. Haldar, and M. K. Mukherjee and A Chakraborty, “A survey on different techniques for covert communication using steganography,” IJORS J. Comput. Eng. (IOSR-JCE), vol. 20, no. 2, pp. 42–52, Apr. 2018.

[6] N. Kaur and A. Kaur, “Art of steganography,” Int. J. Adv. Trends Comput. Appl. (IJATCA), vol. 4, no. 2, pp. 30–33, Feb. 2017.

[7] A. H. Ali, L. E. George, A. A. Zaidan, and M. R. Mokhtar, “High capacity, transparent and secure audio steganography model based on fractal coding and chaotic map in temporal domain,” Multimedia Tools Appl., vol. 77, no. 23, pp. 31487–31516, Jun. 2018.

[8] A. H. Ali, M. R. Mokhtar, and L. E. George, “Enhancing the hiding capacity of audio steganography based on block mapping,” J. Theor. Appl. Inf. Technol., vol. 95, no. 7, pp. 1441–1448, Apr. 2017.

[9] A. A. Alsbahny, F. Ridzuan, and A. H. Azni, “The adaptive multi-level phase coding method in audio steganography,” IEEE Access, vol. 7, pp. 129291–129306, 2019.

[10] M. H. N. Azam, F. Ridzuan, M. N. S. M. Sayuti, and A. A. Alsabhany, “Balancing the trade-off between capacity and imperceptibility for least significant bit audio steganography method: A new parameter,” in Proc. IEEE Conf. Appl., Inf. Secur. (AINS), Nov. 2019, pp. 48–53.

[11] R. Chakraborty and A. Roy, “Audio steganography—A review,” Int. J. Trend Res. Develop., vol. 6, no. 3, pp. 144–149, Jun. 2019. [Online]. Available: http://www.ijtrd.com

[12] P. Bhite and M. R. Sayankar, “A review on audio and video based steganography for data hiding,” JSIRSET, vol. 4, no. 1, pp. 2394–4099, Aug. 2019.

[13] K. Bhowal, “Multilevel steganography to improve secret communication,” in Digital Image and Video Watermarking and Steganography, London, U.K.: IntechOpen, 2019.

[14] S. S. Bharti, M. Gupta, and S. Agarwal, “A novel approach for audio steganography by processing of amplitudes and signs of secret audio separately,” Multimedia Tools Appl., vol. 78, no. 16, pp. 23179–23201, Aug. 2019.

[15] G. Xin, Y. Liu, T. Yang, and Y. Cao, “An adaptive audio steganography for covert wireless communication,” Secur. Commun. Netw., vol. 2018, pp. 1–10, Aug. 2018, doi: 10.1155/2018/7096271.

[16] A. Kanhe and G. Aghila, “DCT based audio steganography in voiced and un-voiced frames,” in Proc. Int. Conf. Informat. Analytics, Aug. 2016, pp. 1–4.

[17] H. I. Shahadi, R. Jidin, and W. H. Way, “A novel and high capacity audio steganography algorithm based on adaptive data embedding positions,” J. Appl. Sci. Eng. Technol., vol. 7, no. 11, pp. 2311–2323, Mar. 2014.

[18] P. Manimengali, K. S. Gomathi, D. Ponssivel, and M. Santha, “The image steganography and steganalysis based on peak-shaped technique for Mp3 audio and video,” Int. J. Comput. Sci. Mobile Comput., vol. 3, no. 1, pp. 300–308, Jan. 2014.

[19] A. H. Ali and L. George, “A review on audio steganography techniques,” Res. J. Appl. Sci., Eng. Technol., vol. 12, no. 2, pp. 154–162, Jan. 2016, doi: 10.19026/jaset.12.2316.

[20] H. Dutta, R. K. Das, S. Nandi, and S. R. M. Prasanna, “An overview of digital audio steganography,” IETE Tech. Rev., vol. 37, no. 6, pp. 632–650, Dec. 2019.

[21] A. Chaudhuri, S. Satam, R. Sood, and D. Bade, “An efficient method for image and audio steganography using least significant bit (LSB) substitution,” Int. J. Comput. Appl., vol. 77, no. 13, pp. 37–45, Sep. 2013.

[22] A. Olawale, A. Adebayo, and G. Arome, “Embedding text in audio steganography system using advanced encryption standard, text compression and spread spectrum techniques in Mp3 and Mp4 file formats,” Int. J. Comput. Appl., vol. 171, no. 21, pp. 46–51, Mar. 2020.

[23] M. Tang, S. Zeng, X. Chen, J. Hu, and Y. Du, “An adaptive image steganography using AMBTC compression and interpolation technique,” Optik, vol. 127, no. 1, pp. 471–477, Jan. 2016.

[24] S. Ali, L. E. George, and H. B. Taher, “Speeding up audio fractal compression,” Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 3, no. 6, pp. 86–92, 2013.

[25] M. Junaid and K. Farhan, “Enhanced audio LSB steganography for secure communication,” Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 1, pp. 340–347, 2016.

[26] G. M. Kamau, “An enhanced least significant bit steganographic method for information hiding,” M.S. thesis, Dept. Softw. Eng., Jomo Kenyatta Univ. Agricult. Technol., Juja, Kenya, 2013.

[27] R. Krenn. (Jan. 2004). Steganography and Steganalysis. [Online]. Available: http://www.krenn.nl/univ/cry/steg/article.pdf

[28] F. Djebbar, B. Ayad, H. Hamam, and K. Abed-Meraim, “A view on latest audio steganography techniques,” in Proc. Int. Conf. Innov. Inf. Technol., Apr. 2011, pp. 409–414.

[29] P. G. P. Jaya S. T., B. Hidayat, and F. Y. Suratman, “Enhanced LSB steganography with people detection as stego key generator,” in Proc. Int. Conf. Signals Syst. (IC3SIGSys), May 2017, pp. 99–104.

[30] H. Ghazemzadeh and M. K. Azmi, “Universal audio steganalysis based on calibration and reversed frequency resolution of human auditory system,” IET Signal Process., vol. 11, no. 8, pp. 916–922, Oct. 2017.

[31] M. L. M. Kiah, B. B. Zaidan, A. A. Zaidan, A. M. Ahmed, and S. H. Al-Bakri, “A review of audio based steganography and digital watermarking,” Int. J. Phys. Sci., vol. 6, no. 16, pp. 3837–3850, 2011.

[32] J. Kour and D. Verma, “Steganography techniques—A review paper,” Int. J. Emerg. Res. Manage. Technol., vol. 3, no. 5, pp. 2278, May 2014.
X. Yi, K. Yang, X. Zhao, Y. Wang, and H. Yu, “AHCM: Adaptive Huffman code mapping for audio steganography based on psychoacoustic model,” IEEE Trans. Inf. Forensics Security, vol. 14, no. 8, pp. 2217–2231, Aug. 2019.

A. A. Alsahbany, F. Ridzuan, and A. H. Azni, “The progressive multi-level embedding method for audio steganography,” J. Phys., Conf. Ser., vol. 1551, no. 1, May 2020, Art. no. 012011.

M. M. Mahmoud and H. T. Elshouss, “A novel enhanced LSB algorithm for high secure audio steganography,” in Proc. IEEE CECCE, Sep. 2018, pp. 125–130.

R. Tanwar, K. Singh, M. Zaman, A. Verma, and P. Kumar, “An optimized approach for secure data transmission using spread spectrum audio steganography, chaos theory, and social impact theory optimizer,” J. Comput. Netw. Commun., vol. 2019, pp. 1–10, Sep. 2019, doi: 10.1155/2019/5124364.

M. A. Ahmed, M. L. M. Kiah, B. B. Zaidan, and A. A. Zaidan, “A novel embedding method to increase capacity and robustness of low-bit encoding audio steganography technique using noise gate software logic algorithm,” J. Appl. Sci., vol. 10, no. 1, pp. 59–64, 2010.

S. Roy, A. K. Singh, J. Parida, and A. S. Sairam, “Audio steganography using LSB encoding technique with increased capacity and bit error rate optimization,” in Proc. CSCEST-12, Oct. 2012, pp. 372–376.

S. Krishnan and M. S. Abdullah, “Enhanced security audio steganography by using higher least significant bit,” J. Adv. Res. Comput. Appl., vol. 2, no. 1, pp. 39–54, 2016.

S. Save, P. Raut, P. Jadhav, and T. Yadav, “Data security using audio-video steganography,” Int. J. Eng. Res., vol. V7, no. 2, pp. 276–279, Feb. 2018.

S. Gudla, S. Reyya, A. Kotyada, and A. Sangam, “Key based least significant bit (LSB) insertion for audio and video steganography,” Int. J. Comput. Sci. Eng. Res. Develop., vol. 3, no. 1, pp. 60–69, Mar. 2013.

C. T. Jian, C. C. Wen, N. H. Binti Ab. Rahman, and I. R. B. A. Hamid, “Audio steganography with embedded text,” IOP Conf. Ser., Mater. Sci. Eng., vol. 226, Apr. 17, 2017, Art. no. 012084.

S. M. Alwahbani and H. T. Elshouss, “Hybrid audio steganography and cryptography method based on high least significant bit (LSB) layers and one-time pad—A novel approach,” in Dr. Yasin Bi Editor: Studies in Computational Studies, vol. 751, Cham, Switzerland: Springer, Jan. 2018, pp. 431–453.

H. Kumar and A. Taluja, “Enhanced LSB technique for audio steganography,” in Proc. 3rd Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT), Jul. 2012, pp. 1–4.

K. Sharma and R. Thakur, “LSB modification based audio steganography using trusted third party key indexing method,” in Proc. 3rd Int. Conf. Image Inf. Process. (ICIIP), Dec. 2015, pp. 403–406, doi: 10.1109/ICIIP.2015.7414805.

M. H. A. Al-Hooti, T. Ahmad, and S. Djanali, “Audio data hiding using octal moduli function based unsigned integer sample values,” in Proc. Int. Conf. Comput. Eng., Netw. Intell. Multimedia (CENIM), Nov. 2018, pp. 48–53.

F. A. P. Petticolas, “Kerczhoﬀ’s Princ’ple,” in Encyclopedia Cryptography Security, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA, USA: Springer, 2011, p. 675.

A. Kerczhoﬀ, “La cryptographie militaire (military cryptography),” (in French), J. Sci. Militaires, 1883.

G. Tzanetakis. (Nov. 22, 2015). “Music Analysis, Retrieval and Synthesis for Audio Signals (Marsyas),” [Online]. Available: http://marsyasweb. appspot.com/download/data-sets/

G. Tzanetakis and P. Cook, “Musical genre classiﬁcation of audio signals,” IEEE Trans. Audio, Speech, and Language Process., vol. 10, no. 5, pp. 293–302, Jul. 2002.

R. D. Ra and D. Zaghaachtzi, “Ideal sampling rate to reduce distortion in audio steganography,” Proc. Comput. Sci., vol. 85, pp. 418–424, Jul. 2016.

F. Djebbar, B. Ayad, K. A. Meramim, and H. Hamam, “Comparative study of digital audio steganography techniques,” EURASIP J. Audio, Speech, Music Process., vol. 2012, no. 1, p. 25, Dec. 2012. [Online]. Available: http://asmp.eurasipjournals.com/content/2012/1/25

J. Chaharlang, M. Mosleh, and S. Rasouli-Heikalabad, “A novel quantum steganography-geostudy system for audio signals,” Multimedia Tools Appl., vol. 79, nos. 25–26, pp. 17551–17577, Feb. 2020.

J. Chaharlang, M. Mosleh, and S. R. Heikalabad, “A novel quantum audio steganography—geostudy approach using LSFPQ-based embedding and QKNN-based classiﬁer,” Circuits, Syst., Signal Process., vol. 39, no. 9, pp. 3925–3957, Jan. 2020.

P. Xue, H. Liu, J. Hu, and R. Hu, “A multi-layer steganographic method based on audio time domain segmented and network steganography,” in Proc. AIP Conf., 2018, Art. no. 020046.

A. A. Zakaria, N. A. M. Yusof, W. Z. Omar, N. A. N. Abdullah, and H. A. Rani, “Analysis of steganography substitution system methods using new testing techniques proposed for strict avalanche criterion,” Int. J. Cryptol. Res., vol. 5, no. 1, pp. 61–76, 2015.

A. Alftigan, S. Hasan, B. Barry, S. Naseredden, M. A. Elsadig, and H. T. Elshouss, “A polymorphic advanced encryption standard—A novel approach,” IEEE Access, vol. 9, pp. 20191–20207, 2021.

M. Buzayr and R. Sridarman, “A new method to increase the capacity of audio steganography based on the LSB algorithm,” J. Technol., vol. 74, no. 6, pp. 49–53, Apr. 2015.

K. N. Vhatkar and G. P. Bhole, “Particle swarm optimisation with grey wolf optimisation for optimal container resource allocation in cloud,” IET Netw., vol. 9, no. 4, pp. 189–199, Jul. 2020.

K. Manjunath, G. N. K. Ramaiah, and D. M. N. G. Rasad, “An eﬃcient audio steganography technique using hybridization of compression and cryptography algorithm,” J. Adv. Res. Dyn. Control Syst., vol. 11, pp. 132–147, Oct. 2019.

K. Manjunath, G. N. K. Ramaiah, and M. N. G. Prasad, “Backward movement oriented shark smell optimization-based audio steganography using encryption and compression strategies,” Digital Signal Processing, Vol. 311, Apr. 2022, Art. no. 103335, doi: 10.1016/j.dsp.2021.103335.