IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 23, 2022, accepted February 8, 2022, date of publication February 28, 2022, date of current version March 22, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3155146

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)

Mahmoud M. Mahmoud and Huwaida T. Elshoush contributed equally to this work.

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].

The associate editor coordinating the review of this manuscript and approving it for publication was Junaid Arshad.

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 (4^{th} to 8^{th} 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 8^{th} 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 1^{st} to 4^{th} LSB, one at a time) and signs (in 5^{th} to 8^{th} 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 11^{th} 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 used to embed the encrypted message in randomly selected audio samples. The use of chaotic map for generating the keys gave their method more strength. The same authors extended their research in a Springer chapter in 2018 [45]. In 2018, A. H. Ali *et al.* [7] proposed an audio steganography model based on fractal coding and chaotic LSB. Their method uses fractal coding to enhance the hiding capacity to 30% and moreover maintain the transparency of the audio cover. This model is secure against brute-force attack and statistical analysis as they embed in a nondeterministic way. Their method is an extension to [8]. Research [38] also used chaotic theory to make the embedding truly random. Capacity is increased by spreading a greater number of bits over the entire spectrum of an audio cover. Moreover, robustness and imperceptibility is also attained to satisfaction.

Moreover, most methods lack adaptivity and dynamic allocation. Research [15], working in a non-deterministic LSB way, proposed an adaptive audio steganography based on interval and variable low bit coding, which can be applied to covert wireless communication. The interval for embedding secret messages into the audio and the threshold in variable low bit coding are used for selecting the embedding location and embedding bits adaptively; thus the embedding capacity and embedding rate are variable. Experimental results demonstrate better performance in embedding rate and invisibility. Also [9] worked adaptively proposing a method called Adaptive Multi-level Phase Coding that achieved the highest robustness levels against AWGN and lossy compression signal attacks, and better transparency, capacity and high embedding rates in the phase coding transform domain. Research [17] proposed an adaptive high capacity audio steganography in transform domain.

Some researchers [46], [47] provide high imperceptibility and robustness. In particular, Harish *et al.* [46] enhanced the LSB algorithm by hiding using Fibonacci series starting with initial index, and the index value of next audio sample is the summation of current index and previous one. Their method is robust but the hiding capacity is reduced. Sharma *et al.* [47], on the other hand, proposed a robust 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*.

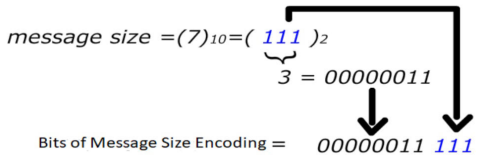


FIGURE 1. Example of encoding binaries of messages size 7B.

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 $(111)_2$. 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

```
Function CalculatingBMSE (MsgSize) :
    c = BinaryOf(MsgSize) // Convert MsgSize to bits
    l = |c|               // l is the length of c
    BMSE = BinaryOf(l) || BinaryOf(MsgSize)  // || is concatenation. Note that BinaryOf(l) has to be in 8 bits
    N_BMSE = |BMSE|
    Return N_BMSE, BMSE
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 $Key_1, Key_2, \ldots, Key_{128}$ bits
// 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: $AF_{leftover}$, Key_{BMSE}, $\{S-array\}$, $\{K-array\}$, $\{MsgSize\}$ // $AF_{leftover}$ = 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**
// MsgSize (Msg size in bytes) is an optional parameter output only when Status is **extract**

Function EmbedExtractBMSE (AF, Key, N, $Status$, $\{BMSE\}$) **:**
  $AF = AF_1, AF_2, \ldots, AF_{LengthOfAF}$ // LengthOfAF is the number of bytes of AF
  Initialize S-array and K-array to 0
  for i = 1 to N **do**
    // Repeat FOR loop until *N* BMSE bits are hidden or extracted
    CO = |BinaryOf(LengthOf(AF))| // Convert no. of bytes of AF to binary & find its length
    if (S-array is empty) OR (last two bits of BinaryOf(K) = "11") **then**
      A = $Key_1, Key_2, \ldots, Key_{CO}$ // A equals the first CO bits of the key
    else if last two bits of BinaryOf(K) = "00" or "01" **then**
      A = Middle CO bits from Key
    else if last two bits of BinaryOf(K) = "10" **then**
      A = Last CO bits from Key
    end
    G = DecimalOf(A)
    S = G mod (LengthOf(AF)) ⇒ $S\text{-array}_i = AF_S$ // read byte no. S from AF and store in $S\text{-array}_i$
    if *Status* == Embed **then**
      Embed $BMSE_i$ in $S\text{-array}_i$ // **Embedding** bit $BMSE(M_i)$ in $S\text{-array}_i$ using **LSB**
    else
      Extract $BMSE_i$ from $S\text{-array}_i$ // **Extracting** bit $BMSE(M_i)$ from $S\text{-array}_i$
    end
    if G is even **then**
      $K\text{-array}_i = AF_{S+2}$ // read byte AF_{S+2} and store in $K\text{-array}_i$
    else
      $K\text{-array}_i = AF_{S-1}$ // read the preceding byte AF_{S-1} and store in $K\text{-array}_i$
    end
    K = ByteValueOf($K\text{-array}_i$) // store the byte value of $K\text{-array}_i$ in K
    ≪ AF bytes two places and arrange $AF_1, AF_2, \ldots, AF_{(LengthOfAF)}$ // shift left AF 2 places
    LengthOf(AF) = LengthOf(AF) - 2 // One byte is used for $S\text{-array}_i$ and another for $K\text{-array}_i$
    L = BinaryOf(K) // K has n bits
    if LengthOf(BinaryOf(K)) < LengthOf(A) **then**
      $L_A = K_1, K_2, \ldots, K_n, K_1, K_2, \ldots, K_m$ // Repeat K bits until L_A = the length of A (=m + n)
    else
      $L_A = K_1, K_2, \ldots, K_A$ // Start reading bits of L till length of A
    end
    $Z = L_A \oplus A$ // L_A XOR it with A
    MixRemainKey = MIX(RemainKey$_{left}$, RemainKey$_{right}$) // Split remainingOf(key), after taken away A bits, into two halfs and MIX their bits
    if BitsOf(MixRemainkey) > BitsOf(Z) **then**
      P = Quotient($\frac{BitsOf(MixRemainkey)}{BitsOf(Z)}$)
      Key = MIX(MixRemainKey,Z) // after every P bits of MixRemainKey, put 1 bit from Z
    else
      P = Quotient($\frac{BitsOf(Z)}{BitsOf(MixRemainkey)}$)
      Key = MIX(Z,MixRemainKey) // after every P bits of Z, put 1 bit from MixRemainKey
    end
  end
  Return $AF_{leftover}$, Key_{BMSE}, $\{S-array\}$, $\{K-array\}$, $\{MsgSize\}$ // optional outputs *S-array* and *K-array* are only when Status is **embed**, while *MsgSize* is when Status is **extract**
End Function

Algorithm 3: The Embedding/Extracting Secret Message Algorithm

Input: $AF_{leftover}$, Key_{BMSE}, *Status*, N, $\{Msg\}$, $\{S-array\}$, $\{K-array\}$
// $AF_{leftover}$ is the leftover blocks from audio file AF
// Key_{BMSE} 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 (=$m_1, m_2, \ldots, m_{NMsg}$ 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 $AF_{leftover}$, S-array and K-array in their original positions in CF to form stego file

1 **Function** EmbedExtract ($AF_{leftover}$, Key_{BMSE}, *Status*, N, $\{Msg\}$, $\{S-array\}$, $\{K-array\}$) **:**

2

$$B = \frac{LengthOf\,(AF_{Leftover})}{N} \tag{1}$$

// B = block size, which is the number of bytes of $AF_{leftover}$ divided by N

3 $AF_{leftover} = AF_{block1}, AF_{block2}, \ldots AF_{blockN}$ // Split the $AF_{leftover}$ into N blocks and arrange them. Note: number of blocks = number of bits in secret Msg; and size of each block = B bytes

4 Split the $AF_{leftover}$ blocks into B bytes // Each bit of Msg is hidden in (or extracted from) one byte of a block of $AF_{leftover}$

5 V = |BinaryOf(B)| // V is how many bits from the key are considered in each step

6 Arrange Key_{BMSE} from 1 to EndOf(Key_{BMSE}) in V's array // BinaryOf(Key_{BMSE}) (=bits of hiding secret Key_{BMSE}) are split into V's bits & put in an array of $V_1, V_2, \ldots,$ EndOf(Key_{BMSE})

7 x := 0

8 j := 0

9 k := 0

10 **for** i = 1 to N **do**

11 H = (B + 1) * x + j // Calculate block no. H to hide or extract bit m_x of Msg

12 **if** H > N **then**

13 **if** Status == Embed **then**

14 $\ll m_x$ x places // '$\ll$' means shift the bits of the secret message x places

15 **end**

16 x := 0

17 j := j + 1

18 goto step 11

19 **end**

20

21 **if** k > endof(Key_{BMSE}) **then**

22 k := 0 // if there are still more bits of the secret message to be hidden, then start the Key_{BMSE} selection bits from the beginning

23 **end**

24 $W = V_k$ // is the binary value of array V_x from key_{BMSE}

25 D_x = DecimalOf(W) // Decimal representation of binary W

26 F = D_x mod B // Compute byte no. F inside block H to hide/extract m_x of Msg

27 **if** *Status* == Embed **then**

28 **Embed** bit m_x inside the block H, in byte F // Embedding inside a byte is done using **LSB**

29 **else**

30 **Extract** bit m_x from block H, in byte F and save in array T_i // m_x will be read from SF & saved in array T_i

31 **end**

32 x := x + 1

33 k := k + 1

34 **end**

35 **if** Status == Embed **then**

36 **Return** Stego file = $AF_{leftover}$, S-array and K-array in their original positions in AF

37 **else if** Status == Extract **then**

38 **Return** Secret message Msg = $T_1, T_2, \ldots, T_{NMsg}$

39 **end**

40 **End Function**

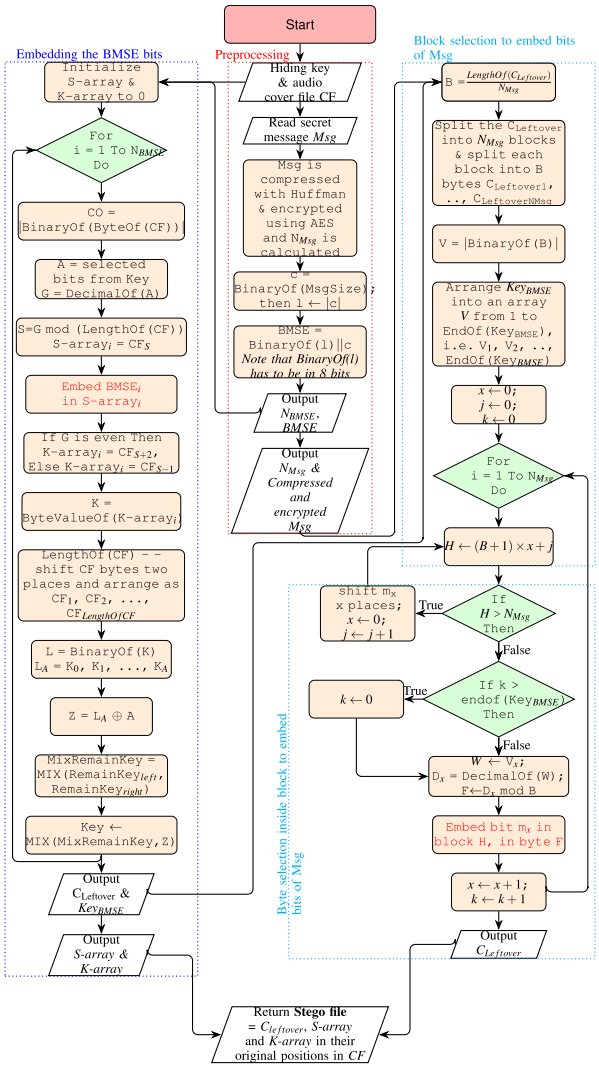


FIGURE 2. The novel proposed LSB-BMSE embedding algorithm flowchart.

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

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($CF_{leftover}$, Key_{BMSE}, Embed, N_{Msg}, Msg, $S-array$, $K-array$)* `//` **`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}, Status, N, {Msg}, {S-array}, {K-array})* of algorithm 3.

$\text{AF}_{leftover}$ 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 $\text{AF}_{leftover}$.

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.

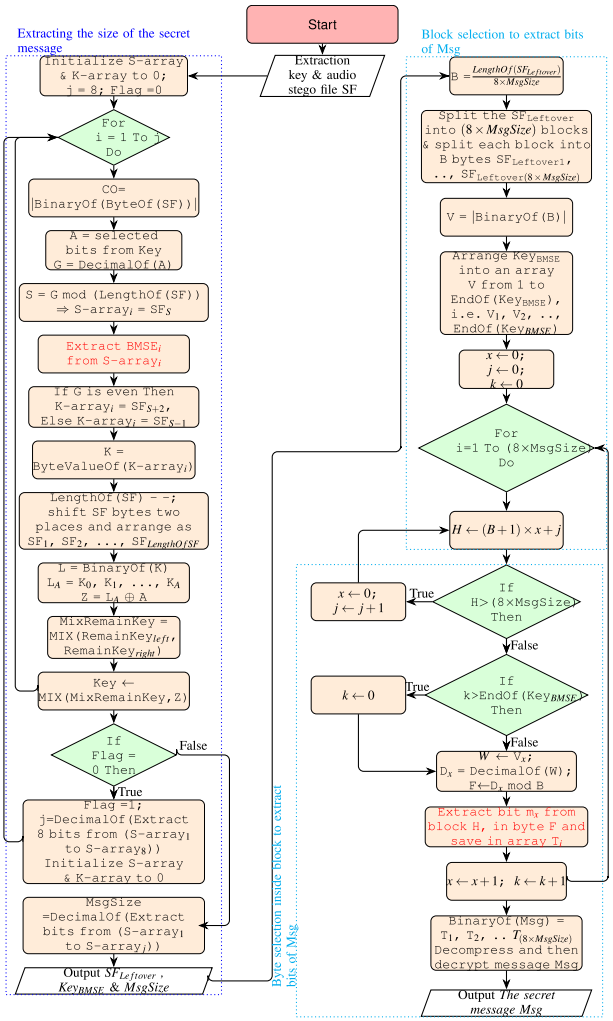


FIGURE 3. The novel proposed LSB-BMSE extraction algorithm flowchart.

Algorithm 5: The Novel Proposed LSB-BMSE Extraction Algorithm

Input: SF: audio stego file using the enhanced proposed LSB-BMSE algorithm
Key: Extraction key 128 bits // is the same hiding key
Output: Msg: The compressed encrypted hidden secret message Msg

1 *EmbedExtractBMSE(SF, Key, 8, Extract)* // **Algorithm 2**
2 TempAF ← $AF_{leftover}$ // Output $AF_{leftover}$ from **Algorithm 2**
3 $TempKey_{BMSE}$ ← Key_{BMSE} // Output KeyBMSE from **Algorithm 2**
4 Q_{bits} = Extract(S-$array_1$, . . . , S-$array_8$) // Extract 8 bits from ($S\text{-}array_1$ to $S\text{-}array_8$) and save in Q_{bits}
5 Q = DecimalOf(Q_{bits}) // Convert the bits of Q_{bits} to decimal to know the size of the secret Msg (=Q)
6 *EmbedExtractBMSE(TempAF, $TempKey_{BMSE}$, Extract, Q)* // Output $AF_{leftover}$ and Key_{BMSE} are from **Algorithm 2**
7 R = Extract(S-$array_1$, . . . , S-$array_Q$)
8 MsgSize = DecimalOf(R) // MsgSize is the size of the secret hidden message in bytes
9 *EmbedExtract($AF_{leftover}$, Key_{BMSE}, Extract, 8 × (MsgSize))* // **Algorithm 3**
10 *Decompress(Msg)* // The Msg is decompressed using Huffman Algorithm
11 *Derypt(Msg)* // The Msg is decrypted using AES-128 Algorithm
12 **Return** Msg // Secret message Msg = Extracted bits from array T_i

B. EXTRACTION PROCESS USING THE PROPOSED ENHANCED LSB-BMSE ALGORITHM

The following subsections details 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, Key_{BMSE}. 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($AF_{leftover}$, Key_{BMSE}, 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.

TABLE 1. Cover audio files specification.

Specification	
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 $\geq$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]:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(s1_i - s2_i)^2 \tag{2}$$

where $s1_i$ and $s2_i$ 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]:

$$PSNR = 10\,log_{10}\frac{(2^n - 1)^2}{MSE} \tag{3}$$

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]:

$$SNR = 10\,log_{10}\frac{\sum_{i=1}^{N}(s1_i)^2}{\sum_{i=1}^{N}(s1_i - s2_i)^2} \tag{4}$$

where $s1_i$ and $s2_i$ are the i^{th} samples of the input and output signals, and N is the number of signal samples.

FIGURE 4. Effect of the MSE versus hiding capacity on the fidelity using several secret message sizes and cover audios.

5) TRANSPARENCY TESTS

These experiments evaluate the transparency of the novel proposed method LSB-BMSE using objective tests with various

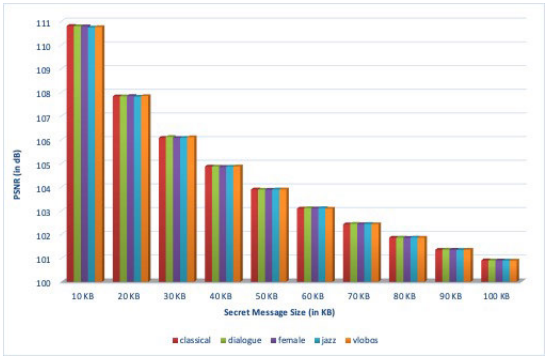


FIGURE 5. Effect of the PSNR versus hiding capacity on the fidelity using several secret message sizes and cover audios.

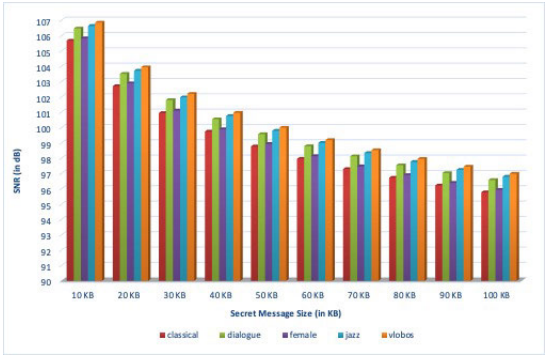


FIGURE 6. Effect of the SNR versus hiding capacity on the fidelity using several secret message sizes and cover audios.

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 110.7568 dB for vlobos and up to 120.4694 dB for female while the MSE values range from 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].

$$Hiding\ Capacity\ (HC) = \frac{Secret\ file\ size}{Cover\ file\ size} \times 100 \qquad (5)$$

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:

$$Hiding\ Capacity\ for\ LSB = \frac{number\ of\ samples}{8} \qquad (6)$$

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 2^{nd} 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	Hiding Capacity of traditional LSB (in bytes)	Hiding Capacity of proposed method (in bytes)	Hiding Capacity of proposed method compared to traditional LSB (in %)
7769	971	1540	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]:

$$BER = \frac{N_{Err}}{N_{Bits}} \tag{7}$$

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 lower BER means better imperceptibility of the algorithm. It is calculated by equation 8 [15]:

$$BER = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1, & \text{y(i)} \neq \text{x(i)} \\ 0, & \text{y(i)} = \text{x(i)}\end{cases} \tag{8}$$

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.

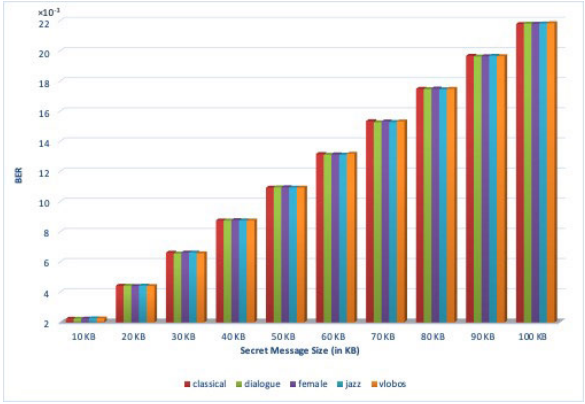


FIGURE 7. BER versus hiding capacity using several secret message sizes and cover audios.

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:

$$NC(M, M') = \frac{\sum_{k=1}^{QG} M(k)M'(k)}{\sqrt{\sum_{k=1}^{QG} M(k)^2}\sqrt{\sum_{k=1}^{QG} M'(k)^2}} \tag{9}$$

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 ($\approx$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 resampling 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$_{BMSE}$ 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 $0.5 \pm \epsilon$. This means, if only one bit is changed in the input key, then every bit in the output key$_{BMSE}$ has a probability of $0.5 \pm \epsilon$ 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 value of only one bit in the key is changed. Each time the bit of the key to be changed differs for each specific secret message size; one time the first bit is changed, secondly the middle bit is chosen and finally the last bit is changed.

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:

$$Key\ avalanche\ score = \frac{number\ of\ changed\ bits}{128} \quad (10)$$

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.

TABLE 3. Experimenting the avalanche criterion.

Secret message in bytes	Changed bit position in key	Percentage change in block positions	Key avalanche score
1000	First	99.15	48.44
	Middle	99.22	50
	Last	98.67	44.53
2000	First	97.71	48.44
	Middle	97.39	52.34
	Last	97.53	51.56
3000	First	96.05	49.22
	Middle	96.31	50
	Last	96.12	55.47
4000	First	94.99	49.22
	Middle	94.81	50
	Last	95.74	55.47
5000	First	93.75	48.44
	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$_{BMSE}$ 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$_{BMSE}$ 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$_{BMSE}$ is statistically indistinguishable from a random output. Thus the proposed BMSE method acts

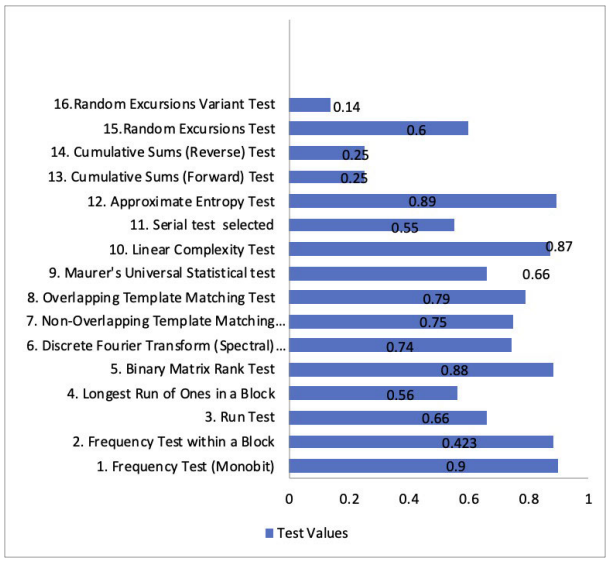


FIGURE 8. Examining the randomness of the output key$_{BMSE}$ using the NIST statistical test suite.

as a Pseudo-Random Number Generator (PRGN). Furthermore, the embedding block positions also change accordingly as it depends on the hiding key and the secret message.

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.

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

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

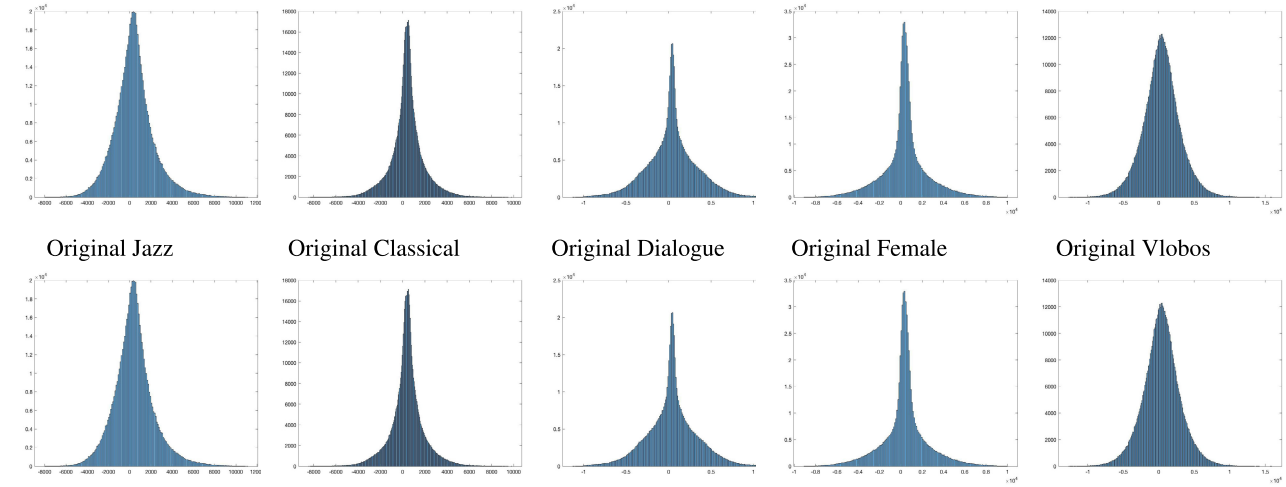


FIGURE 9. Histogram error rates for different cover audios and their corresponding stego audios using 100KB secret message.

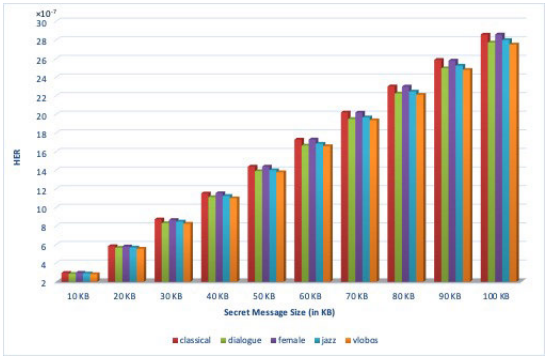


FIGURE 10. Histogram Error Rate (HER) versus hiding capacity using several secret message sizes and cover audio types.

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 0.000002244 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.

TABLE 4. Comparison of the $PESQ_{rsec\&sec}$ values for re-sampling attacks and the randomness of the hiding process with related schemes.

Method	$PESQ_{rsec\&sec}$ for re-sampling attack	Randomness of hiding process
Traditional LSB	1.20	deterministic
4-LSB	0.67	deterministic
Ahmed et al (2010) [39]	not robust	deterministic
Bharti et al (2019) [14]	1.42	non-deterministic
Proposed method	4.5	non-deterministic

TABLE 5. Comparison with related schemes of the fidelity of the stego files with different audios.

Cover file	Ali et al (2018) [7]			Proposed Method		
	MSE	PSNR	SNR	MSE	PSNR	SNR
Dialogue	0.46	99.6	73.6	0.279	101.87	97.54
Female	0.46	99.6	69.1	0.28	101.86	96.9
Jazz	0.47	99.5	69	0.279	101.87	97.76
Vlobos	-	-	71	0.279	101.86	97.96

TABLE 6. Distortion between stego and original audio for proposed and conventional related Audio Steganography schemes.

Method	MSE	SNR (dB)
Vhatkar et al (2020) [63]	0.27645	28.402
Manjunath et al (2020) [64]	0.47686	25.767
Manjunath et al (2021) [65]	0.16641	29.202
Proposed method LSB-BMSE	0.01995	99.98

The distortion between stego and original audio for proposed and related audio steganography schemes are displayed in table 6. The proposed method LSB-BMSE outstandingly outperforms the prevailing schemes.

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_{BMSE}, to embed the secret message in random blocks and bytes using a nondeterministic fashion. The avalanche criterion of key_{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 bits 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 = 01101 & right half = 01011 ⇒ MixRemainKey = 0011100111

New key = MIX(MixRemainKey, Z)

P = Quotient($\frac{BitsOf\,(MixRemainKey)}{BitsOf\,(Z)}$) ⇒ P = Quotient ($\frac{10}{6}$) = 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.

CF

Byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Byte Value	178	208	58	96	79	116	109	233	44	155	212	190	35	82	244	19	69
Byte no.	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Byte Value	141	164	135	225	48	67	145	178	207	139	55	240	198	119	150	245	77

S-array

Byte index	10
Byte value	155

$C_{Leftover}$ of CF

Byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Byte Value	178	208	58	96	79	116	109	233	44	212	190	35	82	244	19	69
Byte no.	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Byte Value	141	164	135	225	48	67	145	178	207	139	55	240	198	119	150	245	77

k-array

Byte index	11
Byte value	190

New Key= 0000101011010111

FIGURE 11. Round 1: Embedding BMSE bits.

$C_{Leftover}$ of CF

Byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Byte Value	178	208	58	96	79	116	109	233	44	212	35	82	244	19	69
Byte no.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Byte Value	141	164	135	225	48	67	145	178	207	139	55	240	198	119	150	245	77

S-array

Byte index	10	23
Byte value	155	178

$C_{Leftover}$ of CF

Byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Byte Value	178	208	58	96	79	116	109	233	44	212	35	82	244	19	69
Byte no.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Byte Value	141	164	135	225	48	67	145	207	139	55	240	198	119	150	245	77

k-array

Byte index	11	22
Byte value	190	145

New Key= 0101001001010111

FIGURE 12. Round 2: 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) ⊕ (010111) = 110011

Divide the remaining key (0000101011) into two parts and MIX them. Left half = 00001 & right half = 01011

⇒ Mixing remaining key, K_1 = 0001000111

If (the bits of (key) > the bits of (z)), then P = Quotient($\frac{the\ number\ of\ key\ bits}{the\ number\ of\ Z\ bits}$) ⇒ P = Quotient($\frac{10}{6}$) = 1

To produce **new key**, MIX K_1 with Z as follows: after every P bits of key K_1, put one bit from Z, and if there are bits remaining from Z, place them at the end of new key.

New Key = 0101001001010111

Round3: CF = 30 ⇒ BinaryOf(CF) = $(30)_2$ = 11110 ⇒ CO = 5

Then take 5 bits from new key as follows:

last K = 145 and K in bits = 10010001, 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) = 233_{10} = $(11101001)_2$

K = 11101 & A = 01001 ⇒ Z = K2 XOR A = 10100

Last key (0101 0 01001 0 1 0111), thus after taking the A bits ⇒ remaining key = (01010010111)

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* = 0111000111100100.

$C_{Leftover}$ of CF

Byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Byte Value	178	208	58	96	79	116	109	233	44	212	35	82	244	19	69
Byte no.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Byte Value	141	164	135	225	48	67	207	139	55	240	198	119	150	245	77

S-array

Byte index	10	23	9
Byte value	155	178	44

$C_{Leftover}$ of CF

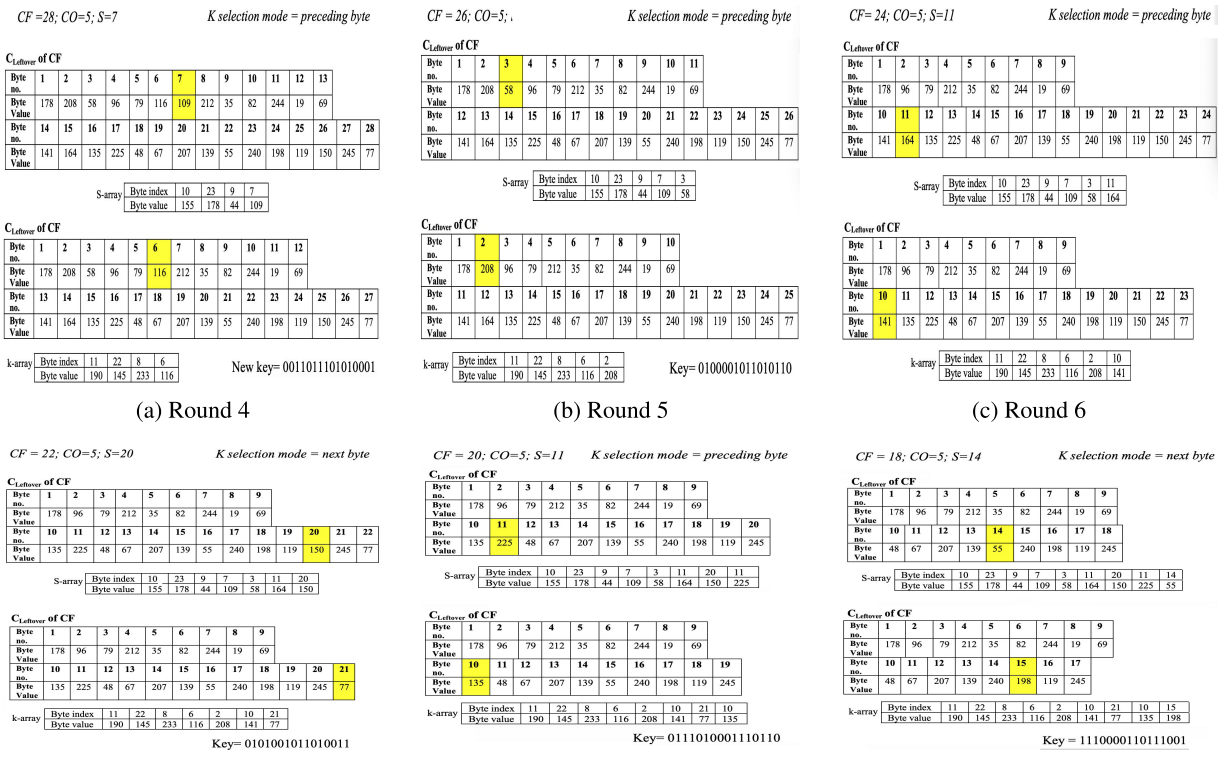
Byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Byte Value	178	208	58	96	79	116	109	233	212	35	82	244	19	69
Byte no.	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Byte Value	141	164	135	225	48	67	207	139	55	240	198	119	150	245	77

k-array

Byte index	11	22	8
Byte value	190	145	233

New key = 0111000111100100

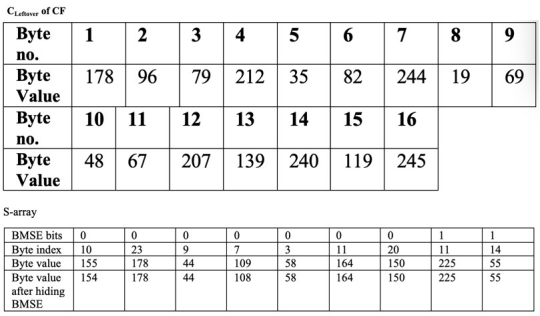
FIGURE 13. Round 3: Embedding BMSE bits.



(a) Round 4 (b) Round 5 (c) Round 6

(d) Round 7 (e) Round 8 (f) Round 9

FIGURE 14. Steps of embedding BMSE example.



Byte no.	1	2	3	4	5	6	7	8	9
Byte Value	178	96	79	212	35	82	244	19	69
Byte no.	10	11	12	13	14	15	16		
Byte Value	48	67	207	139	240	119	245		

S-array

BMSE bits	0	0	0	0	0	0	0	1	1
Byte index	10	23	9	7	3	11	20	11	14
Byte value	155	178	44	109	58	164	150	225	55
Byte value after hiding BMSE	154	178	44	108	58	164	150	225	55

FIGURE 15. Leftover from CF after embedding BMSE bits.

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.

$\text{Key}_{BMSE} = 1110\ 0001\ 1011\ 1001$

Using equation 1, $B = \frac{Length(AF_{leftover})}{N} \Rightarrow B = \frac{16}{8} = 2$

$V = |\text{BinaryOf(B)}| = 2$, Hence 2 bits from the key are considered in each step.

Recall that the secret message m = "0110 1101."

Therefore, to embed the first bit "0," having j initially 0,

$H = (B + 1) * x + j = (2 + 1)*(0) + 0 = 0$, thus the first bit "0" is embedded in block number 0. To calculate the byte inside the block $\Rightarrow$

$D_0 = \text{BinaryOf}(11) = 3 \Rightarrow F = 3 \bmod B = 3 \bmod 2 = 1$

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, $D = \text{BinaryOf}(10) = 2$, 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.

Original input key	0110110101100000101101000011011010111111011010101100100001000000 0110110101111010101110011011101000110000101100110011000010010000
Input key after changing *first* bit	1110110101100000101101000011011010111111011010101100100001000000 0110110101111010101110011011101000110000101100110011000010010000
Input key after changing *middle* bit	0110110101100000101101000011011010111111011010101100100001000000 0110111101111010101110011011101000110000101100110011000010010000
Input key after changing *last* bit	0110110101100000101101000011011010111111011010101100100001000000 0110110101111010101110011011101000110000101100110011000010010001

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 size in bytes	Output key for original input key	Changed bit position in input key	Output key after changing 1 bit in original key	Key avalanche score
1000	10011011111110010101111110110100 01001011101000000010111111011111 11000111100011001011101110111101 01011001111010100011110011010010	First	10101010001010010111010011011011 00100001011110000101011010011010 00000100110010000000100010000111 0 00111111011111001010011011111000	48.44
		Middle	00100010111011001001101111110001 01010001001011011001100111111011 00000000010101100111101110011111 11001000001111011100000110001101	50
		Last	00100100011010110111011100001000 11000111000011001011000111010101 11000001111000001001001001001101 00101001101101101011111010100011	44.53
2000	10000011110000111001111011101010 10101001101011101011100111000111 10000110010101110011001101010101 11011010100000001001000111000111 0	First	10001010010100100000000111000100 01000100010011110011000000111110 111111011101101101111010000001100 01101101111000010010111001100010	48.44
		Middle	00011100101100100011111111001011 10110010011011111101101111011011 11000000100100011101111000010111 1 00001000010011011101000001111001	52.34
		Last	00011110100011000110011011011000 01011000111111100110110100000001 01010011000010110000010011011111 00010100011010010001001111100101	51.56
3000	10010110100100101011111101000000 01001111000001101011100011111110 01011010100010101100011000100110 10011110110011110101100011101000	First	01011001111000010011101100011100 00000010010010101000000111101011 01100111110101100000000011110101 11100011111010101011100010101100	49.22
		Middle	01000000101111101000110111111000 0 01010001010111111111110101001001 01010001101110000110110001111010 00111100010110101011111001101001	50
		Last	00010001010010100101001000011101 10010001111100010010111001111100 10010000000101010010111000101011 11111101010011100010010010011110	55.47
4000	10010110100100101011111101000000 01001111000001101011100011111110 01011010100010101100011000100110 10011110110011110101100011101000	First	01011001111000010011101100011100 00000010010010101000000111101011 01100111110101100000000011110101 11100011111010101011100010101100	49.22
		Middle	01000000101111101000110111111000 0 01010001010111111111110101001001 01010001101110000110110001111010 00111100010110101011111001101001	50
		Last	00010001010010100101001000011101 10010001111100010010111001111100 10010000000101010010111000101011 11111101010011100010010010011110	55.47
5000	10000011101011110011100000110000 11100011010111011010111100101001 00100001000101110000000111001011 1 01000010000111110111100111111000	First	00110011111110001100111111001001 0 00001101001111111001110110100000 10010100000000100001001010111001 1 00110010111000100101101101010100	48.44
		Middle	01110110000100001010110111111101 00001101011011001100110101111111 1 11100000000010001101110010000110 1 01110011111110101111110111101011	50.78
		Last	00011100100101001101110001111110 00011001010011001111111110100101 10100010100110010110010110010101 11100100110000011100001001010100	50

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 stegano communication," *J. Web Develop. Web Designing*, vol. 1, no. 3, 2016.

[4] S. M. Alwahbani and H. T. Elshoush, "Chaos-based audio steganography and cryptography using LSB method and 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 andA Chakraborty, "A survey on different techniques for covert communication using steganography," *IOSR 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. App. (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. Alsabhany, 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. Netw. 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. Bhitre and M. R. Sayankar, "A review on audio and video based steganography for data hiding," *IJSRSET*, vol. 4, no. 1, pp. 2394–4099, 2018.

[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," *Res. J. Appl. Sci., Eng. Technol.*, vol. 7, no. 11, pp. 2311–2323, Mar. 2014.

[18] P. Manimegalai, K. S. Gomathi, D. Ponniselvi, 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/rjaset.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. Chadha, N. 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. 177, no. 41, 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. (ICSigSys)*, May 2017, pp. 99–104.

[30] H. Ghasemzadeh and M. K. Arjmandi, "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, p. 2278, May 2014.

[33] M. M. Sadek, A. S. Khalifa, and M. G. Mostafa, "Video steganography: A comprehensive review," *Multimedia Tools Appl.*, vol. 74, no. 17, pp. 7063–7094, 2015.

[34] 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.

[35] M. Sharma, "Compression using Huffman coding," *Int. J. Comput. Sci. Netw. Secur.*, vol. 10, no. 5, pp. 133–141, 2010.
[36] A. A. Alsabhany, 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.
[37] M. M. Mahmoud and H. T. Elshoush, "A novel enhanced LSB algorithm for high secure audio steganography," in *Proc. IEEE CEEC*, Sep. 2018, pp. 125–130.
[38] R. Tanwar, K. Singh, M. Zamani, 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.
[39] 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.
[40] 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. CCSEIT-12*, Oct. 2012, pp. 372–376.
[41] 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.
[42] 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.
[43] 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.
[44] 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, Aug. 2017, Art. no. 012084.
[45] S. M. Alwahbani and H. T. Elshoush, "Hybrid audio steganography and cryptography method based on high least significant bit (LSB) layers and one-time pad—A novel approach," in *Dr. Yaxin Bi Editor. Studies in Computational Studies*, vol. 751. Cham, Switzerland: Springer, Jan. 2018, pp. 431–453.
[46] 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.
[47] V. 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.
[48] M. H. A. Al-Hooti, T. Ahmad, and S. Djanali, "Audio data hiding using octal modulus function based unsigned integer sample values," in *Proc. Int. Conf. Comput. Eng., Netw. Intell. Multimedia (CENIM)*, Nov. 2018, pp. 48–53.
[49] F. A. P. Petitcolas, "Kerckhoffs' Principle," in *Encyclopedia Cryptography Security*, H. C. A. van Tilborg and S. Jajodia, Eds. Boston, MA, USA: Springer, 2011, p. 675.
[50] A. Kerckhoffs, "La cryptographie militaire (military cryptography)," (in French), *J. Sci. Militaires*, 1883.
[51] G. Tzanetakis. (Nov. 22, 2015). *Music Analysis, Retrieval and Synthesis for Audio Signals (Marsyas)*. [Online]. Available: http://marsyasweb.appspot.com/download/data-sets/
[52] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293–302, Jul. 2002.
[53] *Perceptual Evaluation of Speech Quality (PESQ): An objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*, document Recommendation IT ITU-T P862.2. ITU-T P862.2, 2001.
[54] *Application Guide for Objective Quality Measurement Based on Recommendation*, document Recommendation P ITUt ITUt 862.3:862, 2005.
[55] R. D. Ra and D. Pugazhenthi, "Ideal sampling rate to reduce distortion in audio steganography," *Proc. Comput. Sci.*, vol. 85, pp. 418–424, Jul. 2016.
[56] F. Djebbar, B. Ayad, K. A. Meraim, 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
[57] J. Chaharlang, M. Mosleh, and S. Rasouli-Heikalabad, "A novel quantum steganography-steganalysis system for audio signals," *Multimedia Tools Appl.*, vol. 79, nos. 25–26, pp. 17551–17577, Feb. 2020.
[58] J. Chaharlang, M. Mosleh, and S. R. Heikalabad, "A novel quantum audio steganography–steganalysis approach using LSFQ-based embedding and QKNN-based classifier," *Circuits, Syst., Signal Process.*, vol. 39, no. 8, pp. 3925–3957, Jan. 2020.
[59] 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.
[60] 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.
[61] A. Altigani, S. Hasan, B. Barry, S. Naserelden, M. A. Elsadig, and H. T. Elshoush, "A polymorphic advanced encryption standard—A novel approach," *IEEE Access*, vol. 9, pp. 20191–20207, 2021.
[62] M. Bazyar and R. Sudirman, "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.
[63] 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.
[64] 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.
[65] 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.

MAHMOUD M. MAHMOUD received the B.Sc. degree (Hons.) from the Faculty of Mathematical and Computer Science, University of Gazera, Sudan, and the M.Sc. degree from the Faculty of Mathematical Sciences and Informatics, University of Khartoum, Sudan, where he is currently pursuing the Ph.D. degree with the Faculty of Mathematical Sciences and Informatics. He has four publications, one appearing in *Multimedia Tools and Applications*. His research interests include cryptography, information security, and steganography.

HUWAIDA T. ELSHOUSH received the bachelor's degree in computer science (division 1), the master's degree in computer science, and the Ph.D. degree in information security from the Faculty of Mathematical Sciences and Informatics, University of Khartoum, Sudan, in 1994, 2001, and 2012, respectively.

Her M.Sc. dissertation dealt with Frame Relay Security. She is currently an Associate Professor with the Computer Science Department, Faculty of Mathematical Sciences and Informatics, University of Khartoum, where she is also acting as the Head of Research Office. She has more than 26 publications and some of her publications appeared in *Applied Soft Computing* journal (Elsevier), *PLOS One* journal, IEEE Access, *Multimedia Tools and Applications*, *PeerJ Computer Science*, *Journal of Information Hiding and Multimedia Signal Processing*, and Springer book chapters. Her research interests include the information security, cryptography, steganography, and intrusion detection systems. She is a reviewer of many international reputable journals related to her fields, including *Applied Soft Computing Journal* (Elsevier).

Dr. Elshoush's awards and honors include the second-place prize in the ACM Student Research Competition SRC—SAC in Coimbra, Portugal, in 2013. Her article entitled "An Improved Framework for Intrusion Alert Correlation" has been awarded the Best Student Paper Award of the 2012 International Conference of Information Security and Internet Engineering (ICISIE) in WCE 2012. Other prizes were the best student during the five years of her undergraduate study.