Efficient Audio Steganography Using Bit Comparison and Substitution Algorithms for Improved Embedding Capacity

K Revathi[1], S. Kaja Mohideen[2*], and Latha Tamilselvan[3]
[1,2]Department of Electronics and Communication Engineering, B.S. Abdur Rahman Crescent Institute of Science and Technology, Vandalur, Chennai, India
[3]Department of Information Technology, B.S. Abdur Rahman Crescent Institute of Science and Technology, Vandalur, Chennai, India
E-mail: revathivigneswaranphd@gmail.com, kajamohideen@crescent.education,and latha.tamil@crescent.education
*Corresponding author

Keywords: audio steganography, lsb steganography, wav file, text files, mse, psnr, ber

Received: November 12, 2024

Steganography conceals information, with audio files increasingly used in this practice. The Least Significant Bit algorithm in audio steganography requires numerous samples to embed text files effectively. This study proposes an efficient methodology for embedding text files into a digital audio file. The approach also involves embedding the index value of an audio file, which contains hidden text files in another digital audio file. The proposed algorithms are the Bit Comparison and Substitution Algorithm for embedding text files in audio files and the Bit Comparison and Retrieval Algorithm for extracting text files from audio files. Additionally, a Least Significant Bit Algorithm with a secret key (Ks) embeds the index value of the audio file. The proposed algorithm achieves an average mean squared error of 0.0513, a peak signal-to-noise ratio of 41.13732041dB, and a bit error rate of 0.000114 for embedding 2400 bytes of text files in audio files. These performance metrics demonstrate that the proposed algorithm quantitatively outperforms the traditional least significant bit algorithm by requiring fewer audio samples to embed text files. The proposed algorithm embeds 2400 bytes of text into 2400 audio samples, with each byte corresponding to a single audio sample. Each byte of the text file is embedded by modifying one bit per audio sample, resulting in 2400 difference bits between audio and stego audio files.

Povzetek: Izvirna metodologija za vgradnjo besedilnih datotek v avdio datoteke z uporabo algoritmov za primerjavo bitov in zamenjavo bistveno izboljša kapaciteto vgradnje, s tem pa zmanjšuje število potrebnih vzorcev in povečuje varnost in kakovost skrivnega prenosa podatkov.

1 Introduction

In the domain of steganography, enhancing the secrecy of hidden files requires concealing two key attributes: the embedding method and the precise location within the audio file where the file is incorporated [1].

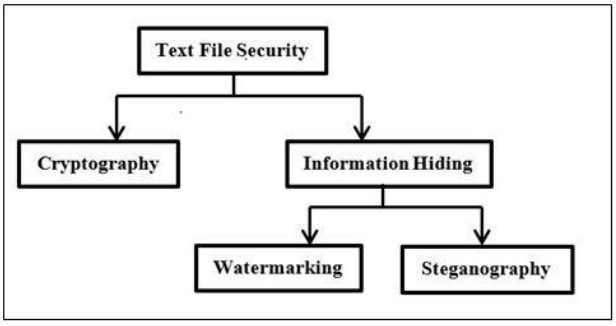


Figure 1: Text file security structure.

Cryptographic and steganographic techniques are essential to ensuring information security during communication. Figure 1 illustrates the structure of text file security [2]. Cryptographic algorithms enable the encryption and decryption of information, ensuring its security through the transformation of plaintext into ciphertext [3].

Steganography, alternatively, conceals information within a medium and traces its origins back to ancient Greece. Information has historically been hidden in various forms like music scores, paintings, letters written with invisible ink, or even human skin. Digital media steganography is a relatively recent area of research that focuses on concealing information in digital content, such as digital images[4], video[5], or audio files [6]. Audio steganography is of particular interest due to the slight distortion in the amplitude of audio files during data embedding, making audio a suitable cover medium. It hides multimedia objects in audio files such as WAV, and MP3. Using commonly available carrier files is highly desirable, as it reduces the risk of detection by

third-party observers and makes carrier modification caused by secret data insertion undetectable[7]. Figure 2 presents the model for audio steganography[8].

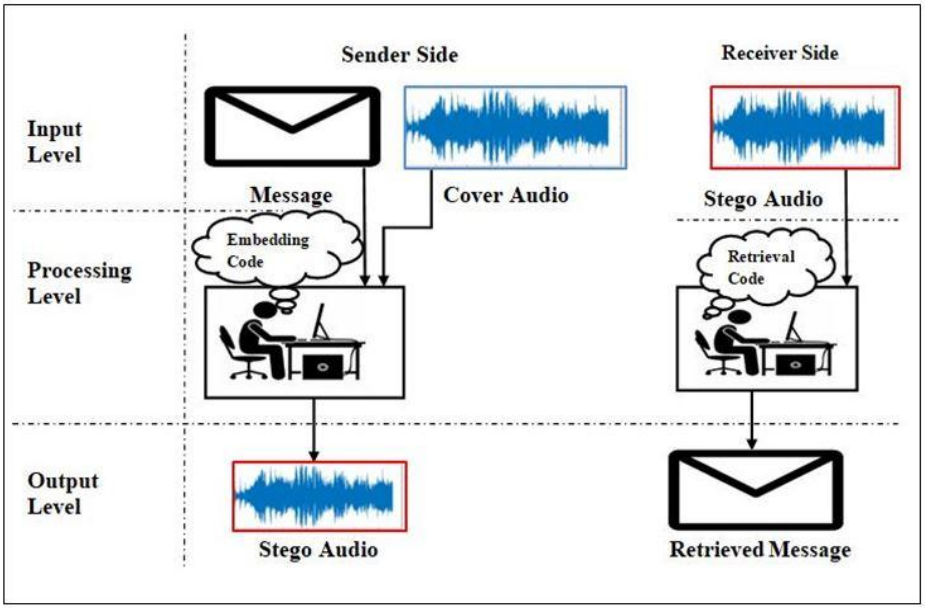


Figure 2: Model for audio steganography.

Efficient steganographic concealment generally requires three unique characteristics: security, capacity, and strength. Capacity refers to the total size of confidential data hidden within a file. An imperceptible steganographic system is considered secure. Robust audio steganography ensures that the parameters of both the cover and stego audio files are identical. Robustness refers to the stego file's resilience against steganalysis and conflicts with hiding capacity. Specifically, increasing hiding capacity reduces the security and imperceptibility of stego files. Balancing capacity, security, and imperceptibility poses a significant challenge.

Accordingly, the principal aim of steganography is to enhance embedding capacity, security, and imperceptibility while preserving robustness[1]. The hiding methods in audio files include Least Significant Bit (LSB), Echo, and Spread Spectrum, each developed for various applications[9], [10]. The LSB algorithm embeds each bit of the secret message into the LSB of the audio sample. Its simplicity and practical success derive from the limited sensitivity of the human auditory system (HAS). The HAS fails to detect subtle changes in audio frequency, particularly in the high-frequency regions.

However, a drawback is the predictable embedding of secret messages, which facilitates easy recovery for potential attackers. In response to these limitations, researchers have proposed various approaches to enhance the traditional LSB algorithm and mitigate these challenges [11], [12], [13]. To enhance embedding capacity, researchers embed the secret bits in the higher-order LSBs of the audio cover sample [9], [14]. At the same time, information loss occurs during communication due to noise.

The widespread use and accessibility of audio files make them suitable for conveying concealed information. Researchers have explored embedding information within audio samples using distortion functions, multi-domain embedding, Secure Advanced Audio Coding steganography scheme based on multi-view statistical distortion (SofMvD) with syndrome trellis code (STC), and adaptive-STC based on the adaptive parity-check matrix and constant distortion function. The targeted intelligent optimization algorithm, Generation of Optimal Allocation Strategy (GOAS), models the adaptive parity-check matrix to address the multi-constrained integer programming problem[15], [16], [17], [18]. The authors provided an in-depth review of audio steganography, discussing various concepts and methodologies in comparison to existing audio steganographic methods [8],[19]. The authors investigated the utilization of commercial off-the-shelf (COTS) software for audio steganography and steganalysis by formulating a model [20].

The authors explore the connections between information society, electronics, and artificial intelligence through twenty-four information society laws that demonstrate the rapid growth of electronic devices. The authors' concept enables the potential implementation of audio steganography communication in real-world applications with the support of AI [21].

The authors introduced a lightweight authentication solution for IoT edge devices, ensuring efficient data transmission under limited bandwidth. This solution uses lightweight symmetric cryptography and leverages the ChaCha20 algorithm for session key establishment, providing robust security with minimal communication and time costs [22]. Similarly, Audio steganography strengthens IoT security by concealing data within

multimedia files (audio), enabling secure communication as a future scope.

Steganography research extends security by integrating cryptography and steganography[23],[24].

The author proposed a novel steganographic scheme based on data decomposition and stego-coding methods to increase the embedding efficiency[25].

The study investigates double-layered security and multiple embedding techniques in steganography. These techniques include embedding an encrypted color image within another color image, hiding an encrypted text file within an image, embedding this image within an audio file, and embedding text files into the highest fitness bitstream position of the audio samples using a genetic algorithm[2],[26],[27].

Most steganography research focuses on images, with fewer studies exploring data hiding in audio files. Hence, audio steganography is an emerging area of research.

The LSB algorithm in current audio steganography methods embeds the least significant bit of an 8-bit audio sample with test file bits at a 12.5% embedding rate, and using two least significant bits results in a 25% embedding rate. Consequently, a low embedding rate restricts the amount of hidden data, and to embed more information, a larger audio file is required.

The probability of errors in the embedded bits in the LSB algorithm depends on the bit pattern of audio samples. The error bits are not predictable; they vary. Therefore, it is essential to propose algorithms that prioritize the reduction of bit error rates in a controlled pattern while simultaneously increasing the embedding rate.

Table 1 summarizes related work features and performance metrics to compare with the proposed algorithms.

Similarly, the Least Significant Bit algorithm uses eight audio samples to embed one byte of a text file. As the embedding capacity increases, more audio samples are needed to accommodate the increased capacity. The LSB algorithm utilizes 19200 samples for embedding 2400 bytes of text files.

Table 1: The summary of related works features and results.

Reference No.	Methodology	Performance / Results
[28]	Audio Steganography: Cover File: Audio Hidden File: Text Embedding Feature: LSB algorithm Encryption Feature: Pattern Matching Algorithm Dual Encryption	The Pattern Matching method achieved Mean Squared Error (MSE) of 0.5619 and 0.3994 in the Normal LSB method. Embedding rate is 12.5%.
[29]	Audio Steganography: Cover File: Audio Hidden File: Text Embedding Feature: Two LSB modification Encryption Feature: RSA to protect data.	MSE, Peak Signal to Noise (PSNR), and Bit Error Rate (BER) sequentially are 9.08875, 34.2775 dB, and 17.995%.
[30]	Audio Steganography: Cover File: Audio Hidden File: Text Embedding Feature: Encrypted Data embedded in 4 and 5 LSB bits of audio file.	A PSNR of 17.71916498dB resulted in the embedding of 103 characters in a JAZZ audio file.
[31]	Audio Steganography: Cover File: Audio Hidden File: Text Embedding Feature: Random Key Indexing method (Trusted Third Party Stego key and Secondary Key generated at encoder). Encryption Feature: AES-256	A 16-bit stereo audio file with 532bytes of embedded message resulted in a BER of Right Channel: 0.0187%. Left Channel: 0.0198%.
[32]	Audio Steganography: Cover File: Audio Hidden File: Audio Embedding Feature: ECA-BM: *Fractal coding and chaotic LSB	The NC of the reconstructed secret file is 0.99992
[33]	Audio Steganography: Cover File: Audio Hidden File: Audio Embedding Feature: HASFC *Fractal coding *Chaotic least significant bit	Audio Steganography model for secure audio transmission resistant to steganalysis attacks, with a Histogram error rate (HER) of 0.1278 and a Difference Ratio (DR) of the fourth first moments with mean:0.0799, variance:0.0008, skewness:0.0018, kurtosis:0.0028

[34]	Audio Steganography: Cover File: Audio Hidden File: Image and Audio Embedding Feature: Integer-to-Integer Lifting Wavelet Transform (Int2Int LWT) and Least Significant Bits (LSBs) substitution. Encryption Feature: Adaptive steganography key (Stego key)	The lossless audio steganography approach underwent a statistical steganalysis test that resulted in HER of 1.2274e-04 and a Difference Ratio (DR) of the first fourth moments with mean:0.2304, variance:0, skewness:0.0004, kurtosis:0.0005

These limitations of low embedding rates and high bit error rates, caused by the increased use of samples for embedding, underscore the need for the proposed algorithm, which aims to enhance embedding rates while minimizing error rates and thereby improving the overall effectiveness of audio steganography

The two proposed algorithms are:

1. The Bit Comparison and Substitution algorithm (BCS)
2. The Bit Comparison and Retrieval algorithm (BCR)

These algorithms significantly enhance the embedding process by embedding one byte (8 bits) from text file into one-byte of a two-byte audio sample. This innovative approach allows embedding one character within one audio sample in contrast to the LSB method. By employing a one-bit difference between the samples in the cover audio file and the resulting stego audio file, our algorithms ensure that only one bit is in error for every 8 bits of data embedded.

The novelty of our approach lies in its efficient utilization of audio samples, achieving high embedding rates while minimizing error rates. This capability enhances the security of the communication with a reduced number of samples involved in embedding.

2 Methodology

The proposed BCS and BCR algorithms address the research gap by using all bits in an 8-bit audio sample to embed an 8-bit character. This embedding process results in a bit error of 1 between the audio sample and the character.

It includes three main components: the Bit Comparison and Substitution algorithm (BCS) for embedding text files (TF) into the audio file (AF1), the Least Significant Bit algorithm for embedding the index of audio samples from the stego audio file (SAF1) into the audio file (AF2), and the LSB Retrieval algorithm for recovering index values from the stego audio file (SAF2), followed by the Bit Comparison and Retrieval (BCR) algorithm for extracting text files from the stego audio file (SAF1).

2.1 Bit comparison and substitution algorithm

2.1.1 Embedding text file into audio file

The BCS algorithm is designed for audio steganography with digital audio WAV files. Each sample in these files consists of two bytes, representing the audio's amplitude. The hidden message is a text file that includes punctuation, numerals, uppercase and lowercase letters, and printable characters. Each character is assigned its ASCII value and converted to a one-byte binary value. Figure 4 depicts the hiding behaviour of the proposed BCS algorithm. The algorithm compares the eight least significant bits from the audio file (AF1) with the corresponding bits from the text file. It calculates the sum of their bit differences, denoted as ‘K’. If K=1, an audio sample from AF1 is computed, embedding eight consecutive LSBs with one byte of text file data. This process is repeated for each text file byte, creating a Stego Audio File (SAF1). This embedding method integrates one byte of the text file into one byte of a two-byte audio sample in AF1, resulting in one incorrect bit. Figure 3 illustrates the framework of this model.

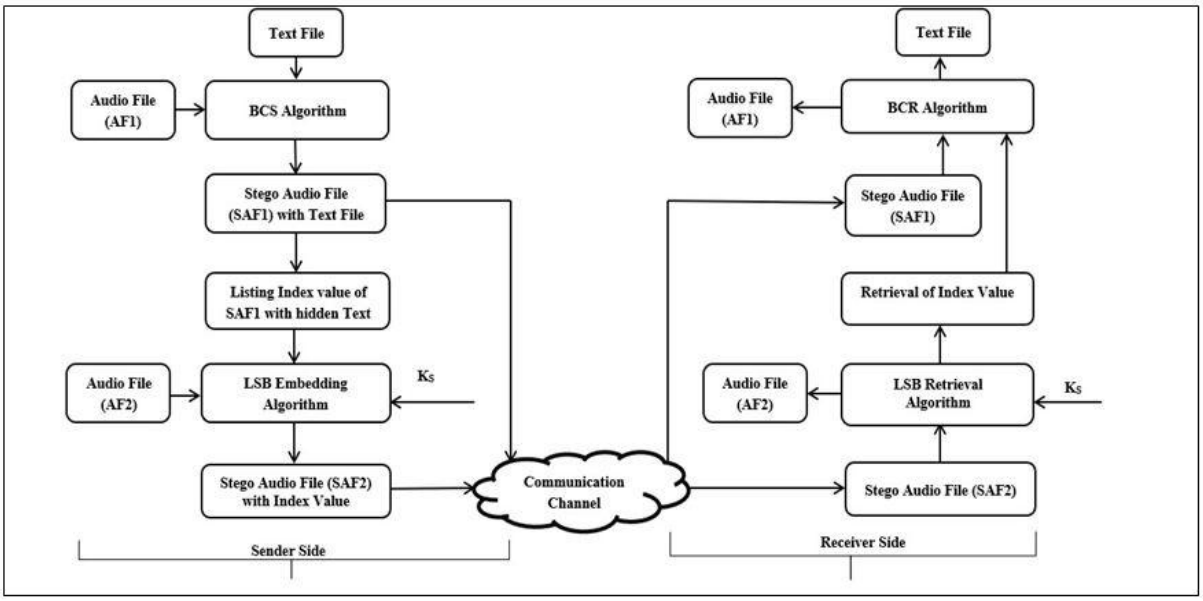


Figure 3: The framework of the proposed method

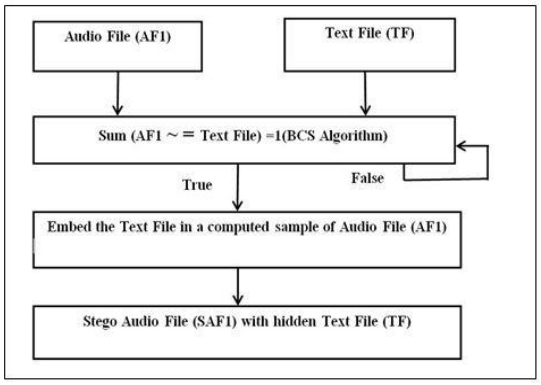


Figure 4: Flow diagram for proposed BCS algorithm.

It outperforms the LSB algorithm by producing fewer incorrect bits, thereby enhancing security and Peak Signal-to-Noise Ratio (PSNR). The size of the text file determines the number of difference bits between the cover (AF1) and stego audio file (SAF1). A 2400-character text file results in 2400 incorrect bits between the cover audio file (AF1) and the stego audio file using the BCS algorithm.

Furthermore, Figure 5 illustrates an example of embedding a one-byte text file in a two-byte audio file (AF1). Initially, both the audio file (AF1) and the text file are converted to binary, resulting in 0011110110101001 0010100101111010 0011110111110000 (Audio File 1) and 11111000 (Text File). Therefore, embedding the text file in the Audio File (AF1) commences if K=1.

2.1.2 BCS Algorithm-embedding text file in audio file

The Sender side in Figure 3 illustrates the algorithm steps for clearer understanding.

Input: Audio File (AF1) and Text File (TF).

Output: Stego Audio File (SAF1)

1. Read the Audio File (WAV) and Text File. //Total Samples in AF1 (N) [Figure 4]
2. Convert the Audio File and Text File into a bitstreams.
3. Conceal an authorized algorithm between the sender and receiver to reveal that the audio file (AF1) contains hidden data, as well as the index value of the SAF1 in the audio file (AF2).
4. For i = 1 to N do // Continue the For loop to embed the Text File (TF).
5. If sum ((AF1) ~=TF) == 1, then // Compute the sum of binary bits difference between AF1 and TF. [Figure 5]
6. Embed TF in AF1 // Embed Text File (TF) in a computed sample of Audio File (AF1). [Figure 5]
7. End
8. End
9. Computation of the stego Audio File (SAF1 with hidden text file). [Figure 4]
10. End

The algorithm's performance is validated with the results of Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) explained in the Results and Discussion section. The algorithm is also tested for steganalysis, proved through the results of histogram error rate (HER) and the fourth first moments. Furthermore, the algorithm undergoes a brute-force attack to evaluate its resistance.

2.1.3 Embedding index values of audio samples into an audio file

Each audio sample contains an index value for quick data retrieval, facilitating efficient database searches. The

index assists the database application in efficiently searching data within an extensive database.

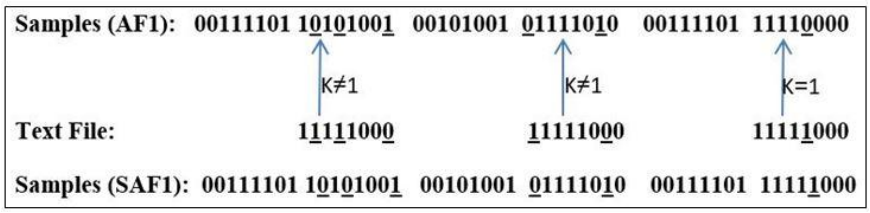


Figure 5: Example of embedding a text file in the audio file (AF1) when K = 1 is true.

An index of the stego audio file (SAF1) with hidden data is listed. The index values were initially in decimal form and then converted to binary. An uncompressed digital audio wave file (AF2) of two bytes is input for the LSB algorithm. In cryptographic processes, a private key (secret key) is essential for securely embedding and extracting data. This key is exclusive to authorized entities responsible for conducting steganographic communication. In the proposed BCS algorithm, the index value within the audio sample, at which the LSB algorithm initiates its execution, is called a private key (Ks) and can be changed for every communication, ensuring that each transmission remains isolated and secure. This algorithm selects K_s=50, indicating that the 50^{th} sample of AF2 is designated as the private key, mutually agreed upon by the sender and receiver for embedding. This random selection enhances security by creating an extensive combination space, with over 661504 samples across various audio files, as shown in Table 2. This results in approximately $2^{\wedge(661504)}$ possible combinations. The number of combinations makes it extremely difficult for an unauthorized party to determine the private key, thereby strengthening the overall security of the transmission. The index values of the stego audio file (SAF1) with hidden text files are embedded into AF2 using the LSB algorithm. This process begins with K_s as the index value from AF2 used as the initial sample for embedding. The LSB algorithm, known for its simplicity and effectiveness, executes after embedding all text files in the AF1. The resulting stego audio files (SAF1 and SAF2) hide text file and index values, respectively. Security of the LSB algorithm increases with the dynamic index length and Ks. Figure 6 shows the flow diagram of the embedding Index value.

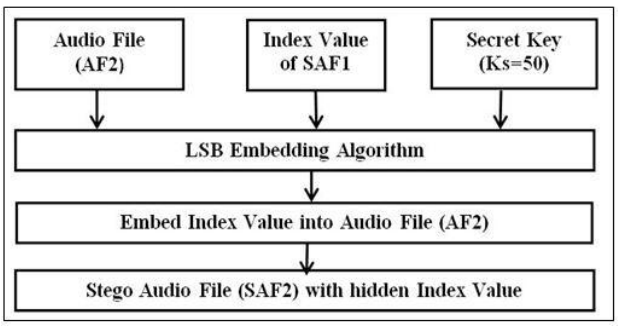


Figure 6: Flow diagram for embedding index values of SAF1 in the cover audio file (AF2).

2.1.4 Algorithm for Embedding Index Values of the Audio Samples in an Audio File

The sender side in Figure 3 illustrates the algorithm steps for clearer understanding.

Input: Stego Audio File (SAF1 with hidden text file) & Audio File (AF2)
Output: Stego Audio File (SAF2 with hidden index value)

1. Read the Audio file (AF2) and convert it into a stream of binary bits. [Figure 6]
2. List the stego audio samples (SAF1) index values containing a hidden text file. [Figure 6]
3. Convert the index values into a sequence of binary bitstreams.
4. Use the LSB embedding algorithm (replacing the least significant bit of each sample in an audio file (AF2) with the index value) to embed all index values into the audio file (AF2), starting from the Ks=50^{th} sample. [Figure 6]
5. Compute the stego audio file SAF2 with the hidden index value. [Figure 6]
6. Transmit both stego audio files SAF1 and SAF2 to the receiver.
7. End

The metrics Peak Signal-to-Noise Ratio and Bit Error Rate (BER) discussed in the Results and Discussion section verify the algorithm's performance.

2.2 Bit comparison and retrieval algorithm

The procedures associated with embedding and extracting multimedia objects are inherently inverse. The primary step in the proposed BCR algorithm is retrieving the index values representing stego audio file (SAF1) samples from the stego audio file (SAF2), with Ks serving as the reference. The Least Significant Bit retrieval algorithm retrieves the index values of the stego audio file (SAF1). The aforementioned indexed values correspond to the stego audio file (SAF1) containing the hidden text file information. The text file is then extracted from the retrieved index value. This process involves specifically extracting the eight consecutive bits from the Least Significant Bit of the previously mentioned index values of samples in SAF1. The complete hidden text files from SAF1 are retrieved by repeating the above process. The extracted text files are

then converted into their decimal representations. Finally, the decimal values of the extracted text files are converted into their equivalent ASCII values. A flow diagram of the Bit comparison and Retrieval (BCR) algorithm is present in Figure 7.

2.2.1 BCR Algorithm-extracting index values and text file from audio files

The receiver side in Figure 3 illustrates the algorithm steps for clearer understanding.

Input: Stego Audio File (SAF1 with hidden text file) and Stego Audio File (SAF2 with hidden index value)
Output: Index value, Text File, and Audio Files (AF1 & AF2).

1. Retrieve the index values corresponding to samples in the stego audio file (SAF1) from the stego audio file (SAF2) using the LSB retrieval algorithm with Ks as the starting sample of SAF2. (Read the least significant bit of the stego audio file samples (SAF2)). [Figure 7]
2. Repeat Step 1 to retrieve all hidden index values.
3. Convert the retrieved index values into decimal representations.
4. Retrieve the text file from the stego audio file (SAF1) using the index value as the reference using the Bit comparison and retrieval algorithm. (Read the eight least significant bits of the stego audio file (SAF1)). [Figure 7]
5. Repeat Step 4 to retrieve all hidden characters in text file from the referenced index values in Step 3.
6. Convert every byte of the retrieved file to decimal notations.
7. Finally, convert the decimal values into their corresponding ASCII characters to retrieve the text file.
8. End

The algorithm's efficiency is validated with the results of Bit Error Rate (BER) and Normalized Cross-Correlation (NCC), as explained in the Results and Discussion section.

2.3 Computational complexity

The computational complexity of the proposed algorithm was evaluated in three stages: Input processing, BCS (Bit Comparison and Substitution algorithm), and BCR (Bit Comparison and Retrieval algorithm). Execution times for each stage were noted by considering different text file sizes and audio file types. The algorithm was executed over 245 runs for each stage, with seven audio file types (Classical, Country, Disco, Hip-hop, Jazz, Pop, and Rock), each tested with seven different text file sizes (80, 400, 800, 1200, 1600, 2000, and 2400 bytes). To ensure an accurate analysis of the execution time for each stage, each combination of audio files and text file sizes was executed five times.

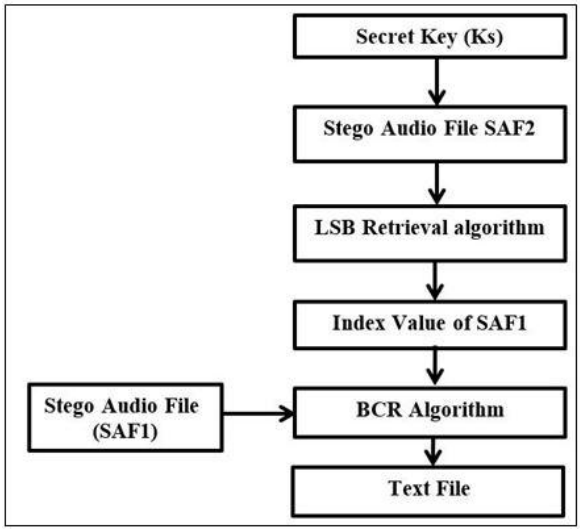


Figure 7: Flow diagram for extracting index values and text file from stego audio files (SAF2 and SAF1).

For input processing, execution times showed a nearly constant pattern, with minimum variations as the text file size increased, as shown in Figure 8(a). The observed pattern suggests an O(1) complexity for this stage. The BCS stage, in contrast, showed a linear increase in execution time with slight variations between audio file types, as shown in Figure 8(b). This linear increase indicates an O(TF) complexity in the BCS stage.

Similarly, the BCR stage showed linear growth in execution times notably lower compared to the BCS stage, as shown in Figure 8(c). The BCR's consistent linear scaling across audio file types further supports its O(TF) complexity. In summary, the algorithm maintains a computationally efficient structure with linear complexity (O(TF)) for both BCS and BCR stages, making it suitable for scalable steganographic applications in high-data environments.

Even though the execution time of the algorithm increases with larger audio files and text sizes, the reduced number of audio samples required for embedding larger text file compensates for this increase. The novel algorithm embeds each character into an audio sample and retrieves the character from the same sample. This approach optimizes memory usage and reduces processing time, ensuring efficiency even with increased text file inputs. The reduced memory usage balances the increased execution time.

2.4 Generalizability of the proposed method

The proposed algorithms have been applied successfully to both speech and ambient audio files, demonstrating their adaptability across different audio types beyond musical genres like Jazz and Rock. Tables 9 and 12 outline the specifications of the speech and ambient audio files. Concurrently, Tables 10 and 13 present the performance metrics for these files. These metrics demonstrate that the algorithms maintain efficient data embedding and retrieval capabilities across varied audio contexts. Tables 11 and 14 provide insights into the audio samples and associated difference bits when embedding text files in these audio files.

To further support the generalizability of the BCS algorithm across different audio files, we present histograms comparing the cover and stego audio files for speech and ambient audio files. These files include hidden text file containing 2400 characters, as shown in Figures 14 and 16. Figures 15 and 17 depict the audio files before and after embedding 2400-byte text file. They show no significant degradation in audio quality.

No significant technical barriers emerged when applying the novel BCS and BCR algorithms to diverse audio types. However, the algorithm's performance depends on specific characteristics of the audio samples and the bit patterns within them. Despite this, the embedding method and the resultant error bit remain consistent across different audio file types.

3 Results and discussion

The MATLAB software was used to evaluate the BCS and BCR algorithms using seven distinct audio files representing genres: Classical, Disco, Hiphop, Country, Jazz, Pop, and Rock. The GTZAN dataset was selected for its genre coverage (1,000 clips of 30 seconds each), compact size, and consistent 22050 Hz sampling rate, making it suitable for efficient embedding without extensive preprocessing. The consistent sampling rate ensures uniform audio quality across the dataset, minimizing variability that could affect embedding or extraction performance. Additionally, the dataset's compact size allows for quick processing and repeated testing, enabling efficient experimentation with the steganography algorithm. Its genre diversity provides a variety of audio contexts to test the adaptability of the algorithm.

Furthermore, its open availability promotes reproducibility. Using unprocessed audio minimizes computation time and accurately reflects performance conditions for the steganography algorithms. Table 2 details the specifications of the audio files from GTZAN.

The experimentation was conducted on a Windows 10 laptop with a Core i3 processor at 2.0 GHz and 4 GB of RAM. Uncompressed audio files from the GTZAN dataset served as the cover audio [1],[33],[35],[36]. The text files used in this study are from the Reuters-21578 collection, based on the 1994 CIA World Factbook, containing Prolog facts about LandBoundaries, NaturalResources, Population, Capital, and more[37]. The evaluation employed MSE, PSNR, BER, and NCC to quantify deviations in the audio files and measure the efficiency of the BCS and BCR algorithms.

3.1 Similarity analysis

3.1.1 Mean squared error (MSE)

MSE measures audio distortion, calculated as the average square difference between the cover and stego audio files using Equation (1)[33]. If the MSE is zero, the files are similar.

$$\mathrm{MSE} = \frac{1}{\mathrm{Q}}\sum_{\mathrm{i}=1}^{\mathrm{Q}}(\mathrm{s1_i} - \mathrm{s2_i})^2 \qquad (1)$$

$\mathrm{s1_i}$ and $\mathrm{s2_i}$ represent samples from the cover and stego audio file, respectively, and Q is the total number of audio samples.

Figure 9(a) represents average MSEs over 49 runs, and their standard deviations. The X-axis represents the text file's length in Bytes, and the Y-axis indicates the MSE. The proposed algorithm achieves an MSE of

Table 2: Description of audio files.

Bits per sample/Sample rate	16/22050Hz
Number of Samples	661504-664180
Channel and Audio Type	Mono and Music
Duration in seconds	1-30

0.0513, indicating minimal distortion. This MSE outperforms existing algorithms [28],[29], which recorded MSE values of 0.5619 and 9.08875 respectively. Table 3 presents the MSE values of both the proposed and existing algorithms, along with the significant differences between these values. This difference results from maintaining a 1-bit difference between the audio and the stego audio samples when embedding one byte of text file. In the proposed BCS algorithm, embedding 2400 bytes (19200bits) of text files results in 2400 error bits, which is successfully accomplished. This controlled one bit difference minimizes distortions, especially when embedding large amounts of data for audio steganographic communication.

The standard deviation of MSE values is 0 across all text files and audio types. This consistency indicates a stable algorithm with a fixed, predictable distortion level in preserving audio quality. The same MSE value is observed across an audio file for different text file sizes, as shown in Figure 9(a), with values such as 0.0031 for 'Classical' audio across text file sizes ranging from 80 bytes to 2400 bytes. This uniformity occurs because the embedding process introduces minimal distortion, and the MSE is calculated as an average, leading to identical values and a standard deviation of zero.

Statistical analysis yields a p-value of 0 ($p < 0.05$), which rejects the null hypothesis and confirms significant differences in MSE across audio files. This analysis suggests that the MSE varies with the bit patterns in the audio samples. Despite these variations, low and consistent MSE values demonstrate that the audio

steganography method effectively controls distortion. As a result, the audio quality remains largely unaffected during data embedding.

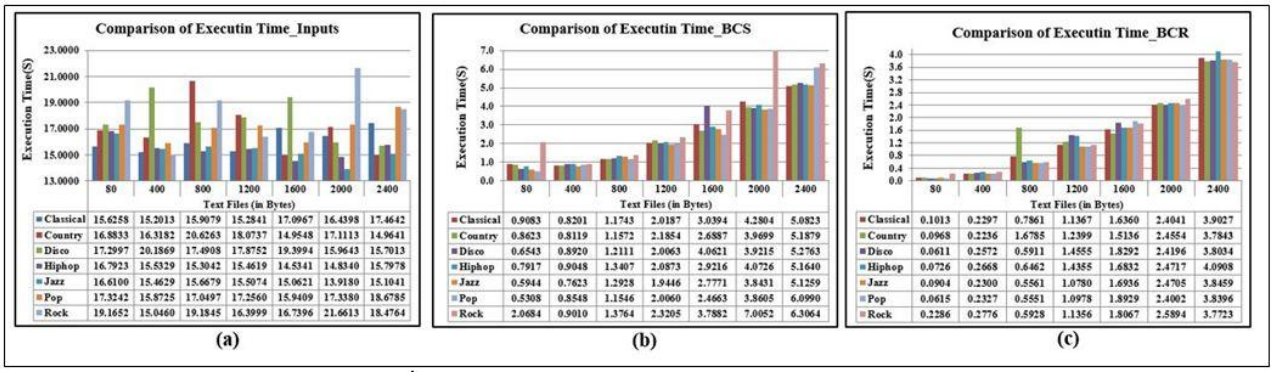
3.1.2 Peak Signal-to-Noise Ratio (PSNR)

PSNR evaluates audio quality in decibels (dB) using Equation (2). A higher PSNR with lower MSE indicates better steganographic quality.

$$\text{PSNR} = 10\log_{10}\frac{(2^{q}-1)^2}{\text{MSE}} \qquad (2)$$

Where q is the maximum bits per signal sample, and MSE is the Mean Squared Error[33].

Figure 9(b) shows the PSNR results over 49 runs for audio clips with different embedding rates: 80B, 400B, 800B, 1200B, 1600B, 2000B, and 2400B. The X-axis indicates the length of text files in bytes, and the Y-axis indicates PSNR. The average PSNR for stego audio (SAF1) is 41.13732041dB, while for stego audio (SAF2), it is 32.4868dB. Table 3 compares the PSNR of the proposed BCS algorithm with that of the existing



(a) Input	80	400	800	1200	1600	2000	2400
Classical	15.6258	15.2013	15.9079	15.2841	17.0967	16.4398	17.4642
Country	16.8833	16.3182	20.6263	18.0737	14.9548	17.1113	14.9641
Disco	17.2997	20.1869	17.4908	17.8752	19.3994	15.9643	15.7013
Hiphop	16.7923	15.5329	15.3042	15.4619	14.5341	14.8340	15.7978
Jazz	16.6100	15.4629	15.6679	15.5074	15.0621	13.9180	15.1041
Pop	17.3242	15.8725	17.0497	17.2560	15.9409	17.3380	18.6785
Rock	19.1652	15.0460	19.1845	16.3999	16.7396	21.6613	18.4764

(b) BCS	80	400	800	1200	1600	2000	2400
Classical	0.9083	0.8201	1.1743	2.0187	3.0394	4.2804	5.0823
Country	0.8623	0.8119	1.1572	2.1854	2.6887	3.9699	5.1879
Disco	0.6543	0.8920	1.2111	2.0063	4.0621	3.9215	5.2763
Hiphop	0.7917	0.9048	1.3407	2.0873	2.9216	4.0726	5.1640
Jazz	0.5944	0.7623	1.2928	1.9446	2.7771	3.8431	5.1259
Pop	0.5308	0.8548	1.1546	2.0060	2.4663	3.8605	6.0990
Rock	2.0684	0.9010	1.3764	2.3205	3.7882	7.0052	6.3064

(c) BCR	80	400	800	1200	1600	2000	2400
Classical	0.1013	0.2297	0.7861	1.1367	1.6360	2.4041	3.9027
Country	0.0968	0.2236	1.6785	1.2399	1.5136	2.4554	3.7843
Disco	0.0611	0.2572	0.5911	1.4555	1.8292	2.4196	3.8034
Hiphop	0.0726	0.2668	0.6462	1.4355	1.6832	2.4717	4.0908
Jazz	0.0904	0.2300	0.5561	1.0780	1.6936	2.4705	3.8459
Pop	0.0615	0.2327	0.5551	1.0978	1.8929	2.4002	3.8396
Rock	0.2286	0.2776	0.5928	1.1356	1.8067	2.5894	3.7723

Figure 8: Execution time for (a) Input (b) BCS (c) BCR.

algorithm[29],[30], highlighting the significant increase. For JAZZ audio with 2400 bytes, PSNR is 36.4463dB, while [29] yields 34.2775dB and [30] yields 17.71916498dB.

The average PSNR of 41.13732041 dB correlates with the near-zero MSE (0.0513). This correlation arises from the logarithmic relationship between MSE and PSNR. The low MSE achieved by the proposed embedding method reflects minimal distortion and leads to a high PSNR value.

The observed range of standard deviation values for PSNR is 4.85 to 5.16 across different audio file types and text file sizes, as shown in Figure 9(b). These values indicate that the algorithm maintains audio quality with some variability in how the embedding process affects each type of audio file. Specifically, the standard deviation of Country audio files is 4.85, while Classical and Hip-hop exhibit 5.16. The error bars in Figure 10(a) represent the standard deviation and show the variation in average PSNR values across different audio files and text file sizes. This results in the spread of PSNR values and the reflection of the range of impact on audio quality.

The moderate standard deviation across all audio file types indicates a balanced trade-off between embedding efficiency and audio quality preservation, indicating reliable algorithm performance.

Figure 11(b) illustrates the cover and stego audio files for various genres, with the stego audio embedded with a text file of 2400 bytes.

Remarkably, no noticeable perceptual changes are observed between the cover audio sample (AF1) and the corresponding stego audio sample (SAF1), as modifications affect only one of the 16 bits per sample to embed a one-byte text file.

3.2 Hiding capacity (payload)

Hiding capacity(HC) represents the maximum size of a text file concealed in an audio file and is calculated by Equation (3) as a standard percentage [1].

$$\text{HC} = \left(\frac{\text{Text File size}}{\text{Audio File size}}\right) * 100 \qquad (3)$$

Similarly, for the LSB algorithm, the hiding capacity is measured by Equation (4)[1].

$$\text{HC} = \left(\frac{\text{number of samples}}{8}\right) * 100 \qquad (4)$$

The tested Jazz audio file with 661794 samples embeds 2400bytes of text files using 19200 samples for the LSB algorithm and 2400 audio samples for the BCS algorithm. Table 4 illustrates the differences in audio samples used at different embedding capacities between the traditional LSB and the proposed BCS algorithm for a Classical audio file with 661794 samples.

The BCS algorithm effectively embeds text data by comparing each byte of the text file with audio samples to find those differing by one bit. The identified one-bit difference facilitates data embedding while preserving audio integrity. This approach achieves a 1:1 ratio, embedding a single character (8 bits) into one audio sample (16 bits). The BCS algorithm successfully embedded 80 bytes of text (80 characters) using 80 audio samples.

Conversely, the Least Significant Bit algorithm embeds each 8-bit character from the text file into the least significant bits of eight corresponding audio samples, resulting in eight audio samples for each embedded character. The discrepancy in Table 4 arises from these differing methodologies. The LSB algorithm consistently requires eight audio samples per character, and the BCS algorithm uses only one audio sample per character based on the one-bit difference criterion. Despite the differing embedding rates, the BCS method maintains consistent sample utilization by embedding each character into an audio sample with the required bit difference. This approach allows for more efficient sample utilization.

Moreover, the proposed BCS algorithm aligns sample usage with the text file size, effectively increasing embedding capacity and rate while reducing the bit error rate.

By selecting audio samples with a one-bit difference between text and audio, the BCS algorithm enhances the robustness and security of audio steganographic communication. Using fewer samples improves efficiency by minimizing processing time and enhances imperceptibility by reducing noticeable audio changes.

3.2.1 Embedding rate

The embedding rate is the number of bits concealed in a one-byte audio sample. The LSB algorithm embeds 1 byte of the text file in 8 audio samples (1 byte) with an embedding rate of 12.5%. In contrast, in the BCS algorithm, 1 byte of the text file is embedded into 1 byte of a 2-byte audio sample with a 50% embedding rate. For the tested JAZZ audio with 661794 samples, the proposed algorithm achieves an increase of 37.5% in embedding rate. Hence, the BCS algorithm has significantly improved its embedding rates compared to the traditional Least Significant Bit.

3.3 Strength analysis

3.3.1 Bit error rate

Equation (5) calculates the Bit Error Rate (BER), representing the percentage of incorrect bits in retrieved text files.

$$BER = \frac{Q_{error}}{Q_{bits}} \qquad (5)$$

Where Q_{error} is the number of differing bits, and Q_{bits} is the total number of bits embedded in the audio file[1]. A lower BER indicates robust steganographic algorithms. The proposed BCR algorithm successfully extracts the original text file for all the tested audio files.

Figure 10(b) illustrates the BER results for the proposed algorithm over 49 runs, where the X-axis represents the length of the text file in bytes, and the Y-axis represents the corresponding BER values. The Proposed algorithm computes a BER of 0.00000753 for a country WAV file of 80 bytes and 0.000226 for a rock WAV file with 2400 bytes. Similarly, the average BER of a stego audio file (SAF2) is 0.0009. The BER value illustrates the proposed algorithm's robustness in security and accuracy in maintaining consistency in both the audio and the stego audio file.

The average BER of the proposed algorithm is 0.000114, outperforming results from previous research [29],[31], with specific increase values as shown in Table 5.

This low BER results from altering only 1 bit out of 16 (1/16) in a 16-bit audio file sample (AF1). Specifically, each 8-bit character in the text file is embedded into a corresponding 16-bit audio sample with a controlled one-bit difference between the cover audio file (AF1) and the stego audio file (SAF1) samples.

By embedding 2400 characters (2400 * 8 = 19200 bits), the BCS algorithm achieved a total controlled bit difference of 2400 bits, resulting in lower BER.

This consistent approach limits the variability in the audio, maintaining audio quality across various genres such as Classical, Disco, Hip-hop, Country, Jazz, Pop, and Rock, thus preserving a satisfactory listening experience. Table 4 conveys the differences in bits used in both algorithms, illustrating the efficiency of the BCS method. In contrast, the conventional LSB algorithm modifies the least significant bits without a predefined or controlled pattern, as the changes depend on the embedded data and the audio samples. This results in unpredictable bit alterations, higher BER variability, increased error rates, and reduced performance due to the ineffective management of bit differences. Conversely, the structured embedding process in the BCS algorithm increases the robustness of security by reducing errors and optimizing the use of embedding samples, thereby supporting claims of improved performance over traditional methods.

The observed standard deviation in the Bit Error Rate was low, approximately 7.9E-05, as shown in Figure 10(b), across all file sizes. The standard deviation value indicates minimal variation in the data embedding process, influenced by the audio files. The algorithm reliably preserves both data and audio quality. Error bars for average BER values in Figure 11(a) show the range of variability in BER. The error bars represent the standard deviation of the average BER values across different audio files and text file sizes.

Statistical analysis yields a p-value of 0.999999986 ($p < 0.05$), indicating no significant differences in BER across audio files. The above analysis suggests that the BER of the algorithm's performance remains consistent, as shown in Figure 10(b), regardless of the audio file and embedding capacity used. The audio steganography method effectively maintains high transparency and robustness in data embedding without significantly affecting the cover audio.

3.3.2 Normalized cross-correlation

Similarity is measured by NCC, with the formula for calculating the similarity between embedded and retrieved text files given by Equation (6)

$$\mathrm{NCC}(\mathrm{M},\mathrm{M}') = \frac{\sum_{k=1}^{P} \mathrm{M}(\mathrm{K})\,\mathrm{M}'(\mathrm{K})}{\sqrt{\sum_{K=1}^{P} \mathrm{M}(\mathrm{K})^2}\sqrt{\sum_{K=1}^{P} \mathrm{M}'(\mathrm{K})^2}} \qquad (6)$$

Table 3: The MSE and PSNR Comparison between Proposed and Existing Algorithms

Method	Text Files (Characters)	MSE	PSNR (dB)
[28] Normal LSB Method	250	0.3994 ▲87.16%	-
[28] Pattern Matching Method	250	0.5619 ▲ 90.87%	-
[29]	107	9.08875 ▲99.44%	34.2775 ▲20.01%
[30]	103	-	17.71916498 ▲132.163%
Proposed	**2400**	**0.0513**	**41.13732041**

Table 4: Audio samples utilized in the proposed algorithm compared with the traditional LSB algorithm.

Length of Text File (in Bytes)	Audio Samples Utilized (ASU) (in samples)		Difference Bit (DB) (in Bits)	
	LSB	**Proposed**	LSB	**Proposed**
80	640	**80**	341	**80**
400	3200	**400**	1541	**400**
800	6400	**800**	3185	**800**
1200	9600	**1200**	4840	**1200**
1600	12800	**1600**	6433	**1600**
2000	16000	**2000**	8043	**2000**
2400	19200	**2400**	9586	**2400**

Table 5: Comparisons of BER between the proposed and existing algorithms

Method	Length of Text File (in Bytes)	BER
[29]	107	17.995% ▲99.94%
[31]	532	0.0187% ▲39.04%
Proposed	**2400**	**0.000114**

Where M and M′represent the embedded and extracted text files, respectively, and P is the number of samples in the audio file[36]. For all tested audio files, an NCC value of 1 indicates that the proposed BCR algorithm successfully extracts the complete text files. Similarly, the cover audio and stego audio files are similar, as shown in Figure 11(b).

Table 6 compares the NCC results obtained using the proposed algorithm with those of existing algorithms reported by researchers in their studies [32], along with the significant differences. The proposed algorithm's NCC values surpassed the existing research, highlighting its superior performance. This performance is due to one bit of difference between the computed audio file and the embedded character. Similarly, the continuity in the pattern of the audio file between the cover and stego audio files is maintained, resulting in a Normalized Cross-Correlation value of 1.

3.4 Steganalysis

3.4.1 Histogram attack

Histogram attacks were examined through forty-nine experiments across seven audio files and text files in various sizes from 80 Bytes to 2400 Bytes. Histogram Error Rate (HER) measures the error between the audio file and the stego file using Equation (7) [33].

$$\mathrm{HER} = \frac{\sum_{i=1}^{Q}(\mathrm{His_a} - \mathrm{His_s})^2}{\sum_{i=1}^{Q} \mathrm{His_a}^2} \qquad (7)$$

Where $\mathrm{His_a}$ and $\mathrm{His_s}$ are the histograms of audio and stego audio files, respectively, and Q is the total audio samples.

The proposed algorithm demonstrates improved histogram error rates, achieving a HER of 0.0000000043.

Table 6: Comparison of NCC between the proposed and existing algorithms.

Method	NCC
[32]	0.99992 ▲0.008%
Proposed	**1**

The experimental HER value approaches zero, primarily due to the uniform probability distribution of amplitudes within the audio files, confirming its resistance to histogram attacks. Figure 11(c) illustrates the histograms for the seven audio files before and after embedding a text file of size 2400 bytes.

3.4.2 Fourth first moments

The difference between the audio file and the stego file was analysed statistically using the fourth first moments, namely, mean (μ), variance (σ^2), skewness (sk), and kurtosis (k). Equations (8),(9),(10), and (11) calculate the fourth first moments[33]. Equation (12) calculates the absolute difference between these moments, yielding the Difference Ratio (DR)[36].

The experimental results of the DR values are close to 0 in Table 7, proving that the proposed algorithm is resistant to statistical investigation.

Moreover, the audio file's Difference Ratio (DR) approaches zero because of its symmetry, as indicated by near-zero values for both mean and skewness. A kurtosis value of zero further confirms that the amplitude distribution of the stego audio file, containing embedded text files, follows a normal distribution, leading to the near-zero DR.

$$\mu = \frac{\sum_{i=1}^{Q} S_i}{Q} \quad (8)$$

$$\sigma^2 = \frac{\sum_{i=1}^{Q} (S_i - \mu)^2}{(Q-1)} \quad (9)$$

$$sk = \frac{\sum_{i=1}^{Q} (S_i - \mu)^3}{(Q-1)\sigma^3} \quad (10)$$

$$k = \frac{\sum_{i=1}^{Q} (S_i - \mu)^4}{(Q-1)\sigma^4} \quad (11)$$

$$\text{Difference Ratio(DR)} = \left|\frac{fm_a - fm_s}{fm_a}\right| \times 100 \quad (12)$$

Where fma and fms are any fourth first moments of the audio file and the stego audio file, respectively.

S_i denotes the Samples in the cover or stego audio files, and Q represents the total number of audio samples in the cover or stego audio files.

3.4.3 Brute-Force attack

The objective of a brute force attack is to attempt multiple methods or keys to uncover the hidden text files from stego audio files. This type of attack is commonly used to test the strength and effectiveness of the steganographic algorithms. The total number of possible sample combinations in an audio file against brute-force attacks on the audio steganographic system is given by $2^{(b*N)}$, where b is the number of bits used for embedding per audio sample (8 bits per sample), and N is the total number of audio samples. This approach derives from cryptographic analyses of key space sizes. Table 8 compares the proposed algorithm with the LSB algorithm under a brute force attack.

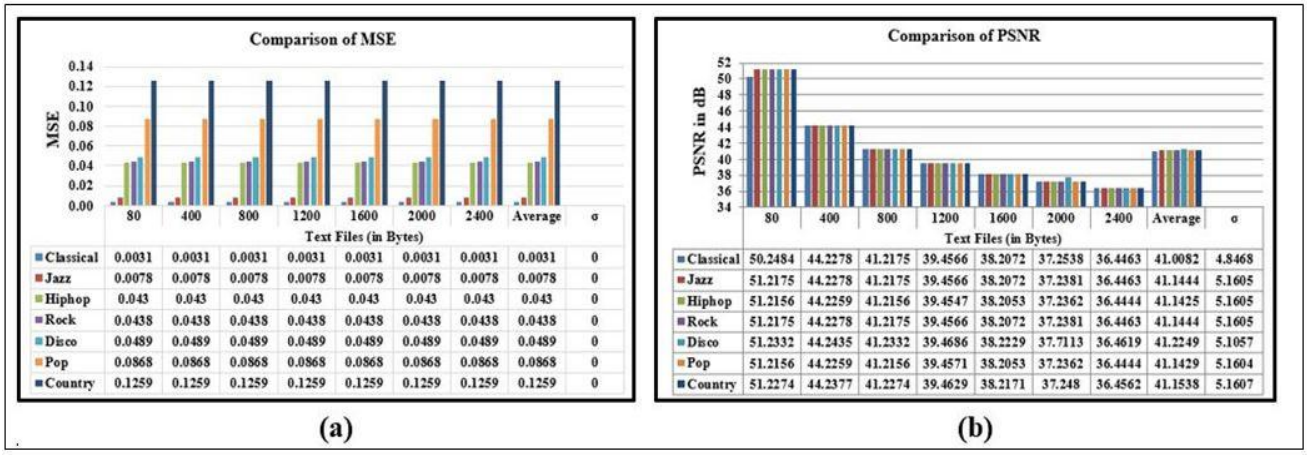


Figure 9: Influence of (a) MSE, and (b) PSNR.

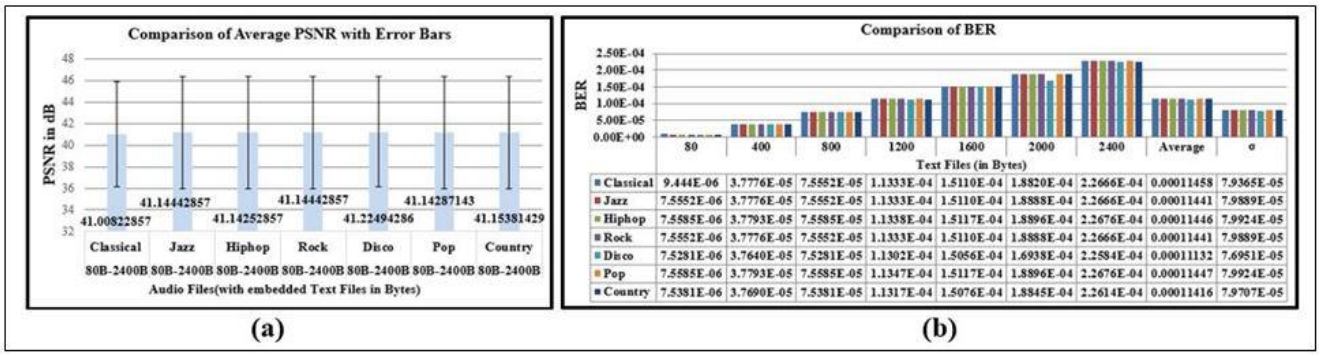


Figure 10: Influence of (a) Average PSNR with Error Bars, and (b) BER.

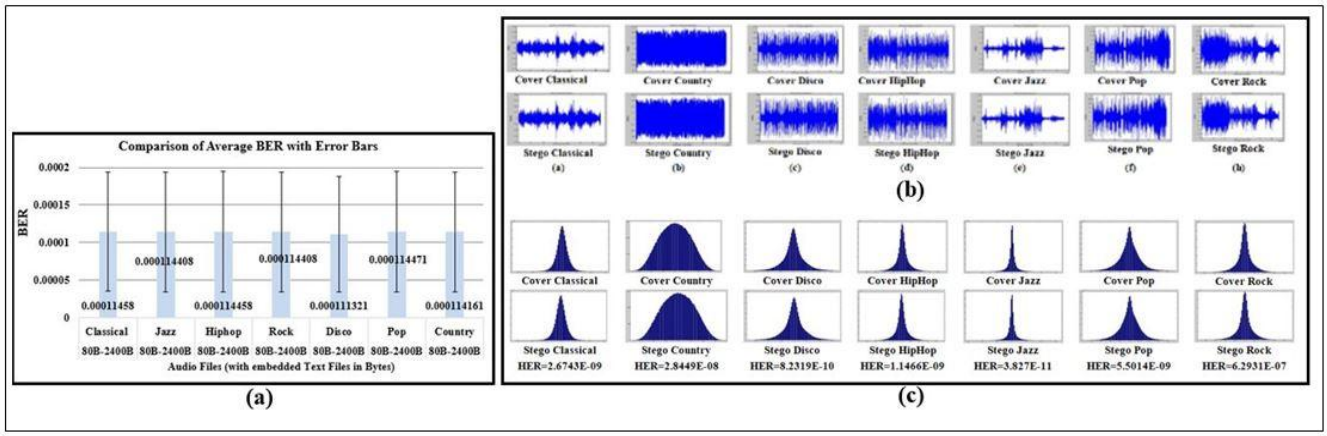


Figure 11: (a) Influence of average BER with error bars, (b) Comparison of cover and stego audio files with 2400 bytes of embedding capacity, and (c) Histograms of cover and stego audio files with 2400 bytes of embedding capacity.

Table 7: Comparison of DR for the Fourth First Moments and HER values for various cover audios using 2400B of embedded Text Files.

Audio File	Difference Ratio (DR)				HER
	1st moment, Mean (μ)	2nd moment, Variance(σ^2)	3rd moment, Skewness(sk)	4th moment, Kurtosis(k)	
[33]	0.0799 ▲99.29%	0.0008 ▲97%	0.0018 ▲95.27%	0.0028 ▲98.68%	0.1278 ▲100%
[34]	0.2304 ▲99.75%	0 ≈	0.0004 ▲78.75%	0.0005 ▲92.6%	1.2274E-04 ▲100%
Proposed	**0.00057**	**0.000024**	**0.000085**	**0.000037**	**4.33E-09**

Table 8: Compare the proposed algorithm with the LSB algorithm under brute force attack.

Audio File			Embedded Bits (b)	Total Combinations $2^{\wedge(b*N)}$	
Name	Samples (N)	Bits		Proposed Algorithm	LSB Algorithm
Classical	661794	16	8	$2^{\wedge(8*661794)}=2^{5294352}$	$2^{\wedge(1*661794)}=2^{661794}$
Country	663300	16	8	$2^{\wedge(8*663300)}=2^{5306400}$	$2^{\wedge(1*663300)}=2^{663300}$
Disco	664180	16	8	$2^{\wedge(8*664180)}=2^{5313440}$	$2^{\wedge(1*664180)}=2^{664180}$
Hip Hop	661504	16	8	$2^{\wedge(8*661504)}=2^{5292032}$	$2^{\wedge(1*661504)}=2^{661504}$
Jazz	661794	16	8	$2^{\wedge(8*661794)}=2^{5294352}$	$2^{\wedge(1*661794)}=2^{661794}$
Pop	661504	16	8	$2^{\wedge(8*661504)}=2^{5292032}$	$2^{\wedge(1*661504)}=2^{661504}$
Rock	661794	16	8	$2^{\wedge(8*661794)}=2^{5294352}$	$2^{\wedge(1*661794)}=2^{661794}$

The large number of sample combinations in the proposed methods indicates that exhaustive search methods are impractical, demonstrating high resistance to brute-force attacks. When comparing the robustness of

the BCS and BCR algorithms with previous methods, it is evident that the innovative embedding techniques employed in these algorithms significantly increase resistance to brute-force attacks. Embedding multiple bits per sample in the proposed algorithms, as compared to a single bit in the LSB algorithm, increases the number of embedding locations and possible sample combinations, making detection more difficult.

Figure 12 illustrates a comprehensive comparison chart of performance metrics between the existing and proposed algorithms. The chart plots performance metric values on the y-axis for both the proposed and existing algorithms with performance metric names listed on the x-axis. The proposed algorithm aims for data concealment where the audio samples used should match the differing bits between the cover and stego audio, as demonstrated in Table 4 and Figure 13. The proposed BCS and BCR algorithms outperform existing ones with their experimental results, highlighting the significance of BER.

A Research gap and addressed novelty

The research gap in existing audio steganography methods lies in the limitations of the traditional Least Significant Bit algorithm. In the LSB algorithm, embedding one character from a text file requires eight audio samples, resulting in an embedding rate of only 12.5%. Using the 2LSBs to embed a character reduces this to four audio samples, achieving a 25% embedding rate. However, as the hiding capacity increases, the number of audio samples required also increases, highlighting the need for an algorithm that achieves higher embedding rates while minimizing the use of audio samples. Additionally, the BER in LSB methods is highly variable, depending on the characteristics of the audio file, leading to unpredictable error rates that may either be high or low. As embedding rates increase, BER increases, further impacting data integrity.

This variability emphasizes the need for an algorithm that maintains a consistent and predictable BER, regardless of the embedding rate and audio file characteristics.

The novelty of the proposed algorithm lies in balancing embedding capacity and BER by embedding a character (8 bits) from a text file within a single audio sample and ensuring that the bit difference between the samples in cover audio and the stego audio file is limited to one. The proposed method reduces the required audio samples while achieving a controlled error pattern.

B Strength of the proposed work

The proposed work embeds an entire character (8 bits) within a single audio sample, maintaining a 1:1 ratio between characters in the text file and audio samples.

A comparison methodology involves embedding text files into the audio samples. Hence the algorithm is resistant to statistical and brute-force attacks, which enhances the security and robustness of the algorithm. This mechanism results in a controlled one-bit difference between the sample of a cover audio file and the corresponding embedded sample in the stego audio file. This approach preserves the integrity of the audio file. Additionally, the algorithm predicts error bits between the cover and stego files, maintaining an effective trade-off between embedding capacity and error rate.

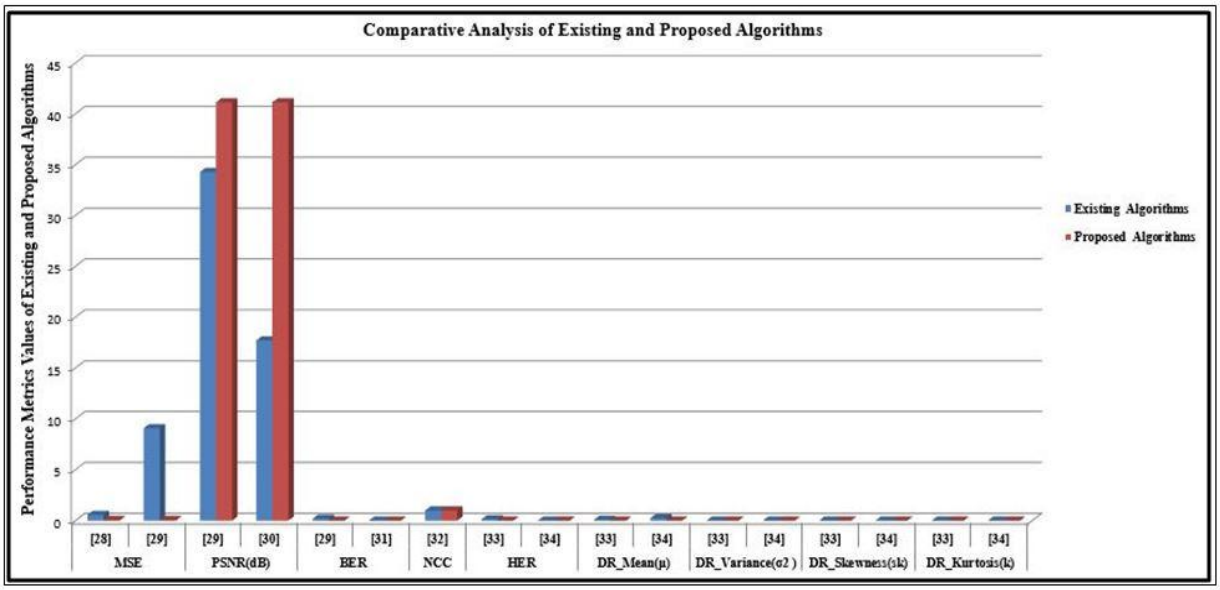


Figure 12: Comparative analysis of proposed and existing algorithms.

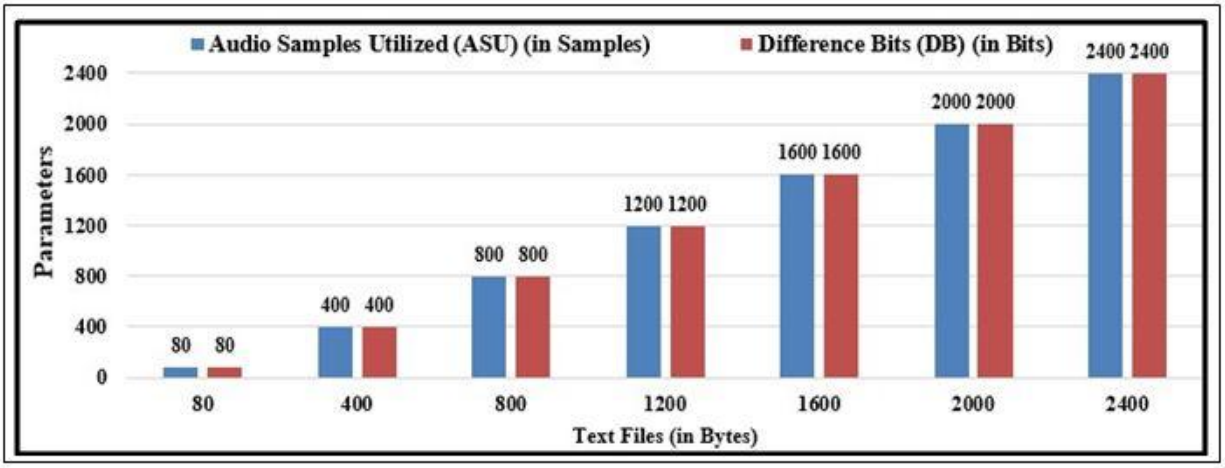


Figure 13: Influence of parameters of audio files versus hiding capacity.

This capability optimizes data hiding by minimizing audio quality degradation, addressing challenges in audio steganography.

4 Conclusion and future work

The research presented BCS and BCR algorithms to enhance the Least Significant Bit Algorithm in Audio Steganography. The BCS algorithm embeds each byte of the text file into the least significant 8 bits (one byte) of a two-byte audio sample from AF1, and the BCR algorithm retrieves the embedded data. Simultaneously, the Least Significant Bit algorithm embeds the index value of the samples from the Stego Audio File (SAF1) with hidden text file into the Audio File (AF2). Moreover, the proposed method was analyzed through MSE and PSNR similarity tests, yielding an MSE of 0.0513 and an average PSNR of 41.13732041 dB, thereby outperforming the existing algorithm. Furthermore, there is a 37.5% enhancement in the embedding rate compared to the conventional LSB algorithm. The stego signal is evaluated using histogram distribution, yielding an average HER of 4.33E-09. Specifically, the proposed algorithm's comparative methodology yielded identical values with an embedded text file of 2400 characters (2400 bytes) in 2400 audio samples, resulting in a 2400-bit difference between the cover audio file and the stego file. For future research, semiconductors could implement real-time audio steganographic systems.

Acknowledgement

The authors thank the anonymous reviewers for their insightful feedback. Their suggestions significantly improve the quality of this research article.

Data availability statement

Music audio files:
https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification
Text File:
http://www.daviddlewis.com/resources/testcollections/reuters21578/feldman-cia-worldfactbook-data.txt
Speech Audio Files:
https://www.kaggle.com/datasets/lnicalo/gtzan-musicspeech-collection?select=speech_wav
Ambient Audio Files:
https://mixkit.co/free-sound-effects/

References

[1] M. M. Mahmoud, and H. T. Elshoush.Enhancing LSB Using Binary Message Size Encoding for High Capacity, Transparent and Secure Audio Steganography-An Innovative Approach.IEEE Access, 10:29954–29971, 2022. https://doi.org/10.1109/ACCESS.2022.3155146

[2] S. A. S. Almola, N. H. Qasim, and H. A. A. Alasadi.Robust Method for Embedding an Image Inside Cover Image Based on Least Significant Bit Steganography.Informatica, 46(9):53–60,2022. https://doi.org/10.31449/inf.v46i9.4362

[3] F. R. Shareef.A novel crypto technique based ciphertext shifting.Egyptian Informatics Journal,21(2):83–90,2020. https://doi.org/10.1016/j.eij.2019.11.002

[4] S. A. S. Almola.Hiding Images in the Spatial Domain.Informatica,46(8):165–174,2022. https://doi.org/10.31449/inf.v46i8.4252

[5] Yingnan Zhang, Minqing Zhang ,Xu An Wang,Ke Niu, and Jia Liu.A Novel Video Steganography Algorithm Based on Trailing Coefficients for H.264/AVC. Informatica, 40(1):63–70,2016. https://www.informatica.si/index.php/informatica/article/view/1095/824

[6] M. Bazyar, and R. Sudirman.A new method to increase the capacity of audio steganography based on the LSB algorithm.Jurnal Teknologi (Sciences & Engineering), 74(6): 49–53, 2015. https://doi.org/10.11113/jt.v74.4667

[7] W. Fraczek, W. Mazurczyk, and K. Szczypiorski.Multilevel steganography: Improving hidden communication in networks.Journal of Universal Computer Science, 18(14):1967–1986, 2012. https://doi.org/10.3217/jucs-018-14-1967

[8] A. A. Alsabhany, A. H. Ali, F. Ridzuan, A. H. Azni, and M. R. Mokhtar.Digital audio steganography: Systematic review, classification, and analysis of the current state of the art.Computer Science Review, 38:100316,2020. https://doi.org/10.1016/j.cosrev.2020.100316

[9] A. Tahseen Suhail, and H. Ghanim Ayoub. A new method for hiding a secret file in several WAV files depends on circular secret key.Egyptian Informatics Journal, 23(4):33–43, 2022. https://doi.org/10.1016/j.eij.2022.06.003

[10] Adeboje Olawale Timothy, Adetunmbi Adebayo Olusola, and Gabriel Arome Junior. Embedding Text in Audio Steganography System using Advanced Encryption Standard, Text Compression and Spread Spectrum Techniques in Mp3 and Mp4 File Formats.International Journal of Computer Applications.177(41):46–51, 2020. https://doi.org/10.5120/ijca2020919914

[11] L. Chen, R. Wang, D. Yan, and J. Wang. Learning to Generate Steganographic Cover for Audio Steganography Using GAN.IEEE Access.9:88098–88107,2021. https://doi.org/10.1109/ACCESS.2021.3090445

[12] W. Al-chaab, Z. A. Abduljabbar, E. W. Abood, V. O. Nyangaresi, Hussein M. Mohammed , and Junchao Ma.Secure and Low-Complexity Medical Image Exchange Based on Compressive Sensing and LSB Audio Steganography.Informatica,47(6): 65–74,2023. https://doi.org/10.31449/inf.v47i6.4628

[13] Z. N. Sultani, and B. N. Dhannoon.Image and audio steganography based on indirect LSB.Kuwait Journal of Science,48(4):1–12, 2021. https://doi.org/10.48129/kjs.v48i4.8992

[14] Y. Yang, H. Yu, X. Zhao, and X. Yi.An adaptive double-layered embedding scheme for MP3 steganography. IEEE Signal Processing Letters,27:1984–1988,2020. https://doi.org/10.1109/LSP.2020.3032875

[15] K. Chen, H. Zhou, W. Li, K. Yang, W. Zhang, and N. Yu.Derivative-Based Steganographic Distortion and its Non-additive Extensions for Audio.IEEE Transactions on Circuits and Systems for Video Technology,30(7):2027–2032,2020. https://doi.org/10.1109/TCSVT.2019.2918511

[16] Y. Yang, X. Yi, X. Zhao, and J. Zhang.A fast and secure MP3 steganographic scheme with multi-domain.Signal Processing,190:108332, 2022. https://doi.org/10.1016/j.sigpro.2021.108332

[17] Y. Ren, S. Cai, and L. Wang.Secure AAC steganography scheme based on multi-view statistical distortion (SofMvD). Journal of Information Security and Applications, 59:102863,2021. https://doi.org/10.1016/j.jisa.2021.102863

[18] K. Ying, R. Wang, Y. Lin, and D. Yan.Adaptive Audio Steganography Based on Improved Syndrome-Trellis Codes.IEEE Access,9:11705–11715,2021. https://doi.org/10.1109/ACCESS.2021.3050004

[19] F. Djebbar, B. Ayad, K. A. Meraim, and H. Hamam.Comparative study of digital audio steganography techniques.EURASIP Journal on Audio, Speech, and Music Processing, 25:1–16,2012. https://doi.org/10.1186/1687-4722-2012-25

[20] P. Marszałek, and P. Bilski.Steganography in Audio Files – COTS Software Analysis. International Journal of Electronics and Telecommunications,69(1),121-126,2023. https://doi.org/10.24425/ijet.2023.144340

[21] Gams. M, and Kolenik. T. Relations between Electronics, Artificial Intelligence and Information Society through Information Society Rules.Electronics, 10(4), 514:1-16, 2021. https://doi.org/10.3390/electronics10040514

[22] Imane Zerraza.Lightweight Authentication for IOT Edge Devices.Informatica,48(18),15–20,2024. https://doi.org/10.31449/inf.v48i18.6012

[23] W. A. Awadh, A. S. Hashim, and A. K. Hamoud.Efficiently Secure Data Communications Based on CBC-RC6 and the Overflow Field of Timestamp Option in an IPv4 Packet.Informatica,46(6):125–133, 2022. https://doi.org/10.31449/inf.v46i6.4005

[24] Md. R. Rahman, P. Chakraborty, Md. Z. Rahman, and Md. G. Moazzam.Hiding Confidential File using Audio Steganography.International Journal of Computer Applications,178(50):30–35, 2019. https://doi.org/10.5120/ijca2019919422

[25] X. Zhang, S. Wang, and W. Zhang.Steganography Combining Data Decomposition Mechanism and Stego-coding Method.Informatica,33(1):41–48,2009. https://www.informatica.si/index.php/informatica/article/view/222/219

[26] M. H. Mohammed.A new approach to hide texts into images and audio files using steganography and cryptography techniques.International Journal of Advances in Applied Sciences (IJAAS),11(4):312–323,2022. https://doi.org/10.11591/ijaas.v11.i4.pp312-323

[27] Orora Tasnim Nisha, Md. Selim Hossain, and Mahfujur Rahman.Audio Steganography with Intensified Security and Hiding Capacity. European Chemical Bulletin,12(10):162–173,2023. https://www.eurchembull.com/archives/volume-12/issue-10/8322

[28] R. Chowdhury, D. Bhattacharyya, Samir Kumar Bandyopadhyay, and Tai-hoon Kim .A View on LSB Based Audio Steganography.International Journal of Security and Its Applications ,10(2),51-62,2016. https://doi.org/10.14257/ijsia.2016.10.2.05

[29] B. Harjito, B. Sulistyarso, and E. Suryani, .Audio steganography using two LSB modification and RSA for security data transmission.Journal of Telecommunication, Electronic and Computer Engineering,10(2–4):107–111, https://jtec.utem.edu.my/jtec/article/view/4326/3173

[30] M. Parthasarathi, and T. Shreekala.Secured Data

Hiding in Audio Files Using Audio Steganography Algorithm.International Journal of Pure and Applied Mathematics, 116(21):619–628, 2017. https://acadpubl.eu/jsi/2017-116-13-22/articles/21/79.pdf

[31] Vipul Sharma, and Ravinder Thakur.LSB Modification based Audio Steganography using Trusted Third Party Key Indexing Method.IEEE 2015 Third International Conference on Image Information Processing(ICIIP), 403–406, 2015. https://doi.org/10.1109/ICIIP.2015.7414805

[32] A. H. Ali, M. R. Mokhtar, and L. E. George .Enhancing the hiding capacity of audio steganography based on block mapping. Journal of Theoretical and Applied Information Technology,95(7):1441–1448,2017. https://www.jatit.org/volumes/Vol95No7/13Vol95No7.pdf

[33] 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 and Applications, 77(23):31487-31516,2018. https://doi.org/10.1007/s11042-018-6213-0

[34] H. I. Shahadi, R. Jidin, and W. H. Way.Lossless audio steganography based on lifting wavelet transform and dynamic stego key.Indian Journal of Science and Technology,7(3):323–334, 2014. https://doi.org/10.17485/ijst/2014/v7i3.14

[35] K. Bhowal, D. C. Sarkar, S. Biswas, and P. P. Sarkar.A steganographic approach to hide secret data in digital audio based on XOR operands triplet property with high embedding rate and good quality audio .Turkish Journal of Electrical Engineering & Computer Sciences,25(3):2136–2148,2017. https://doi.org/10.3906/elk-1602-267

[36] H. T. Elshoush, and M. M. Mahmoud.Ameliorating LSB Using Piecewise Linear Chaotic Map and One-Time Pad for Superlative Capacity, imperceptibility and Secure Audio Steganography.IEEE Access,11:33354–33380,2023. https://doi.org/10.1109/ACCESS.2023.3259902

[37] Prolog fact base about countries derived from the 1994 CIA World Factbook, Reuters-21578 dataset [Online]. http://www.daviddlewis.com/resources/testcollections/reuters21578/feldman-cia-worldfactbook-data.txt.

Appendix A: Performance of the novel proposed algorithm on speech audio files.

Table 9: Specifications of speech audio files.

Audio File	Sampling Rate	Duration(s)	Bits per sample	Total Samples	Channel
dialogue.wav	22050	30	16	661500	Mono
news2.wav	22050	30	16	661500	Mono
india.wav	22050	30	16	661500	Mono

Table 10: Summary of performance metrics of the proposed BCS and BCR algorithms over the speech audio files.

Text File (Bytes)	Audio File	MSE	PSNR (dB)	BER	NCC	HER	Difference Ratio			
							Mean (μ)	Variance (σ^2)	Kurtosis (k)	Skewness (sk)
80	dialogue.wav	3.7735E-10	51.2156	7.5586E-06	1	7.4654E-11	1.2179E-06	1.0904E-06	6.4684E-07	3.4098E-05
800		2.9894E-09	41.2156	7.5586E-05	1	2.3516E-09	2.9968E-06	8.8253E-06	3.0053E-06	4.4714E-04
1600		6.111E-09	38.2053	1.5117E-04	1	1.5677E-09	1.7737E-05	1.7243E-05	1.1680E-05	5.4327E-04
2400		9.7950E-09	36.4444	2.2676E-04	1	1.0377E-08	3.1769E-05	2.0424E-05	1.3245E-05	9.8719E-04

80	news2.wav	2.3138E-10	51.2156	7.5586E-06	1	7.1816E-11	4.9431E-06	2.1313E-07	4.2572E-07	3.1299E-5
800		2.6409E-09	41.2156	7.5586E-05	1	1.2568E-09	8.8116E-05	1.2225E-05	1.0849E-05	1.0091E-04
1600		5.4481E-09	38.2053	1.5117E-04	1	2.1006E-09	5.0011E-05	3.1264E-05	2.0823E-05	5.6029E-04
2400		8.4483E-09	36.4444	2.2676E-04	1	7.2893E-09	1.0810E-05	5.1305E-05	2.3609E-05	4.6241E-04
80	india.wav	3.1742E-10	51.2156	7.5586E-06	1	9.7261E-12	5.5306E-06	8.2757E-07	1.5051E-06	8.7075E-07
800		2.4231E-09	41.2156	7.5586E-05	1	3.8904E-11	6.3265E-06	1.2215E-06	2.2362E-06	1.3568E-06
1600		5.5930E-09	38.2053	1.5117E-04	1	1.6437E-09	2.7714E-05	2.5489E-05	6.1954E-05	6.6199E-05
2400		7.5263E-09	36.4444	2.2676E-04	1	3.0054E-09	1.5816E-05	3.1367E-5	4.453E-06	2.6017E-06

Table 11: Performance of proposed BCS Algorithm.

Length of Text File (in Bytes)	Audio File	Audio Samples Utilized (ASU) (in samples)	Difference Bits (DB) (in bits)
80	dialogue.wav news2.wav india.wav	80	80
800		800	800
1600		1600	1600
2400		2400	2400

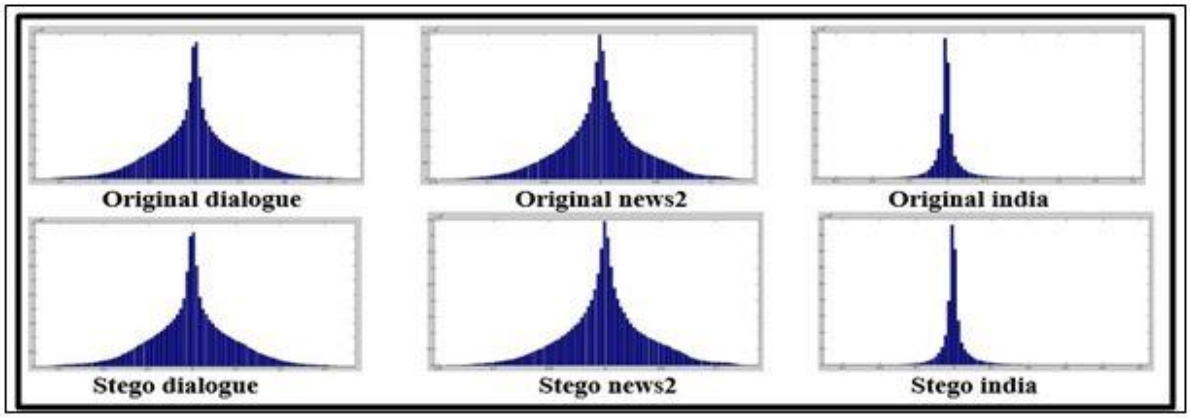


Figure 14: Histograms of cover and stego speech audio files with hidden text files of 2400bytes.

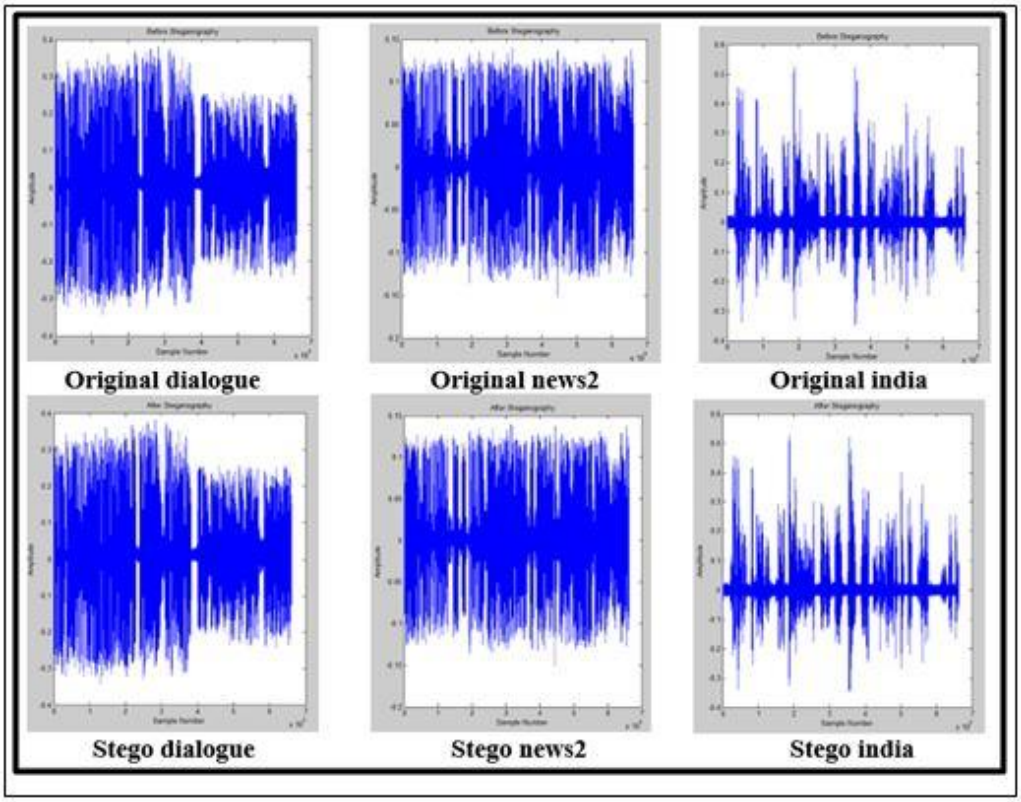


Figure 15: Cover and stego speech audio files with hidden text files of 2400 bytes.

Appendix B: Performance of the novel proposed algorithm on ambient audio files.

Table 12: Specifications of ambient audio files.

Audio File	Sampling Rate	Duration (s)	Bits per sample	Total Samples	Channel
Ambient AF1.wav	44100	29	16	1278900	2
Ambient AF2.wav	44100	15.3386	16	676432	2
Ambient AF2.wav	44100	51.7152	16	2280640	2

Table 13: Summary of performance metrics of the proposed BCS and BCR algorithms over the ambient audio files.

Text File (Bytes)	Audio File	MSE	PSNR (dB)	BER	NCC	HER	Difference Ratio			
							Mean (μ)	Variance (σ^2)	Kurtosis (k)	Skewness (sk)

80	Ambient AF1.wav	7.2556E-11	57.14359	1.9303E-06	1	0	1.1217E-04	4.4115E-08	8.4537E-08	6.6923E-06
800		5.9549E-10	47.08896	1.954E-05	1	0	1.0339E-04	4.5997E-08	8.8644E-08	6.1574E-06
1600		1.1511E-09	44.08138	3.9071E-05	1	3.9206E-13	1.1897E-04	4.6446E-07	9.1644E-07	7.7408E-06
2400		1.8947E-09	42.31775	5.8644E-05	1	3.9206E-13	2.4504E-04	2.0188E-06	4.0948E-06	1.1244E-04
80	Ambient AF2.wav	1.4504E-10	54.32284	3.6959E-06	1	2.3782E-12	6.1081E-04	5.2939E-07	8.2397E-07	4.3568E-05
800		1.6812E-09	44.32284	3.6959E-05	1	1.6648E-11	0.00107564	3.6071E-06	4.9900E-06	3.7829E-05
1600		2.8646E-09	41.31254	7.3917E-05	1	7.3725E-11	0.00144827	6.4778E-06	9.4422E-06	4.0325e-04
2400		4.1733E-09	39.55162	1.1088E-04	1	1.7836E-10	2.6507E-04	6.2648E-06	9.0433E-06	1.7764E-04
80	Ambient AF3.wav	4.0455E-11	59.60116	1.0962E-06	1	0	0.01618593	1.5061E-09	2.8664E-09	1.5991E-06
800		3.6637E-10	49.60116	1.0962E-05	1	1.9251E-13	0.08072214	1.2254E-07	2.4212E-07	7.5774E-06
1600		7.5242E-10	46.59086	2.1924E-05	1	0	0.07989209	5.8145E-07	1.1329E-06	5.2409E-06
2400		1.1499E-09	44.82995	3.2886E-05	1	1.9251E-13	1.02842913	9.6143E-07	1.7830E-06	7.4944E-05

Table 14: Performance of proposed BCS algorithm.

Length of Text File (in Bytes)	Audio File	Audio Samples Utilized (ASU) (in samples)	Difference Bits (DB) (in bits)
80	Ambient AF1.wav Ambient AF2.wav Ambient AF3.wav	80	80
800		800	800
1600		1600	1600
2400		2400	2400

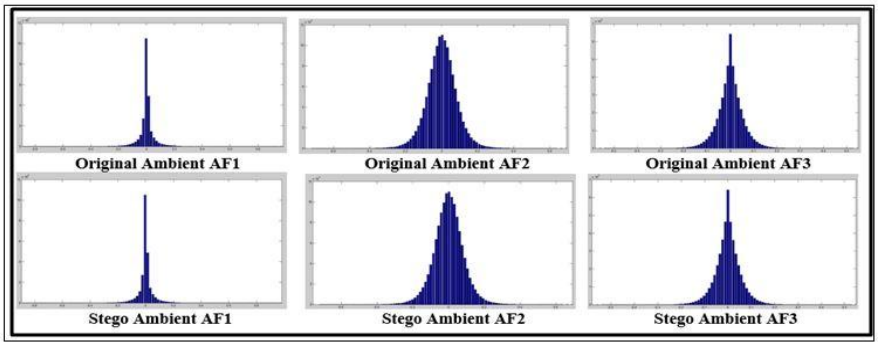


Figure 16: Histograms of cover and stego ambient audio files with a hidden text file of size 2400bytes.

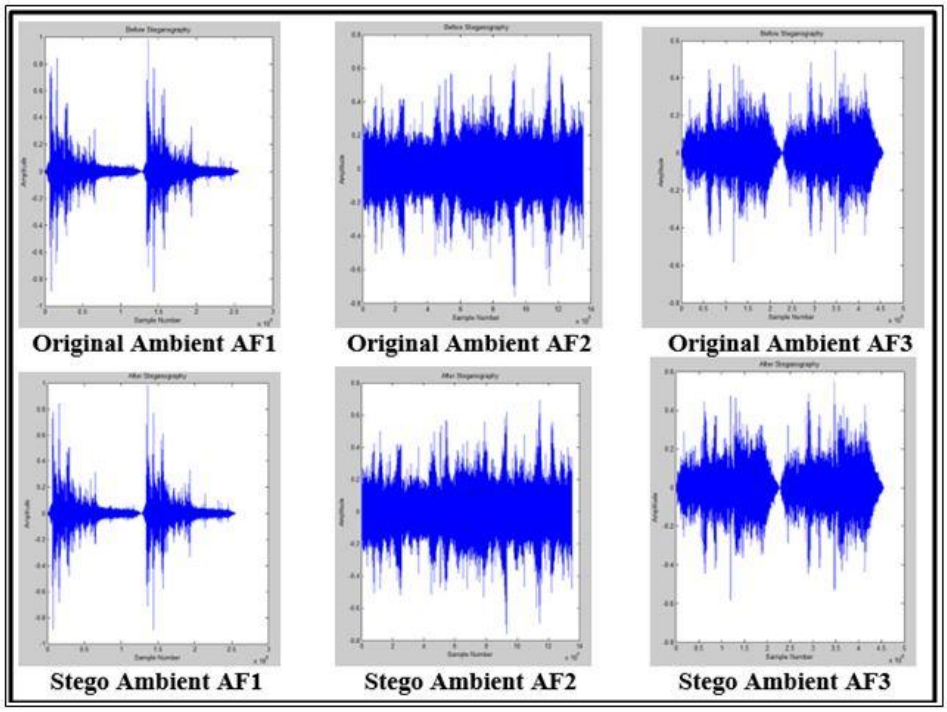


Figure 17: Cover and stego ambient audio files with a hidden text file of size 2400bytes.