Audio Steganography for Healthcare Data Using 24x8 Compression and Bit Comparison Substitution Algorithm

K Revathi, and S. Kaja Mohideen*
Department of Electronics and Communication Engineering, B.S.Abdur Rahman Crescent Institute of Science and Technology, Vandalur, Chennai, Tamil Nadu, India
E-mail: revathivigneswaranphd@gmail.com, kajamohideen@crescent.education.
*Corresponding Author

Keywords: Audio steganography, least significant bit algorithm (LSB), run length encoding (RLE) compression, text files, encryption, decryption, waveform audio file format

Received: July 3, 2025

Audio steganography embeds multimedia files within audio files. Traditional audio steganography used the Least Significant Bit algorithm, which required eight samples to embed one byte of a text file. This study reduces the utilization of audio samples by compressing healthcare data from three characters to one using the proposed 24×8 compression algorithm. Three audio files were used to transmit the probability distributions, encoded values, and index values. The encoded values are embedded using the Bit Comparison and Substitution-3 algorithm, with a three-bit difference from audio samples. It is retrieved using the Bit Comparison and Retrieval-3 algorithm and decompressed with the 8×24 decompression algorithm. For enhanced security, healthcare data was encrypted using the Incremental Order Value Algorithm and decrypted with the Decremental Reverse Order Value Algorithm. The least significant bit algorithm embeds the probability distributions and index values with a secret key. Audio files from Mixkit and healthcare data from the COVID Dialogue Dataset were used for evaluation. The proposed algorithms achieved an average throughput of 8592.74 KB/s, surpassing the 3-DES algorithm due to the incremental shift in ASCII values within healthcare data. A compression ratio of 3:1 was achieved by compressing 3 bytes of data to 1, outperforming Huffman and LZW. The embedding algorithm achieved a PSNR of 42.1480 dB and a BER of 7.0165×10^{-5}, demonstrating improved efficiency with reduced audio samples in embedding compared to the traditional LSB algorithm. 15000 bytes of healthcare data were embedded into 5000 audio samples, resulting in 15000-bit differences between the cover and stego audio files.

Povzetek: Študija predlaga zvočno steganografijo za zdravstvene podatke, ki z lastnim 24×8 stiskanjem in (šifriranim) vgrajevanjem prek bitne primerjave/substitucije zmanjša porabo vzorcev in ob ohranjeni kakovosti zvoka omogoči učinkovitejše skrivanje kot klasični LSB.

1 Introduction

The COVID-19 pandemic, combined with advanced technology, has highlighted the significance of telemedicine, enabling patients to consult medical experts wirelessly. Rising healthcare expenses have also driven the need for telemedicine, increasing communication between doctors and between doctors and remote patients. Telemedicine services, such as video conferences and web-based visits, provide remote healthcare. This approach can make healthcare more efficient, centralized, and accessible.

With telemedicine, patients receive services for distant healthcare, critical situations, medication management, and chronic health conditions. It also lowers the risk of hospital infections for patients with weak immune systems. Additionally, several resources, such as self-examinations, step-by-step training programs customized for their conditions, and email-based symptom descriptions, help patients communicate their symptoms to healthcare providers. Monitoring apps and smartphones assist in the care of patients with chronic illnesses. Despite its benefits, telemedicine poses risks to the security of healthcare data when using public networks. Hence, protecting healthcare data is a primary concern. Cryptography and steganography are information-hiding techniques for secure communication of healthcare data. These information-hiding techniques, along with IoT devices and artificial intelligence (AI), enable the successful transmission of healthcare data [1], [2], [3].

This research focused on combining cryptography and steganography to ensure double-layered security for communicating healthcare data.

Cryptography converts plaintext into ciphertext using encryption keys. The plaintext in this research denotes unique healthcare data (HCD) transmitted to the receiver, and the ciphertext represents its unreadable form.

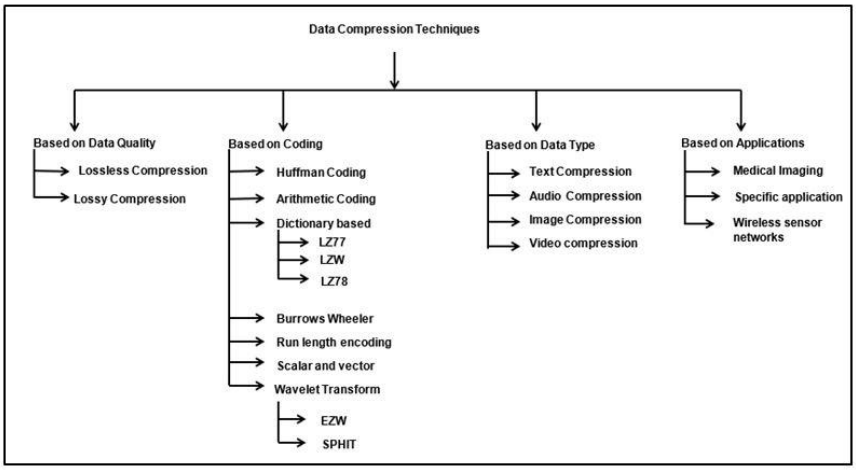


Figure 1: Data compression techniques

Various encryption algorithms and keys convert the medical data into ciphertext. The encryption algorithms include the Advanced Encryption Standard (AES), Data Encryption Standard (DES), Triple DES (3DES), and the International Data Encryption Algorithm (IDEA). At the receiver's end, the decryption algorithm and keys decrypt the ciphertext to plain text. These algorithms are selected based on capacity, security, and reliability [4], [5], [6].

In the proposed work, healthcare data is encrypted into a ciphertext using the Incremental Order Value Algorithm (I-OVA). The Decremental Reverse Order Value Algorithm (D-RVA) decrypts the unreadable form of healthcare data.

Cryptography provides the first layer of security by transforming plain healthcare data into ciphertext.

The second layer of healthcare data security is steganography, which conceals the existence of HCD during communication. Steganography embeds multimedia files within other multimedia files that serve as carriers. These files include audio, video, image, and text formats. Audio carriers are highly efficient due to the human auditory system (HAS) being less sensitive to slight variations in sound than the human visual system (HVS). Audio steganography specifically embeds multimedia files in audio carriers, characterized by high capacity, robustness, and imperceptibility. The maximum amount of data hidden in a cover audio file without affecting its overall quality is known as capacity. Imperceptibility refers to the ability to alter an audio file without compromising its stego audio quality. Robustness is the strength of steganographic communication and data retrieval from the stego audio file. There is a trade-off between these characteristics of audio steganography. Increased embedding capacity often decreases robustness and imperceptibility. The proposed work addresses this trade-off by compressing the healthcare data before embedding it in audio files [7], [8], [9].

Compression is the process of reducing the data size from its original size. The main objectives of multimedia data compression are to reduce transmission time, network bandwidth, and storage volume.

Various compression techniques apply to text, images, videos, and other digital data types. According to the literature, data compression involves encoding, while data decompression is decoding. Decompression restores the compressed data to its original form [10], [11]. Data compression can be classified into two types: lossy and lossless compression. Figure 1 shows the various data compression techniques [11]. Lossless compression is applied when the accuracy of the original data is essential. In lossy compression, data portions are removed by preserving the potential of compression [12]. This research combines double-layered security with lossless compression of encrypted healthcare data using the 24x8 compression algorithm. The Run Length Encoding (RLE) algorithm compresses the probability distribution, a simple form of lossless data compression, and saves data sequences with the same value into a data file.

A comprehensive review of related works details the literature on telemedicine, information-hiding techniques such as cryptography and its algorithms, and audio steganography with its characteristics. The proposed work describes the embedding and extraction algorithms, which are subsequently analyzed and evaluated in the results and discussion section. Theoretical analysis examines these algorithms in depth. Finally, the paper concludes in the conclusion and future scope section.

1.1 Problem statements

The literature survey identifies a significant research gap in securely communicating healthcare data (HCD) using audio steganography in telemedicine.

1. Telemedicine is an emerging area of research with compromised data security and efficiency.
2. Ensuring the security of healthcare data (HCD) in remote consultations is essential.
3. Currently, telemedicine practices image steganographic communication over audio steganography, leaving the latter underexplored despite its potential.
4. Most conventional audio steganography methods, including those based on LSB, do not incorporate compression before embedding, as referred to in Table 1, leading to inefficient use of audio capacity.
5. Existing compression methods result in variable-length encoded data and require transmission of

codebooks or dictionaries for reconstruction. This results in increased overhead and limits their suitability for fixed-size, low-latency audio embedding of compressed data. A fixed-length compression scheme with minimal computational and embedding overhead is required to enhance efficiency in audio steganographic systems.

6. The LSB algorithm has an embedding rate of 12.5%, with 1 byte of a text file embedded into eight audio samples of 1 byte. This introduces a trade-off between embedding capacity and audio sample utilization. Increasing the embedding capacity requires additional audio samples, reducing the available samples for further embedding.
7. In the LSB algorithm, the error bits vary inconsistently with the bit pattern of the audio samples and the embedded text, especially as the embedding capacity increases. This inconsistency limits its effectiveness. There is no consistent relationship between error bits, the embedded text file, and the audio samples.

In response to these challenges, this work proposes effective embedding and extraction algorithms that combine cryptography and steganography for secure communication of compressed healthcare data in telemedicine.

1. I-OVA and D-RVA: The Incremental Order Value algorithm (I-OVA) and Decremental Reverse Order Value algorithm (D-RVA) to encrypt and decrypt healthcare data, ensuring data security.
2. Merging algorithm (MA): Three 8-bit characters are combined into a 24-bit sequence, optimizing data representation.
3. The probability distribution (PD) compression: RLE compresses redundant probability distribution data using value-count pairs, eliminating redundancy, reducing data size, and enabling efficient embedding with minimal retrieval overhead.
4. 24x8 compression algorithm: Cipher Healthcare data (C-HCD) of 24 bits is compressed into an 8-bit encoded value to optimize embedding and is subsequently reconstructed using the 8×24 decompression algorithm.
5. BCS-3 and BCR-3: The Bit Comparison and Substitution-3 (BCS-3) and Bit Comparison and Retrieval-3 (BCR-3) algorithms embed and retrieve the encoded values in Audio File 2 (AF2), ensuring secure embedding with efficient utilization of audio samples.
6. The LSB algorithm embeds and extracts the RLE count and its values for probability distribution in Audio File 1(AF1) and the Index value of Stego Audio Samples (SAF2) in Audio File 3(AF3) with secret key Ks as a reference.

This study investigates the secure communication of compressed healthcare data using double-layer security by integrating cryptographic encryption with efficient steganographic embedding. The proposed design uses a 24×8 compression algorithm, BCS-3 embedding, and cryptographic encryption to reduce healthcare data size and the number of audio samples needed for embedding.

It is hypothesized that this integration ensures consistent error control and accurate data recovery while minimizing retrieval overhead. Simulations show that 15,000 bytes of data were compressed to 5,000 encoded bytes (1 byte each), improving the compression ratio and optimizing audio sample utilization.

The framework of the proposed method is shown in Figure 2, which is more self-explanatory. Table 1 summarizes the related works.

2 Related works

A detailed study on audio steganography discussed techniques such as linear and sequential methods, selective embedding, frequency masking and thresholding, error minimization-based embedding, pattern matching, phase coding, and spread spectrum methods [13]. The merits, demerits, and applications of audio steganography in medicine, transportation, and government domains were discussed. Additionally, they highlighted the need to explore a broader range of steganography methods for transmitting healthcare data. It was also noted that research should use various cover media like audio, video, and text, as 7 out of 9 articles utilized image steganography [14].

The traditional LSB algorithm for audio steganography is enhanced to increase embedding capacity and security. A robust hybrid steganographic system was proposed in [15], where text messages are initially encrypted using bit cycling operations and embedded in randomly selected bits of an audio file.

However, increased embedding capacity also raises the risk of steganalysis. So, advanced symmetric and asymmetric cryptography algorithms like AES, DES, 3DES, Blowfish, IDEA, RSA, ECC, and ElGamal are used to strengthen security [16].

Recent research focuses on providing double-layer security to embedded data by combining cryptography with steganography. The study [17] addresses the critical challenge of securely transmitting sensitive medical data, such as COVID-19-related information, with a novel crypto-steganography-based scheme, incorporating hybrid cryptography for encryption, random block, and pixel selection for enhanced security, and an inversion method for efficient data embedding.

The paper [18] studied the security of the modern steganography method using a lossless compression called arithmetic coding. Integrating compression techniques in audio steganography enhances embedding capacity and security.

Various compression algorithms, such as run-length encoding, Huffman Coding, and arithmetic coding, highlight their efficiency, merits, demerits, and corresponding applications. They also explored different types of data compression, including text, audio, video,

Table 1: Technical overview of methods and results [25]

Reference No.	Methodology	Inferences / Results
[26]	Steganography: Audio Carrier Medium: Audio Concealed Data: Text Steganographic Technique: Two LSB modification Cryptographic Technique: RSA.	Embedding Efficiency (Sample-to-Byte Ratio): 4:1 Payload:107 bytes MSE = 9.08875 Peak Signal to Noise (PSNR) = 34.2775dB Bit Error Rate (BER) =17.995%.
[27]	Steganography: Audio Carrier Medium : Audio Concealed Data: Text Steganographic Technique: The Fourth and Fifth LSBs of audio samples are embedded with encrypted data.	A PSNR of 17.71916498 dB was achieved after embedding the message into a JAZZ audio file. Payload:103 bytes Embedding Efficiency(Sample-to-Byte Ratio): 4:1
[28]	Steganography: Audio Carrier Medium : Audio Concealed Data: Text Steganographic Technique : Duel Encryption-LSB algorithm Cryptographic Technique : Duel Encryption-Pattern Matching Algorithm	A Mean Squared Error of 0.5619 resulted in Pattern Matching method in embedding target text of varying sizes, small, medium, and large in audio file cover1.wav and cover2.wav Payload: 250 bytes Normal LSB method : Sample-to-Byte Ratio: 8:1 Embedding rate =12.5%. MSE = 0.3994
[29]	Steganography: Audio Carrier Medium : Audio Concealed Data: Text Steganographic Technique: DWT for Embedding. Cryptographic Technique : Dynamic encryption algorithm	MSE of 0.8578 resulted in embedding Test5.txt in Two.wav audio file. Payload :9405 bytes.
[30]	Steganography: Audio Carrier Medium : Audio Concealed Data: Text Steganographic Technique :Random Key Indexing method (Trusted Third Party Stego key and Secondary Key generated at encoder) Cryptographic Technique: AES-256.	The right channel of a 16-bit stereo audio file resulted in a BER of 0.0187% after embedding 532 bytes of message data.
[31]	Steganography: Audio Carrier Medium : Audio Concealed Data: Audio Steganographic Technique: ECA-BM ,Fractal coding and chaotic LSB	The Normalized Correlation of the retrieved file is 0.99992. Hiding Capacity:80%
[32]	Steganography: Audio Carrier Medium : Audio Concealed Data: Audio Steganographic Technique : HASFC, Fractal coding and Chaotic least significant bit	Hiding Capacity:100% Histogram error rate (HER) = 0.1278 Difference Ratio (DR) of the fourth first moments: mean: 0.0799, variance: 0.0008, skewness: 0.0018, kurtosis: 0.0028
[33]	Steganography: Audio Carrier Medium : Audio Concealed Data: Image and Audio Steganographic Technique : Integer-to-Integer Lifting Wavelet Transform (Int2Int LWT) and Least Significant Bits (LSBs) substitution Cryptographic Technique: Adaptive steganography key (Stego key) for encryption.	Hiding Capacity:25% HER = 1.2274e-04 Difference Ratio (DR) of the first fourth moments: mean: 0.2304, variance: 0, skewness: 0.0004, kurtosis: 0.0005

and image. Text data is compressed using algorithms such as Burrows-Wheeler Transform and Boolean minimization, logical truth table, and graph-based method [11]. Authors [19] proposed compressive sensing and

audio steganography techniques to secure and compress medical images. Nathaniel Fout et al. highlighted the difficulty of compressing floating-point data compared to integer data. Lossless compression methods for scientific and medical floating-point volume data utilizing prediction-based techniques like the Adaptive Polynomial Encoder (APE) and Adaptive Combined Encoder (ACE) are proposed [20].

In medical image steganography, the medical image is decomposed using an integer wavelet filter, compressed with arithmetic coding, and encrypted with DES. The secret bits are then embedded in the LSB of high-frequency coefficients in the wavelet domain [21].

The pandemic situation has raised the need for telemedicine [22]. The authors [23] proposed a speech-to-speech workflow (STSW), a deep learning-based framework designed for multilingual telemedicine. Information-hiding techniques are an emerging area of research for the secure transmission of medical images for telemedicine [24]. This paper explores the integration of telemedicine with information-hiding techniques.

3 Proposed work

3.1 Embedding algorithm

3.1.1 Phase I: Incremental-order value algorithm (I-OVA)

Input: Healthcare data (HCD).
Output: Cipher healthcare data (C-HCD).

1. The 15000B HCD will be divided into sequential segments of 501 characters each, with the last segment containing the remaining characters.
2. Assign decimal Order Values (OV) ranging from 127 down to 32 to all printable ASCII characters (ASCII 32 to 127), such that OV 127 corresponds to ASCII 32, OV 126 corresponds to ASCII 33, and so on, forming a one-to-one mapping.
3. Assign an index value from 0 to 95 to each printable ASCII character (ASCII 32 to 127), representing its sequential position within the printable ASCII set. These indices serve as the numeric representation of the Cipher Healthcare Data (C-HCD).
4. Read the ASCII character codes for all printable characters in the HCD.
5. For each character in the HCD, treat its ASCII code as an Order Value (OV_input) and retrieve the corresponding OVA character using the OV-to-ASCII mapping (OV 127 → ASCII 32, OV 126 → ASCII 33, …, OV 32 → ASCII 127). This transformation produces the OVA-encrypted representation of the HCD.
6. Determine the ASCII codes of all OVA characters obtained in Step 5. These ASCII values serve as the basis for the subsequent incremental transformation in the I-OVA encryption process.
7. For each segment, increment the ASCII value of each OVA character by its position within the segment (e.g., first character by 1, second character by 2, and so on). If the incremented value exceeds 127, wrap around to 32 to ensure it remains within the printable ASCII range.
8. Convert the incremented ASCII values to their corresponding characters to obtain the I-OVA encrypted characters. Simultaneously, determine the index of each character within the printable ASCII set (Index 0–95) to represent the Cipher Healthcare Data (C-HCD).
9. Repeat Steps 5–8 for all characters in each segment to perform the complete I-OVA encryption of the Healthcare Data (HCD).
10. The indices of the incremented characters within the printable ASCII set (Index 0–95) represent the Cipher Healthcare Data (C-HCD).
11. End.

The metrics throughput and entropy verify the performance of the proposed I-OVA algorithm, as discussed in the Results and Discussion section.

3.1.2 Theoretical discussion of I-OVA

Table 2 presents the I-OVA to the HCD (id=1ht).
The first character of the HCD, ‘i’(ASCII 105), is its order value. The algorithm maps this order value to the character ‘6’(ASCII 54). Since ‘6’ is the first occurrence, its ASCII is incremented to 55, mapped to the character ‘7’ with index 23. Thus, ‘i’ is encrypted to ‘7’, and the same process applies to the other characters.

3.1.3 Phase II: Merging algorithm (MA)

Input: Cipher Healthcare data (C-HCD)
Output: Merged Cipher healthcare data(M-C-HCD) (24 bits)

1. Read the C-HCD in each segment and convert it into 8-bit binary streams.
2. Measure the length (L) of the C-HCD.
3. If L is divisible by three without a remainder, merge the three one-byte characters in the C-HCD to get the character of three bytes (24 bits).
4. If L is not divisible by 3, additional bytes of "00000000" are appended at the end of the C–HCD to ensure the remaining characters form complete sets of three for merging. If one byte remains, two bytes of "00000000" are added; if two bytes remain, one byte of "00000000" is added.
5. Repeat steps 3 or 4 to merge all the characters in each segment of C-HCD.
6. The merged cipher healthcare data (M-C-HCD) for 501-byte segments results in 167 × 24 bits.
7. End.

The simulation results of the merging algorithm are presented and discussed in Figure 11 in the Results and Discussion section

.

3.1.4 Theoretical discussion of MA

The output of the I-OVA algorithm, as shown in Table 2, is used as the input for the merging algorithm. The

decimal values of the I-OVA output are converted to 8-bit binary streams. Since six characters are processed, they are merged into groups of three characters per row, as shown in Table 3.

The 15000B HCD is divided into 30 segments, with the first 29 segments containing 501 characters each. The last segment holds the remaining 471 characters. Each segment is then arranged into three characters (24 bits) per row.

3.1.5 Phase III: 24x8 compression algorithm

Input: M-C-HCD
Output: Probabilities and Encoded value

Step 1: Initialization
Initialize an array named probabilities with a length of 16,777,216, setting all elements to zero.
Initialize index_array ← empty
Initialize encoded_binary_array ← empty
Step 2: Index Construction and Event Assignment
For each segment in M-C-HCD:
Convert the 24-bit binary sequence to its decimal equivalent: index ← bin2dec(segment)
Increment the decimal index by 1: index ← index + 1. In the probability distribution, represent the occurrence at this index as one (1): probabilities[index] ← 1
Append the index to the index_array

Step 3: Normalization
Compute the total sum of all values in the probabilities array: Total ← sum(probabilities)
Step 4: Encoded Value Calculation
For each index in index_array:
Compute the probability value: P ← probabilities[index] / Total
Compute the sum of probabilities up to the element preceding the index:
C ← sum(probabilities[1 to index - 1]) / Total
Calculate the encoded value:
encoded_value ← C + (P / 2)
Scale the encoded value to the 8-bit binary range:
scaled_value ← round(encoded_value × 255)
Convert the rounded value to 8-bit binary:
binary_8bit ← dec2bin(scaled_value, 8)
Store the binary representation in the encoded_binary_array
Step 5: Output Return probabilities, encoded_binary_array

The algorithm compresses 24-bit M-C-HCD to an 8-bit encoded value using the 24x8 compression algorithm. The proposed 24×8 compression algorithm encodes 15000B of HCD into 5000 values, with each row in every segment contributing to an encoded value.

As discussed in the Results and Discussion section, the efficiency of the proposed algorithm is evaluated using compression ratio and space savings.

Table 2: Encryption of HCD using I-OVA algorithm

S:NO	HC Data [step 1]	ASCII Code [step 4]	Order Value [step 5]	OVA Character [step 5]	ASCII Code [step 6]	I-OVA Value [step 7]	I-OVA Character [step 8]	Index: I-OVA Character (C-HCD) [step 3 & 8]
1	i	105	105	6	54	54+1=55	7	23
2	d	100	100	;	59	59+2=61	=	29
3	=	61	61	b	98	98+3=101	e	69
4	1	49	49	n	110	110+4=114	r	82
5	h	104	104	7	55	55+5=60	<	28
6	t	116	116	+	43	43+6=49	1	17

Table 3: Merging algorithm

C-HCD	Binary Value								Merged cipher healthcare (M-C-HCD) data																							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
23	0	0	0	1	0	1	1	1																								
29	0	0	0	1	1	1	0	1	0	0	0	1	0	1	1	1	0	0	0	1	1	1	0	1	0	1	0	0	0	1	0	1
69	0	1	0	0	0	1	0	1																								
82	0	1	0	1	0	0	1	0																								
28	0	0	0	1	1	1	0	0	0	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	1
17	0	0	0	1	0	0	0	1																								

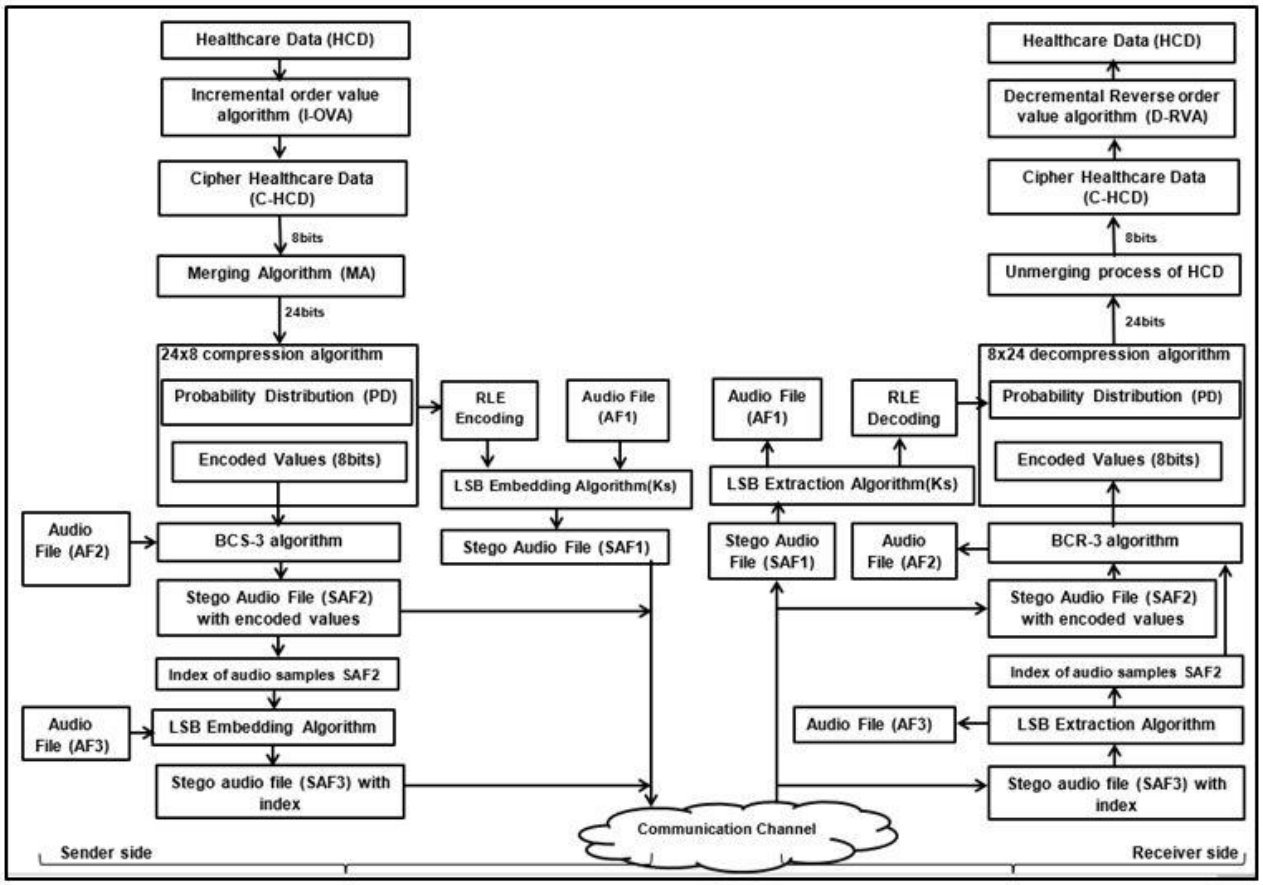


Figure 2: The framework of the proposed algorithm

3.1.6 Theoretical discussion of 24x8 compression algorithm

Figure 3 describes the step-by-step process of compressing M-C-HCD, as shown in Table 3.

1. A probability array of length 16,777,216 was computed, with all elements initialized to zero.
2. The decimal value of the binary "000101110001110101000101" is 1514821. Similarly, the decimal value of "010100100001110000010001"is 5381137.
3. An index array is created by incrementing each decimal value obtained in Step 2 by 1, resulting in indices 1514822 and 5381138.
4. The probabilities are P[1514822]=P[5381138]=1, indicating a single occurrence for each index in the array.
5. The total probability is P[1514822]+P[5381138]=2.
6. Finally, the probabilities are calculated as PD[index]=P[index]/Total probability, resulting in P[1514822] = ½ = 0.5.
   Similarly, P[5381138] = ½ = 0.5.
7. The encoded value is calculated by:
   Encoded_Value[index]=sum(P[index-1]+P[index]/2).
   For index 1514822,
   Encoded_Value[1514822]=0+0.5/2=0.25,
   Similarly, for the index 5381138,
   Encoded_Value[5381138]=0.5+0.5/2=0.75.
8. Scaling the encoded value to 8-bit binary by multiplying the Encoded_Value by 255.
   For index 1514822, Scaled_Encoded_Value [1514822] = 0.25x255=63.75.
9. Similarly, for the index 5381138, Scaled_Encoded_Value [5381138]=0.75x255=191.25.
10. Rounding the Scaled_Encoded_Value to the nearest integer, resulting in 64 for index 1514822 and 191 for index 5381138.
11. Convert the rounded encoded value to binary: 64 = 01000000 and 191 = 10111111.

3.1.7 Phase IV: bit comparison and substitution-3 algorithm (BCS-3)

Input: Audio File (AF2), and encoded_binary_array
Output: Stego Audio File (SAF2)

1. Read the AF2 (.wav) and encoded_binary_array of 8 bits.
2. Convert AF2 to a 24-bit binary stream.
3. Hide an approved code between the sender and receiver to reveal that the SAF2 contains hidden encoded_binary_array data.
4. Compare the AF2 and encoded_binary_array.
5. For i = 1 to N do // Repeat For loop
   until encoded_binary_array
   are hidden.
6. If sum ((AF2) ~= encoded_binary_array) = 3

Merged cipher healthcare (M-C-HCD) data																								Decimal Value	Incremented Decimal Value	Probability	Total Probability	Normalization Probability	Cumulative Probability	Encoded Value	Scaled_Encoded _Value	Rounding_ Scaled_Encoded _Value	Encoded_ binary _array
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24										
0	0	0	1	0	1	1	1	0	0	0	1	1	1	0	1	0	1	0	0	0	1	0	1	1514821	1514822	P[1514822]=1	2	P[1514822]=0.5	0	0.25	0.25*255=63.75.	64	01000000
0	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	1	5381137	5381138	P[5381138]=1		P[5381138]=0.5	0.5	0.75	0.75*255=191.25.	191	10111111

Figure 3: Theoretical discussion of the 24x8 compression algorithm

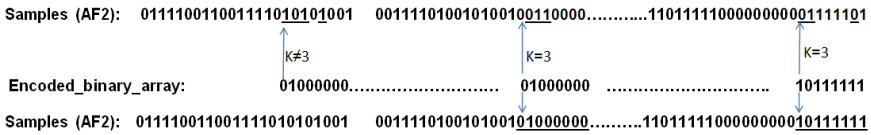


Figure 4: Embedding of encode value 0.25 and 0.75 using the BCS-3 algorithm

7. Embed encoded_binary_array in selected samples of AF2.

   // Embed the Encoded_binary_array in the AF2 selected samples, which differ by exactly 3 bits.
8. End If
9. End For
10. The stego Audio File (SAF2 with hidden encoded_binary_array) is computed.
11. End

The proposed BCS-3 algorithm extends the embedding method [25] through compressed inputs and K. Due to the reduced size, more audio samples are available for comparison and embedding, allowing for improved hiding of encoded values with greater flexibility in sample selection. These modifications in embedding are reflected in the retrieval process by the Bit Comparison and Retrieval-3 (BCR-3) algorithm, as an extension of the retrieval algorithm in [25]. The BCS-3 algorithm compares eight consecutive LSB bits from AF2 with a one-byte Encoded_binary_array. BCS-3 calculates the bit differences (K). If K=3, the algorithm computes and outputs an audio sample from the audio file 2 (AF2). Within this audio sample, eight consecutive bits from the LSBs of AF2 are embedded with a one-byte Encoded_binary_array data.

This process repeats for embedding the Encoded_binary_array data within the audio file 2 (AF2), producing a Stego Audio File 2 (SAF2) with the hidden Encoded_binary_array.

Figure 4 illustrates embedding the encoded_binary_array value obtained from the theoretical discussion of the 24x8 compression algorithm using the BCS-3 algorithm. The first sample is skipped because it has 4-bit differences ($K \neq 3$). The comparison then continues until an AF2 sample with exactly 3-bit differences from the compressed encoded_binary_array is found. In that selected sample, the data 01000000 is embedded. The process then continues with subsequent samples, where the next data value 10111111 is embedded.

The performance of the BCS-3 algorithm is validated using metrics such as Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR), as explained in the Results and Discussion section. Additionally, the algorithm undergoes testing for steganalysis, as claimed by the results of histogram error rate (HER) and the fourth first moments.

3.1.8 Phase V: embedding probabilities in AF1 and indices of SAF2 samples in AF3

Run-length encoding compressed the probabilities with a sequence length of 2^24 for each segment.

It compresses data by representing sequences of repeated values into a single value and a count. The probability sequence for each 501-byte segment typically consists of 16777049 zeros and 167 ones, where the ones correspond to HCD occurrences and are compressed using run-length encoding (RLE). Applying RLE to a 501-byte segment reduces the sequence length from 16777216 to 331 counts, along with their corresponding values, depending on the HCD distribution. The RLE algorithm is applied to all 30 segments within the 15000-byte M-C-HCD, and the resulting RLE counts and values are stored in the probability array.

Decoding the encoded values depends on the probabilities of HCD occurrences at the receiver side. Probability counts and their values are represented in binary using 24 bits for precise retrieval. The traditional least significant bit algorithm embeds probability counts and values using the reference Ks [25] in the least significant bit of audio samples.

An uncompressed digital audio wave file (AF1) with a sample size of three bytes is the input for the LSB algorithm. A private key, known as a secret key (Ks), determines the starting audio sample index for executing the LSB algorithm in the proposed method. Such keys should be shared only with individuals authorized to decode the information. This secret key is selected and mutually agreed upon by both the sender and the receiver. The proposed LSB algorithm utilizes Ks=60 as the private key. For each audio-steganographic communication, the secret key is assigned a new starting embedding sample to enhance security and prevent unauthorized access.

The RLE counts and values with the total in the probability array undergo a conversion from decimal to binary notation of 24 bits. Following this conversion, the resultant binary values of total, counts, and values are sequentially integrated into the AF1 using the LSB algorithm with Ks as reference. Therefore, pinpointing the exact audio sample in the AF1 where the LSB algorithm initiates embedding the probability counts and their values with total poses a challenge.

Figure 3 shows that the probability sequence contains occurrences of M-C-HCD at indices 1514822 and 5381138, while all other indices hold zero values. Applying the RLE algorithm to this input segment

compresses the probability sequence into five counts with their corresponding values, as presented in Table 4.

Each audio sample includes an index value, enabling efficient and direct access. The index of the Stego Audio File 2 (SAF2) containing the embedded Encode_binary_value is documented.

The index value is expressed as a decimal number and converted into a binary representation. The above-indexed values are incorporated into the cover audio file 3 (AF3) through the Least Significant Bit (LSB) algorithm with Ks as a reference. This method extends the approach in [25] by embedding compressed HCD into audio samples to reduce data size, which consequently decreases the number of required index values, with only 5000 indices needed for 15000B of data. It also uses fixed 24-bit index lengths and modified inputs, including Ks and audio samples with encoded values. AF3 is an uncompressed digital audio wave file with a sample size of three bytes. The LSB algorithm embeds the index value in the least significant bit of the audio samples. For embedding 15000B of HCD, the total count of index values is 5000 of 24bits. Pinpointing the exact audio sample in AF3, where the LSB algorithm begins embedding the index value from SAF2, is challenging. Additionally, the size of the index value (24 bits) adds complexity by requiring the handling of larger data segments, making it significantly harder to compromise the LSB algorithm.

The stego audio file SAF1 with RLE counts and probability values, with total and SAF3 with the index values of audio samples SAF2, are transmitted to the receiver.

The metrics PSNR and BER validate the LSB algorithm. The results and discussion section discuss the above metrics.

The private key (Ks) is the initial sample in audio files AF1 and AF3, which initiates the

execution for embedding RLE total, counts, values, and index values. The value of Ks varies for each communication to ensure isolation and security.

The enhanced LSB algorithm uses Ks = 60, where the 60^{th} sample in AF1 and AF3 is used as the private key, mutually agreed upon by the sender and receiver for embedding. This random selection significantly enhances security by expanding the key space, represented as 2^N, where N is the total number of samples, nearly 5300000. Table 6 provides the details of the audio files used in this work. The large number of audio samples and the expanded key space make it computationally difficult for a steganalyst to extract the private key, thereby ensuring robustness in embedding the RLE total, counts, values, and index values.

3.2 Extraction algorithm

3.2.1 Phase I: Extraction of probabilities

Three uncompressed 24-bit stego audio files, SAF1, SAF2, and SAF3, are received. SAF1 contains hidden RLE total, count, and probability values. SAF2 contains encoded values in 8-bit, and SAF3 contains index values of stego audio samples (SAF2).

The initial step in the proposed extraction algorithm involves extracting the RLE total, count, and values of the probability distribution of HCD

from SAF1, with Ks as the reference. An LSB extraction algorithm reads the least significant bit of the stego audio file samples (SAF1) to retrieve the RLE total, counts, and probability values.

The LSB retrieval starts from sample index 60. The first 24 bits represent the total value converted to decimal. This total value indicates the number of 24-bit segments for both the RLE counts and values. The next 24-bit segment represents the first set of counts. The process then repeats, with the next 24 bits representing the next total, determining the number of subsequent 24-bit segments for the next set of counts. This continues until all the count data is retrieved. After all count values are retrieved, the corresponding values are extracted based on the same total values. The extracted counts and values are stored separately and used to reconstruct the probability distribution for further processing. Expanding the RLE count and value generates the probability distribution, which is then input to the 8x24 decompression algorithm.

Table 4: Run Length encoding (RLE) algorithm

Probability zero occurrence	**Probability one occurrence**	**Run Length Encoding Algorithm (RLE)**					
***P[1-1514821]=0** ***P[1514823-5381137]=0** ***P[5381139-16777216]=0**	***P[1514822]=1** ***P[5381138]=1**	**Counts**	**1514821**	**1**	**3866315**	**1**	**11396078**
		Values	**0**	**0.5000**	**0**	**0.5000**	**0**

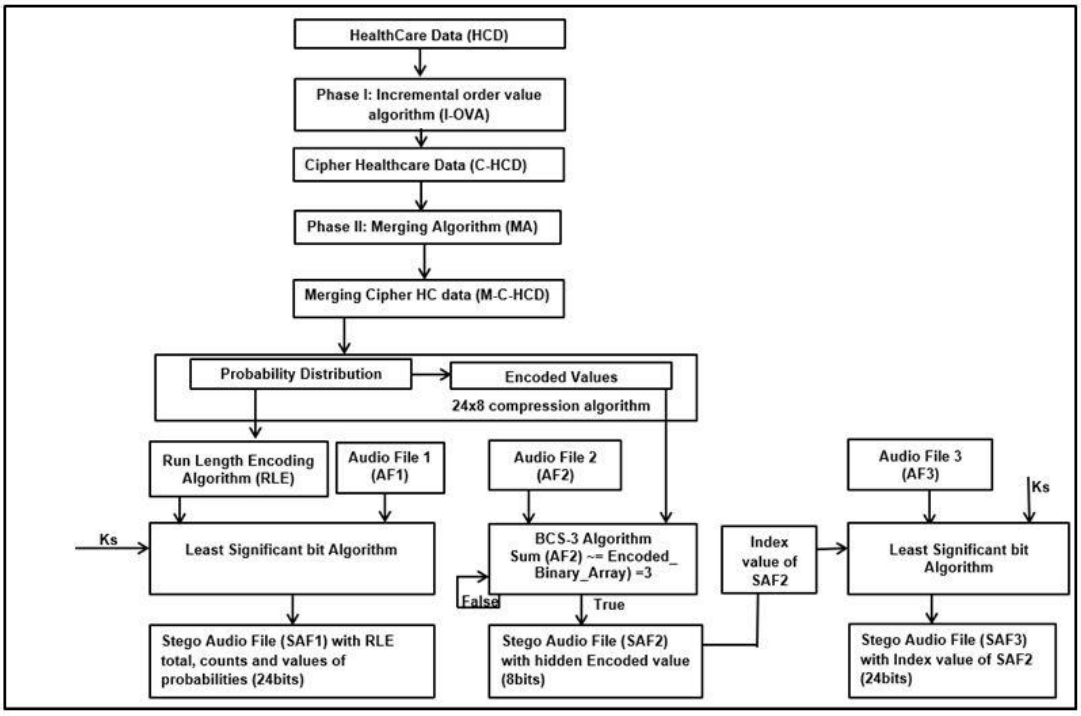


Figure 5: Block diagram for proposed embedding algorithm

3.2.2 Phase II: bit comparison and retrieval-3 algorithm (BCR-3)

The received stego audio file SAF3 was converted into 24-bit binary streams. An LSB extraction algorithm is applied to retrieve the index values, which correspond to the index numbers of audio samples in SAF2 that contain encoded_ binary_values with a 3-bit difference condition. Based on a mutually agreed code between the sender and receiver, the algorithm reads the least significant bit of audio samples in SAF3, starting from sample index 60. Each 24-bit segment corresponds to one index value, converted to equivalent decimal values. This decimal value is the retrieved index value, and the process continues for all retrieved bits. For 15000B of HCD, 5000 index values are retrieved. The retrieved index values are then input to the Bit comparison and retrieval-3 (BCR-3) algorithm. The received stego audio file SAF2 is converted to 24bit binary streams. The encoded_binary_values are extracted using the retrieved index numbers as references. Specifically, the extraction involves retrieving eight consecutive bits from the LSB of the specified index values samples in SAF2. The complete encoded_binary_values are retrieved by repeating the above process. The retrieved encoded values are input into the 8x24 decompression algorithm. The extraction algorithms for retrieving index values are validated using the results of Bit Error Rate (BER) and Normalized Cross-Correlation (NCC), as detailed in the Results and Discussion section.

3.2.3 Phase III: 8x24 decompression algorithm

Input: Probability distribution and Encoded values (8 bits)
Output: Merged Cipher healthcare (M-C-HCD) data (24 bits)

Step 1: Read Inputs
Read the normalized probability distribution: probabilities.
Read the array of encoded values: encoded_binary_array.
Step 2: For each encoded_value in encoded_binary_array:
Convert the 8-bit encoded binary to decimal.
encoded_value ← bin2dec(encoded_binary) / 255
Initialize cumulative_sum ← 0
Step 3: Locate the Index
For J = 1 to N1 // N1 = length of probabilities
cumulative_sum ← cumulative_sum + probabilities[J]
If encoded_value < cumulative_sum: Break
Step 4: Convert Index to Binary
decoded_index ← J - 1
Convert decoded_index to 24-bit binary:
M-C-HCD_segment←dec2bin(decoded_index, 24)
Step 5: Store the Recovered Segment
Append M-C-HCD_segment to M-C-HCD_array
Step 6: Output
Return M-C-HCD_array as the reconstructed M-C-HCD.

3.2.4 Phase IV: decremental reverse order value algorithm (D-RVA)

Input: Merged cipher Healthcare data (M-C-HCD in 24 bits)
Output: Healthcare data (HCD)

1. Read the M-C-HCD of each segment of 15000B.
2. Split the 24 bits of merged cipher healthcare data (M-C-HCD) into 8 bits of cipher HCD.
3. If the length of C-HCD is not divisible by 3, remove the last row or the last two rows if they contain exactly 1 byte of zero binaries added during the merging algorithm.
4. Convert each 8-bit binary cipher HCD segment to its corresponding decimal value. Each decimal value

represents the index of printable characters in the ASCII table.
5. Using the decimal values as indices, fetch the corresponding characters from the ASCII table.
6. Retrieve the ASCII value that corresponds to the fetched cipher HCD.
7. Decrement the ASCII value based on the position of C-HCD.
8. Fetch the cipher character corresponding to the decremented ASCII value.
9. Retrieve the assigned order value for the fetched character.
10. Use the order value as an ASCII value and fetch the corresponding character.
11. The characters fetched in step 10 result in the decryption of the cipher healthcare data.
12. Repeat the above steps for each segment within the 15000B data.
13. Combine each decrypted segment to form the complete decrypted HCD.
14. End

3.3 Computational complexity

To efficiently handle the large probability distribution (PD) of size 2^{24}, the proposed compression–decompression framework processes input in fixed 501-character segments, limiting active PD storage to O(u), where u is the number of unique entries in a segment. During compression, each PD is RLE-compressed, reducing storage from O(2^{24}) to a few hundred entries O(u). Each segment uses a dedicated PD in memory, and memory usage remains low without needing explicit variable clearance. In decompression, RLE-decoded PDs are stored in a cell array, retrieved per segment, used, and cleared, maintaining constant memory usage. The system runs without memory overflow on a 4 GB RAM machine, confirming feasibility. Computational complexity for both compression and decompression is O(1) within the tested input range, as shown in Figure 6. The results were based on 75 simulations supported by very low R^2 values (0.045 for compression, 0.047 for decompression), indicating minimal dependency on input size and ensuring scalability for larger datasets.

The execution time was evaluated through 75 simulations using five audio files and five dataset sizes (3000–15000 bytes), each combination executed three times for consistency. The computational complexity of both the single and triple-audio-file approaches is O(n), with execution time increasing proportionally to the HCD size. The average execution time of the proposed algorithm is 56.1236 s for the three-audio-file approach and 19.65634 s for the single-audio-file implementation. Despite the lower execution time for the single-audio-file approach, the three-audio-file approach enables more efficient and modular management of structured healthcare data. By isolating the embedding of healthcare data (HCD), index values, and RLE counts/values into separate audio streams, the system reduces coupling and supports incremental updates, where only modified components require re-encoding instead of reprocessing all three files. This modular design also allows independent validation of each HCD, minimizing the risk of systemic corruption. Furthermore, the three audio files provide scalability for larger datasets and establish a foundation for future parallelized I/O and processing pipelines. Consequently, despite the additional runtime, the three-audio-file approach provides greater technical robustness and long-term operational efficiency.

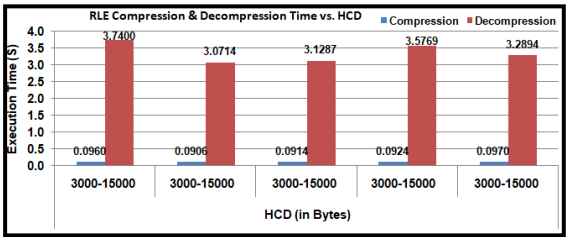


Figure 6: Execution time for RLE Compression and Decompression

4 Results and discussion

The proposed embedding and extraction algorithm was implemented and evaluated using MATLAB software on over fifteen audio files with embedding probability distribution, healthcare data, and index values of audio samples. Uncompressed audio files from the Mixkit dataset [34], with a consistent 44,100 Hz sampling rate, ensure uniform audio quality and minimize variability during embedding or extraction processes.

The selected audio files have a one-minute duration, enabling fast processing and facilitating repeated testing within a reduced time frame.

The dataset's genre diversity offers a variety of audio contexts, allowing a comprehensive assessment of the algorithm's adaptability across different scenarios.

Open availability of the Mixkit dataset ensures the reproducibility of experiments.

Utilizing unprocessed audio files minimizes computational overhead and accurately reflects performance conditions for the steganography algorithms.

Table 6 lists the specifications of the audio files used in the algorithms [35].

The HCD consists of the COVID Dialogue Dataset, which includes conversations between patients and doctors about COVID-19 and other types of pneumonia, sourced from websites such as icliniq.com, healthcaremagic.com, and healthtap.com [36].

4.1 Throughput

Throughput evaluates the performance of encryption and decryption algorithms by measuring the data transfer rate using Equation (1) [37].

$$\text{Throughput} = \frac{\text{Total Size of Healthcare data(B)}}{\text{Total Execution Time}} \quad (1)$$

The data transfer rate for 15000B using the proposed I-OVA and D-RVA algorithms is 8592.74 KB/s, which signifies that 15000B of data was transferred in 0.0017s. The above indicates that the time taken for each step of the

proposed algorithm is minimal, resulting in high throughput.

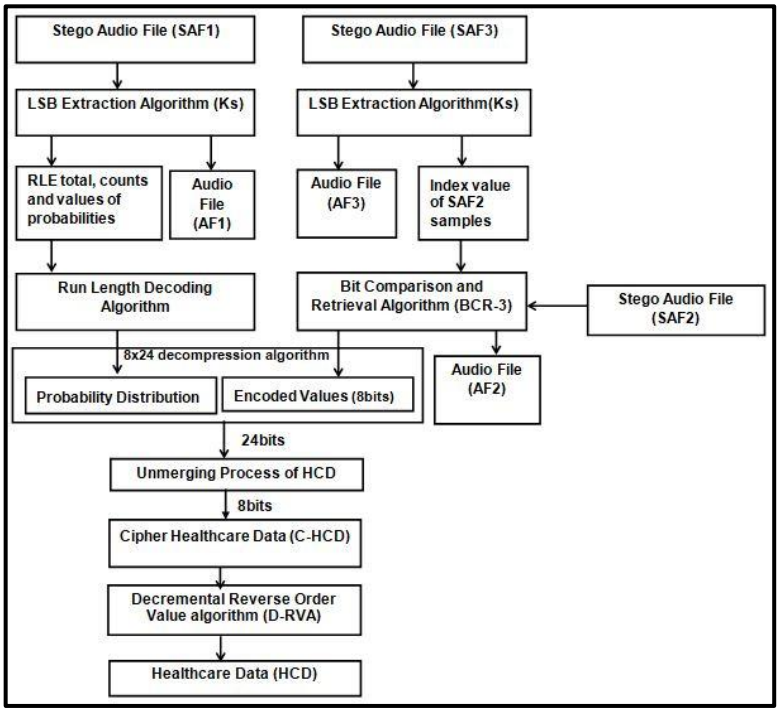


Figure 7: Block diagram for proposed extraction algorithm

Probability	Binary_ encoded _Value	Decimal _ encoded_ Value	Encoded_Value	Cumulative Probability (J)	J Decimal Value	J-1 Decimal Value	Merged cipher healthcare data (M-C-HCD)																									C-HCD							
							1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	6	7	8
																															0	0	0	1	0	1	1	1
P[1514822]=1	00111111	63	63/255=0.247088	0.5	1514822	1514821	0	0	0	1	0	1	1	1	0	0	0	1	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	1	0	1
																															0	1	0	0	0	1	0	1
																															0	1	0	1	0	0	1	0
P[5381138]=1	10111111	191	191/255=0.74901	1	5381138	5381137	0	1	0	1	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	1	0	0
																															0	0	0	1	0	0	0	1

Figure 8: Theoretical discussion of the proposed extraction algorithm

Table 5: Decremental reverse order value algorithm

Binary Value_Index I-OVA								Index I-OVA	C-HCD	ASCII Code	D-RVA value	ASCII Code	ASCII character	OVA value	ASCII Code	HC data
1	2	3	4	5	6	7	8									
0	0	0	1	0	1	1	1	23	7	55	55-1=54	54	6	105	105	i
0	0	0	1	1	1	0	1	29	=	61	61-2=59	59	;	100	100	d
0	1	0	0	0	1	0	1	69	e	101	101-3=98	98	b	61	61	=
0	1	0	1	0	0	1	0	82	r	114	114-4=110	110	n	49	49	1
0	0	0	1	1	1	0	0	28	<	60	60-5=55	55	7	104	104	h
0	0	0	1	0	0	0	1	17	1	49	49-6=43	43	+	116	116	t

Table 6: Description of audio files [35]

Specifications	
Bit per sample	24
Sample Rate	44100
Channel	2
Audio Type	Music
Duration in Minutes	1

Recent research used 3-DES for securing healthcare data [38], [39].To assess efficiency, the proposed algorithm is compared with 3-DES. Figure 9 compares the throughput values achieved by the proposed algorithms, I-OVA and D-RVA, relative to those of the 3-DES algorithms.

The x-axis denotes the size of healthcare data in bytes, and the y-axis represents throughput in kilobytes per second. The improved throughput is due to the algorithm's simple structure, which avoids complex key scheduling and multiple encryption rounds, reducing overall computation time.

The results prove that the algorithm is scalable, transferring large volumes of HCD without significant degradation. Hence, the proposed algorithm is suited for telemedicine consultations involving healthcare data, ensuring efficient and secure communication.

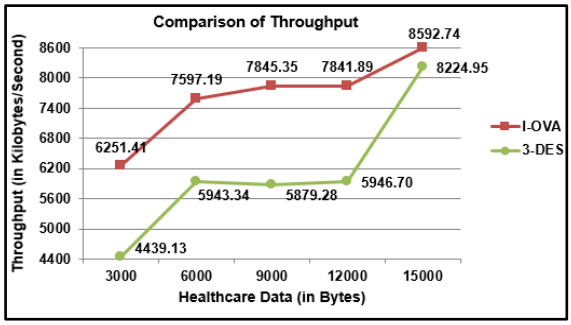


Figure 9: Compares the throughput achieved by the proposed techniques with other encryption algorithms

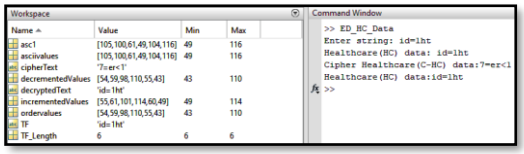
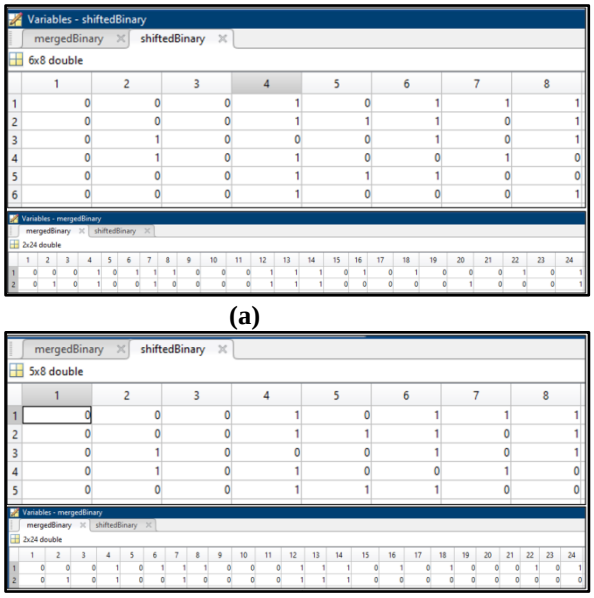


Figure 10: Displays the results of the incremental order value algorithm (I-OVA) and the decremental reverse order value algorithm (D-RVA)



(a)

(b)

Figure 11: Results of the merging algorithm for C-HCD with length (a) divisible by 3 and (b) not divisible by 3

Figure 10 presents the simulation results for the proposed I-OVA and D-RVA algorithms for the HCD listed in Table 2.

Figure 11 displays the outcomes of the merging algorithm. It verifies that when the length of the C-HCD is not divisible by 3, the algorithm appends one or two bytes of zero binary bits at the end of the C-HCD. In Table 2, the C-HCD "23,29,69,82,28,17" has a length of 6, divisible by 3, resulting in the M-C-HCD as "23,29,69" and "82,28,17", which does not require the addition of zero bits. Conversely, when C-HCD is "23, 29, 69, 82, 28" with a length of 5, which is not divisible by 3, the algorithm appends one byte of zero bits. This results in the M-C-HCD as "23, 29, 69" and "82, 28, **00000000**".The decimal values of the C-HCD are first converted to their binary equivalents before merging.

4.2 Entropy

The statistical security of cipher HCD in randomness is measured with entropy given by H(s) in Equation (2) [37].

$$H(s) = \sum_{i=0}^{n-1} p(s_i) \log_2 \frac{1}{p(s_i)} \quad (2)$$

The probability symbol S_i is represented as $P(S_i)$

Figure 12 shows the entropy comparison with the 3-DES algorithm, with the x-axis representing HCD in bytes (B) and the y-axis representing entropy. The entropy of the proposed algorithms is 6.5, which results in strong randomness and resistance to frequency analysis and pattern recognition attacks of the ciphertext. The reason for randomness is the positional increment of the ASCII value in I-OVA. The comparison graph shows that the proposed encryption algorithm is robust, achieving an entropy of 6.5, approximately the same as the standard 3-DES value of 7. This suggests that the algorithm provides strong security with efficiency comparable to 3-DES. The proposed I-OVA ensures the first layer of protection for HCD in audio-steganographic communication.

4.3 Compression ratio (CR)

CR is the ratio of the total number of bits in uncompressed data to the total number of bits in compressed data, expressed in bit per bit (bpb) in Equation (3) [11].

$$CR = \frac{\text{No: of bits in uncompressed data}}{\text{No: of bits in compressed data}} \quad (3)$$

The average CR of the proposed 24x8 compression algorithm is 3:1, as shown in Figure 13, compared to values from existing compression algorithms. The proposed 24×8 compression is more efficient than existing algorithms by eliminating statistical and dictionary dependencies, ensuring faster and consistent compression with minimal complexity. The x-axis denotes the size of healthcare data in bytes (B) (3000B, 6000B, 9000B, 12000B, and 15000B), while the y-axis denotes the compression ratio in bpb. This compression ratio indicates that the healthcare data is compressed to one-third of its original size, reducing 24 bits of M-C-HCD to 8 bits. This results in a significant reduction of the healthcare data from 15000B to 5000B.

Space saving is defined as the reduction in file size relative to the uncompressed size, as given in Equation (4) [11]

$$\text{Space savings} = 1 - \frac{\text{No.of bits in compressed data}}{\text{No.of bits in uncompressed data}} \quad (4)$$

The proposed 24x8 compression algorithm achieves a space savings of 66.67%. The above indicates that the compressed data occupies about one-third of the original storage space (33.33%), which results in significant storage reduction.

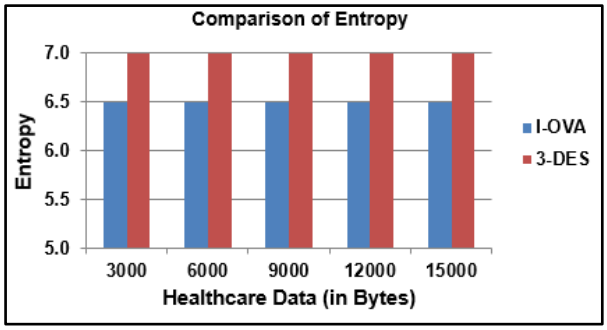


Figure 12: Compares the entropy achieved by the proposed techniques with that of the 3-DES algorithm

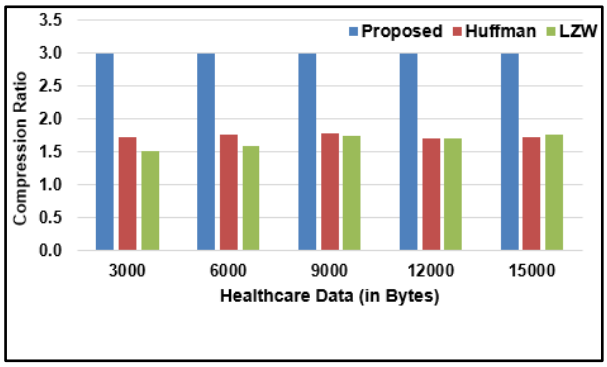


Figure 13: Comparison of different compression schemes in terms of Compression Ratio

Indirectly, efficient compression reduces transmission costs by enabling faster transfer of highly compressed data. Therefore, the 24x8 compression and 8x24 decompression algorithms are efficient. Hence, the simulation results demonstrate that these 24x8 compression and 8x24 decompression algorithms communicate the HCD faster with enhanced capacity during telemedicine consultations.

To handle larger datasets efficiently, the 24-bit HCDs are compressed to 8-bit representations. The normalized values from the compression algorithm are multiplied by 255; the results are rounded and converted to 8-bit binary, as explained in the 24x8 compression algorithm. This reduces the memory required to store each value from 24 bits (3 bytes) to one-third of the original size, 8 bits (1 byte). The 24x8 compression algorithm significantly reduces the storage requirements and memory usage. Hence, HCD datasets larger than 15 KB can be effectively handled without affecting the computational integrity of the algorithm.

4.4 Similarity analysis

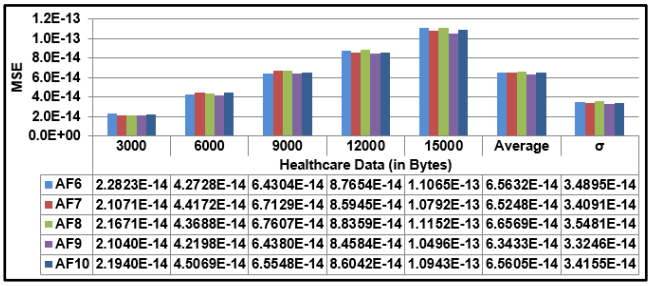
4.4.1 Mean squared error (MSE)

The MSE measures the distortions between the cover and stego audio files. An MSE value of approximately zero indicates minimal distortions, as given by Equation (5) [35].

$$\mathrm{MSE} = \frac{1}{Q}\sum_{i=1}^{Q}(\mathrm{CAF_i} - \mathrm{SAF_i})^2 \qquad (5)$$

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

The average MSE value of the proposed BCS-3 algorithm was 6.5297E-14 from 25 simulation runs. Figure 14 illustrates the relationship between different sizes of HCD (3000B, 6000B, 9000B, 12000B, and 15000B) on the x-axis and the MSE of various audio files (AF6, AF7, AF8, AF9, and AF10) on the y-axis. Table 7 compares the average MSE of the proposed and existing algorithms, highlighting significant differences. The proposed BCS-3 algorithm embeds an 8-bit encoded binary value into 24-bit audio samples, resulting in a consistent 3-bit distortion between the CAF and SAF for each 1-byte embedding, thereby outperforming the conventional LSB algorithm, which



	3000	6000	9000	12000	15000	Average	σ
AF6	2.2823E-14	4.2728E-14	6.4304E-14	8.7654E-14	1.1065E-13	6.5632E-14	3.4895E-14
AF7	2.1071E-14	4.4172E-14	6.7129E-14	8.5945E-14	1.0792E-13	6.5248E-14	3.4091E-14
AF8	2.1671E-14	4.3688E-14	6.7607E-14	8.8359E-14	1.1152E-13	6.6569E-14	3.5481E-14
AF9	2.1040E-14	4.2198E-14	6.4380E-14	8.4584E-14	1.0496E-13	6.3433E-14	3.3246E-14
AF10	2.1940E-14	4.5069E-14	6.5548E-14	8.6042E-14	1.0943E-13	6.5605E-14	3.4155E-14

Figure 14: The impact of MSE on different sizes of healthcare data and audio files

may alter between 0 to 8 least significant bits per byte, leading to non-deterministic and potentially higher distortion. This difference progressively increases during the embedding process, reaching 15000 bits out of 40000 bits for an HCD of 15000B. Specifically, 5000 encoded values for 15000B of HCD result in 40000 (5000x8) embedding bits. For each 8-bit embedding, 3 bits will be error bits, reaching 15000 error bits. The above ensures that only 3 bits in a 24-bit audio sample are distorted, resulting in a reduced MSE of 6.5297E-14, close to the benchmark value.

The standard deviation of MSE values ranges from 3.3246E-14 to 3.5481E-14 across all HCD and audio types, as shown in Figure 14. This results from the algorithm's consistent data embedding across all audio files, regardless of their specific properties, resulting in stable performance.

The uniform embedding process introduces minimal distortion, further supported by the consistent sampling rate of the audio files.

Statistical analysis yields a p-value of 0.999931 ($p > 0.05$), accepting the null hypothesis and indicating no significant differences in MSE across audio files.

This result suggests that the MSE does not vary with the bit patterns in the audio samples and HCD. The algorithms introduce a consistent error bit pattern to demonstrate that the audio steganography method effectively controls distortion in HCD. Consequently, the audio quality remains largely unaffected during HCD embedding.

4.4.2 Peak signal to noise ratio (PSNR)

In Equation (6) [35], the PSNR expressed in decibels is a logarithmic measure of the stego signal quality.

$$\text{PSNR} = 10\log_{10}\frac{(2^{q}-1)^2}{\text{MSE}} \qquad (6)$$

Where q is the maximum number of bits in the cover audio sample.

Higher PSNR results in enhanced quality of the stego audio file. Figure 15 illustrates the relationship between different sizes of HCD (3000B, 6000B, 9000B, 12000B, and 15000B) on the x-axis and the PSNR of various audio files (AF6, AF7, AF8, AF9, and AF10) on the y-axis. Table 7 compares the average PSNR of the proposed algorithm with the existing algorithms. The average PSNR of the proposed BCS-3 algorithm is 42.1480 dB. This is due to the average MSE of 6.5297E-14, which is inversely related to PSNR. Hence, the proposed BCS-3 algorithm improves upon existing methods by using 8-bit embedding to achieve higher payload capacity. It embeds 15,000 B of data without significantly altering the stego audio file, resulting in 15,000 error bits that enhance resistance to steganalysis while maintaining high audio quality. This higher PSNR is due to the algorithm's design, where each 8-bit embedded value differs by only 3 bits from the original, minimizing distortion compared to traditional LSB-based methods.

Similarly, PSNR is measured for the stego audio files (SAF1, SAF2, SAF3, SAF4, and SAF5) with hidden RLE total, counts, and values, is 30.1417dB. PSNR for the stego audio files (SAF11, SAF12, SAF13, SAF14, and SAF15) with hidden index values of stego audio samples (SAF6, SAF7, SAF8, SAF9, and SAF10) containing the encoded value is 36.5499 dB. Both PSNRs indicate no degradation in the quality of stego audio files.

The 15000-bit differences resulting from embedding do not impact audio intelligibility, as evidenced by an average PSNR of 42.1480 dB and a low standard deviation of 2.7 across various audio files. A PSNR above 36 dB is typically considered imperceptible to the human ear [40]. The consistent PSNR and MSE values across different HCD sizes confirm the stability of the BCS-3 algorithm.

Figure 16 graphically depicts the imperceptible difference between the cover audio file and the stego audio file for various genres of audio files ranging from AF6 to AF10, each embedded with 15000B of healthcare data. These results confirm that the embedding is both imperceptible and resistant to detection.

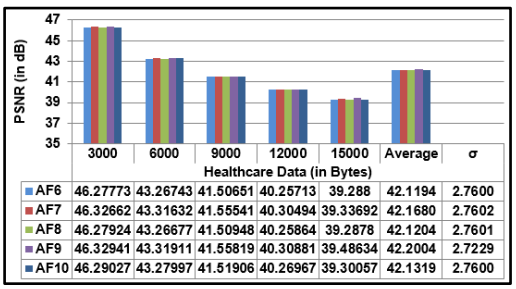


Figure 15: The impact of PSNR on different sizes of healthcare data and audio files

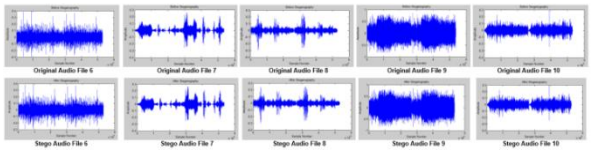


Figure 16: Various cover audios and stego audios with 15000B of healthcare data

4.5 Hiding capacity (Payload)

The hiding capacity represents the maximum size of healthcare data embedded in an audio file, calculated using Equation (7) and expressed as a percentage [35].

The hiding capacity in the LSB algorithm is given by Equation (8) in % [35]

$$\text{Hiding Capacity (HC)} = \left(\frac{\text{Text File size}}{\text{Audio File size}}\right) * 100 \qquad (7)$$

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

Table 7: The MSE and PSNR Comparison between the Proposed and Existing Algorithms

Method	Text Files (Bytes)	MSE	PSNR (dB)
[26]	107	9.08875 ▲100%	34.2775 ▲22.96%
[27]	103	-	17.71916498 ▲137.87%
[28] LSB	250	0.3994 ▲100%	-
[28] Pattern Matching	250	0.5619 ▲100%	-
[29]	9405	0.8578 ▲100%	-
Proposed	**15000**	**6.5297E-14**	**42.1480**

The simulated results for embedding 15000B (5000 encoded values) of HCD into an audio file AF7 with 5365030 samples require 5000 samples for the proposed BCS-3 algorithm, as summarized in Table 8. Each encoded value represents 3 bytes of HCD. In comparison, the LSB algorithm requires 120000 samples, indicating a reduction of 115000 samples with the proposed method. For 15000B of HCD, the number of audio samples needed by the BCS-3 algorithm is 5000, based on its 3:1 embedding ratio. Each encoded value requires 1 audio sample. In contrast, the LSB algorithm requires 15000B×8=120000samples, reflecting its 1:8 embedding ratio.

The proposed BCS-3 algorithm ensures that the number of samples used for embedding is proportional to one-third of the length of HCD in bytes, based on its 3:1 embedding ratio.BCS-3 embeds 1 byte per sample, whereas LSB embeds 1 byte per 8 samples. This ensures the BCS-3 algorithm optimizes sample utilization by reducing the number of samples required for embedding based on the embedding ratio. Table 8 compares the number of audio samples used by the LSB method and the proposed BCS-3 algorithm for embedding healthcare data of varying sizes.

4.5.1 Embedding rate

The embedding rate is the number of bits embedded in a one-byte audio file sample. The LSB algorithm embeds 1 byte of the HCD in 8 audio samples of 1 byte. This results in an embedding rate of 12.5%. However, the BCS-3 algorithm embeds 1 byte of encoded value into 1 audio sample (24 bits) with a 33.33% embedding rate. For the tested audio AF8 with samples 5306818, the proposed algorithm achieves a 20.83% increase in embedding rate compared to the LSB algorithm. The proposed algorithm requires only 5000 audio samples to embed 5000 bytes of encoded values. Since each audio sample is 3 bytes, the 5000 samples correspond to 15,000 bytes of audio, resulting in 33.33% embedding efficiency. Hence, the BCS-3 algorithm has significantly improved its embedding rate compared to the traditional Least Significant Bit algorithm.

Table 8: Audio samples utilized by the proposed BCS-3 algorithm for AF7 compared with the traditional LSB algorithm

Healthcare data (in Bytes)	Samples Utilized (in samples)		Difference Bit (DB) (in Bits)	
	LSB	Proposed Method	LSB	Proposed Method
3000	24000	1000	4033	3000
6000	48000	2000	7968	6000
9000	72000	3000	12160	9000
12000	96000	4000	16011	12000
15000	120000	5000	19876	15000

4.6 Strength analysis

4.6.1 Bit error rate (BER)

BER measures the correctness of embedding with the percentage of the embedded HCD bits that were retrieved incorrectly, as expressed by Equation 9.

$$\mathrm{BER} = \frac{\mathrm{HCD_{error}}}{\mathrm{HCD_{bits}}} \tag{9}$$

Where $\mathrm{HCD_{error}}$ is the retrieved error bits and $\mathrm{HCD_{bits}}$ is the total number of bits embedded. In the absence of attacks, the embedded HCD is retrieved 100% successfully using the proposed BCR-3 retrieval algorithm for all tested audios with varying HCD sizes [35].

It is also defined as the ratio of the difference in bits between cover and stego audio files with hidden healthcare data to the total number of bits in the audio file.

$$\mathrm{BER} = \frac{1}{Q}\sum_{i=1}^{Q} \begin{matrix} 1, \mathrm{CAF(i)} \neq \mathrm{SAF(i)} \\ 0, \mathrm{CAF(i)} = \mathrm{SAF(i)} \end{matrix} \tag{10}$$

Where CAF is the original audio, SAF is the stego audio, and Q is the total samples in the audio files. The lower BER indicates the quality stego audio file

The average BER of the proposed BCS-3 algorithm across the tested files (AF6–AF10) is 7.0165E-05, confirming the preservation of audio quality. Since the embedding process modifies the audio in a controlled manner, it ensures the reliable extraction of the embedded Healthcare data.

The above is evident in Figure 17, plotted between the length of HCD (Bytes) on the horizontal axis and BER on the vertical axis. The computed BER ranges from 2.3284E-05 for audio file AF9 with an HCD of 3000B to 1.1782E-04 for audio file AF8 with an HCD of 15000B, as compared to the BER reported by existing researchers [26, 30] in Table 9. The proposed BCS-3 algorithm outperforms the results of previous researchers, in which only one-eighth (3/24) of the original audio file is distorted during the embedding process. This accomplishment was achieved by embedding the healthcare data into the stego audio files (SAF6-SAF10), with alterations occurring in 3 out of 24 bits corresponding to the cover audio files (CAF6-CAF10). Specifically, the BCS-3 algorithm embeds three characters as one byte encoded value in 24-bit

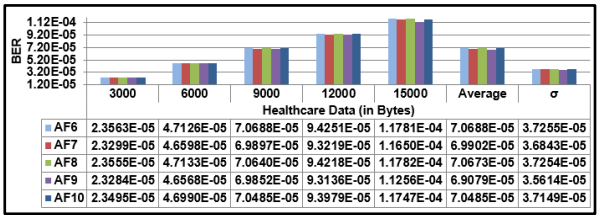


Figure 17: Influence of BER on various HCD sizes and audio files

audio samples with a bit difference of 3, reducing error bits during the embedding process. The BER resulted from altering 3 out of 24 bits while embedding 1 byte of HCD in 24-bit audio samples. Embedding 15000B of HCD generated 15000 error bits. The error bits produced during the embedding process are proportional to the length of the HCD in bytes at a 1:3 ratio, while for audio samples, the ratio is 3:1. This predictable bit-alteration pattern minimizes bit errors and ensures consistent BER performance. The BCS-3 algorithm optimizes the embedding process through effective error bit control and efficient audio sample utilization, ensuring predictable BER outcomes. In contrast, the LSB algorithm shows no predictable pattern of error bits, as its error bits vary depending on the bit pattern of the HCD and audio samples. The above leads to unpredictable bit alterations, higher BER variability, increased error rates, and reduced performance due to ineffective bit management.

The average BER values indicate that the algorithm's performance remains consistent regardless of the specific audio sample used. The standard deviation values, ranging from 3.561E-05 to 3.725E-05, reflect this consistency and are further supported by the p-value of 0.9999934($p>0.05$) in statistical analysis. Figure 17 presents these low standard deviations, indicating that the proposed algorithm performs consistently across different inputs.

The average BER of the stego audio files SAF1-SAF6 for retrieving the run-length encoding total count, individual count values, and corresponding data values is 0.001118, indicating successful decoding of the RLE values. The stego audio files SAF11-SAF15 successfully decoded the index values, achieving a BER of 0.000258.

4.6.2 Normalized cross-correlation(NCC)

NCC measures the degree of correlation between embedded and extracted Healthcare data and between cover audio and stego audio files. NCC given by Equation (11).

$$\mathrm{NCC}(\mathrm{HCD},\mathrm{HCD}') = \frac{\sum_{K=1}^{P}\mathrm{HCD}(K)\,\mathrm{HCD}'(K)}{\sqrt{\sum_{K=1}^{P}\mathrm{HCD}\,(K)^2}\sqrt{\sum_{K=1}^{P}\mathrm{HCD}'(K)^2}} \quad (11)$$

Where HCD and HCD' are embedded and extracted Healthcare data. P is the number of samples in the audio file [35]. The correlation values of NCC range from -1 to 1, where 1 indicates similarity, -1 indicates dissimilarity, and 0 indicates no correlation.

The NCC value between the embedded and extracted healthcare data is 1, indicating perfect correlation. This confirms the accurate and complete retrieval of the secret message in simulations of the proposed BCR-3 algorithm in the absence of attacks [35].

The NCC value of the proposed BCS-3 algorithm is 0.999999, indicating a high degree of correlation between the cover audio file and the stego audio file. Table 9 compares the NCC value obtained from the proposed BCS-3 algorithm with existing algorithms [31]. The NCC value of 0.999999 results from the predictable 3-bit differences introduced when embedding 15000 B of healthcare data into audio files (AF6-AF10), amounting to 40,000 bits (5,000 bytes × 8) after compression. Each encoded value (1 byte or 8 bits) in HCD is incorporated in the cover audio file with a 3-bit difference between the cover audio file sample and its corresponding 24-bit stego audio file sample on comparison. The bit difference generated for the 15000B embedded data is 15000 bits. This predictable bit-alteration pattern ensures a consistent error pattern in the cover audio file with minimal distortion after the embedded process, as proved in Figure 16.

4.7 Steganalysis

4.7.1 Histogram attack

The Histogram Error Rate (HER) measures the deviations between the cover and stego audio files while embedding healthcare data. The frequency distribution of cover and stego audio file samples is compared based on HER values ranging from 0 to 1. An HER of 0 indicates perfect similarity between the cover and stego audios.

In this study, 25 simulations were conducted using five different audio files labeled AF6 to AF10. These involved embedding healthcare data of various lengths, including 3000B, 6000B, 9000B, 12000B, and 15000B. The Histogram Error Rate (HER) of the proposed BCS-3 algorithm was assessed for each simulation using Equation (12), with the findings presented in Figure 18 [35].

$$\mathrm{HER} = \frac{\sum_{i=1}^{Q}(\mathrm{His_a} - \mathrm{His_s})^2}{\sum_{i=1}^{Q}(\mathrm{His_a}^2)} \quad (12)$$

Where $\mathrm{His_a}$ and $\mathrm{His_s}$ are the histograms of audio and stego audio files.

Table 9: The BER and NCC comparison between the proposed and existing algorithms

Method	BER	NCC
[26]	17.995% ▲99.96%	-
[30]	0.0187% ▲62.47%	-
[31]	-	0.99992 ▲0.008%
Proposed	**7.0165E-05**	0.999999

Table 11 compares the HER values between the proposed BCS-3 algorithm and existing algorithms. The average HER obtained for the proposed algorithm is 0, indicating a high resistance to histogram attacks.

Additionally, Figure 18 displays the histograms of five tested cover audio files (AF6 to AF10) with embedded 15000B healthcare data. The histograms before and after steganography are similar, indicating that the magnitude distribution of the stego audio file closely resembles that of the cover audio file.

4.7.2 Fourth first moments

The statistical distribution of the amplitude of cover and stego audios with embedded healthcare data is measured by the fourth first moments: average (μ), variance (σ^2), skewness (sk), and kurtosis (k). The Equations (13),(14),(15),(16), and (17) find the fourth first moments and the Difference Ratio (DR), respectively [32].

$$\mu = \frac{\sum_{i=1}^{Q} S_i}{Q} \quad (13)$$

$$\sigma^2 = \frac{\sum_{i=1}^{Q}(S_i - \mu)^2}{(Q-1)} \quad (14)$$

$$\mathrm{sk} = \frac{\sum_{i=1}^{Q}(S_i - \mu)^3}{(Q-1)\sigma^3} \quad (15)$$

$$\mathrm{k} = \frac{\sum_{i=1}^{Q}(S_i - \mu)^4}{(Q-1)\sigma^4} \quad (16)$$

$$\text{Difference Ratio(DR)} = \left|\frac{\mathrm{fm_a} - \mathrm{fm_s}}{\mathrm{fm_a}}\right| \times 100 \quad (17)$$

Where fma, and fms are any fourth first moments of an audio file and a 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.

The DR ranges from 0 to 1, with 0 indicating similar statistical distributions between cover and stego audio and 1 indicating significant differences. Table 11 presents the simulation results of the DR for the fourth first moments values in comparison with existing algorithms. These results were obtained by testing five different audio files (AF6-AF10) with 15000bytes of healthcare data embedded into them. The simulation results of DR are approximately zero, indicating resistance to statistical attack.

The average skewness is 6.7953E-07, showing no significant outliers in the amplitude of stego audio files with hidden healthcare data. Similarly, the average kurtosis of 3.5282E-09 indicates a stable amplitude distribution of audio samples with light tails, even after embedding 15000B of healthcare data. The average

variance value of 2.5759E-09 confirms that the trade-off between the embedding capacity of 15000B healthcare data and the integrity of cover audio files is maintained. Lastly, the mean of 1.8331E-04 indicates minimal perceptible change in the listening experience of stego audio files with 15000B hidden healthcare data during audio steganographic communication.

4.7.3 Re-sampling and AWGN attack:

Re-sampling involves down-sampling the stego audio from 44.1 kHz to 22.58 KHz and subsequently reconstructing it back to 44.1 KHz before transmission. This process alters sample values and can distort the embedded HCD. To evaluate resampling attacks, 25 simulation tests were conducted on five audio files. The stego audios were subjected to the resampling process, and performance was analyzed. The results demonstrate that the BCS-3 algorithm preserved audio quality under this attack. Specifically, the stego audios achieved a PSNR of 54.5 dB, a very low MSE of 6.0×10^{-6}, and a high NCC value of 0.987 between the original and resampled audio files. These values confirm that the distortion introduced by re-sampling was negligible.

White Gaussian noise was added to the stego audios at SNR levels of 0.001dB, 5dB, and 25dB. The algorithm was tested with 25 simulations on 5 audio files with HCD data, 3000 to 15000bytes. The attack resulted in HER of 2.06×10^{-1}, 7.34×10^{-2}, and 1.14×10^{-4} for 0.001 dB, 5 dB, and 25 dB, respectively. Correspondingly, the NCC values were 0.707, 0.872, and 0.999. Similarly, the stego audio with hidden index value has average HER of 0.394944, 0.235005, and 0.024533, with corresponding NCC values as 0.541275, 0.752979, and 0.970612. The stego audio with hidden RLE count and values has average HER and NCC values of 0.3937, 0.213304, 0.006237, 0.554481, 0.996419, and 0.763709, respectively.

These results demonstrate that the proposed BCS-3 algorithm and the enhanced LSB algorithm preserve the quality of stego audio files, with audio quality decreasing slightly as noise increases. The stego audio files are transmitted sequentially, and both the sender and receiver know the order and hidden content of the files. Under these controlled conditions, decoding proceeds without alignment issues, and the overhead of managing multiple files is minimal.

While the proposed BCS-3 algorithm preserves the quality of the stego audio under adversarial conditions, the robustness of the recovered HCD remains challenging [35]. The key reason for preserving perceptual quality is that only 3 bits out of 24 bits in each audio sample are affected by the embedding process. However, while decoding works perfectly under controlled conditions where the sender and receiver know the order of the files, in real-world noisy environments, synchronization among multiple stego audio files may be challenging, potentially causing misalignment during decoding and affecting data recovery. Future work will focus on improving overall robustness to address both hidden data recovery and multi-file synchronization challenges.

4.7.4 Subjective auditory evaluations

The Mean Opinion Score (MOS) was employed as the subjective evaluation metric for the BCS-3 algorithm. MOS provides a numerical measure of audio quality on a scale from 1 (very Annoying) to 5 (Imperceptible), as shown in Table 10 [41]. The final score is obtained as the arithmetic mean of all listener ratings.

In this study, a total of 25 pairs of audio files (cover and stego) were evaluated. Listening tests involved 30 participants from diverse backgrounds, including audio engineers, industry professionals, academicians, researchers, and students. Each participant was asked to carefully evaluate paired audio signals (cover and stego) and assign quality ratings.

The results, presented in Table 10, indicate consistently high MOS values across all test cases, confirming the strong perceptual transparency of the proposed algorithm. The high MOS values are attributed to the fact that three characters (24 bits) are compressed into 8 bits using the proposed 24×8 compression algorithm. These 8 bits are then embedded into 24-bit audio samples, resulting in only a 3-bit modification per sample.

Figure 19 provides a comprehensive comparison chart outlining performance metrics between the existing and proposed algorithms. The chart employs the y-axis to represent performance metric values for both the proposed and existing algorithms, and the x-axis denotes references and names for the performance metrics. The proposed 24x8 compression algorithm aims to reduce three characters into a single character, which enhances the embedding capacity for audio steganographic communication.

The proposed BCS-3 algorithm aims to ensure that the length of hidden data in bytes and the differing bits between the cover and stego audio files are equal. This approach enables predictable bit errors during the embedding process, resulting in 15000 error bits for 15000 bytes of healthcare data. This helps to predict bit errors in advance and establishes a consistent standard. Additionally, the algorithm maintains a standard by embedding three characters (compressed into one character) with a three-bit difference in the corresponding audio samples.

This results in a well-maintained trade-off between embedding capacity and the integrity of the audio file. Furthermore, the total number of audio samples required for embedding should be one-third the length of the embedded data, as demonstrated in Table 8 and Figure 20. The BCS-3 algorithm embeds data in audio samples with a 3-bit difference, reducing the iterations needed to select audio samples for data embedding.

The proposed method introduces innovative algorithms like I-OVA and D-RVA for secure encryption and decryption, the Merging Algorithm (MA) for optimizing data representation, the 24x8 Compression Algorithm for compressing three characters to one character, BCS-3 and BCR-3 for systematic embedding and retrieval, and an enhanced LSB algorithm for embedding probability distribution and indexing audio samples securely. Together, these algorithms ensure double-layer security

with efficient utilization of audio samples by embedding three characters of 24 bits as 8-bit encoded values in an audio sample of 24 bits. This approach results in an error bit of 3, with each character contributing one error bit. This results in a relationship where the HCD length in bytes corresponds to error bits, resulting in a 3:1 utilization ratio of audio samples.

Addressed novelty and research gap

The proposed work addresses a research gap in telemedicine by overcoming the limitations of existing audio steganography techniques for secure healthcare data communication. Traditional LSB algorithms exhibit inefficiencies in embedding capacity, inconsistent error patterns, and a lack of integrated security mechanisms. Image steganography is commonly used in telemedicine, while audio steganography is less explored for secure healthcare data communication.

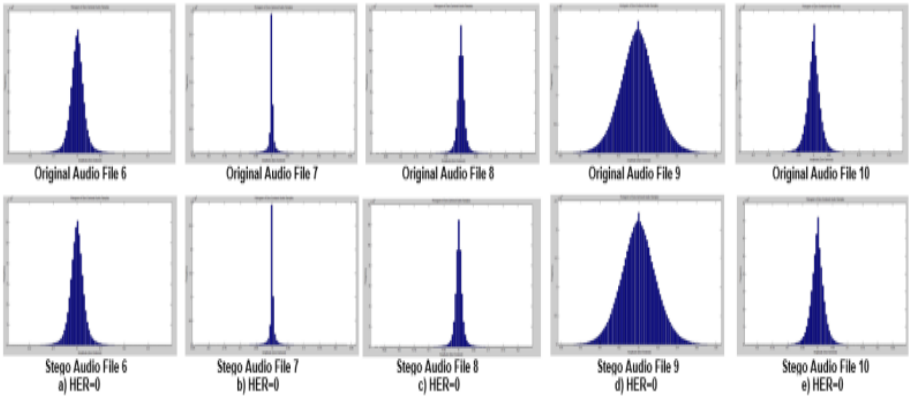


Figure 18: Histograms for various cover audios and stego audios using 15000B Healthcare data

Table 10: MOS grading scale and scores for HCD payloads (3000–15000 bytes) across various audio files

MOS Grade	1	2	3	4	5
Remarks	Very Annoying	Annoying	Slightly Annoying	Perceptible Not Annoying	Imperceptible
Source audio	A6	A7	A8	A9	A10
MOS Grade	5	5	5	5	5

Table 11: Comparison of DR for the Fourth First Moments and HER for various cover audios using 15000B of concealed HCD

Audio File	Difference Ratio (DR)				HER
	(μ)	(σ^2)	(sk)	(k)	
[32]	0.0799 ▲99.77%	0.0008 ▲99.99%	0.0018 ▲99.96%	0.0028 ▲99.99%	0.1278 ▲100%
[33]	0.2304 ▲99.92%	0 ≈	0.0004 ▲99.83%	0.0005 ▲99.99%	1.2274E-04 ▲100%
Proposed	**1.8331E-04**	**2.5759E-09**	**6.7953E-07**	**3.5282E-09**	**0**

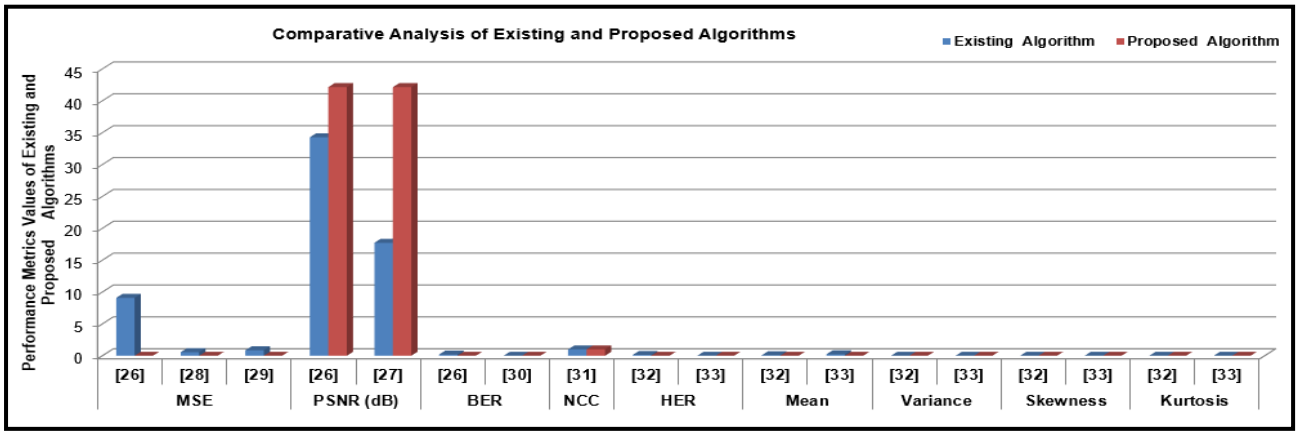


Figure 19: Comparative analysis of proposed and existing algorithms

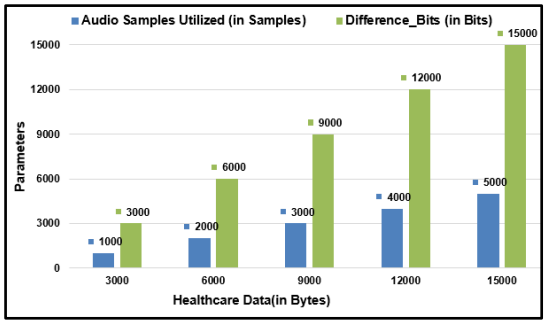


Figure 20: Influence of parameters of audio files versus hiding capacity with various text and audio files

This work addresses these challenges by introducing a robust and efficient framework designed for telemedicine applications. The framework proposes a novel method for communicating compressed healthcare data in telemedicine by integrating cryptography and steganography.

B The significance of the proposed work

The integration of cryptography and steganography provides double-layer security for healthcare data.

The proposed method compresses 15000B of healthcare data into 5000B of encoded values, achieving a threefold reduction and improving communication efficiency.

Efficient utilization of audio samples enhances the embedding capacity while maintaining audio quality.

The approach ensures consistent and predictable error bit patterns, enhancing the reliability of the embedding process.

The framework communicates compressed HCD, making it adaptable to various telemedicine scenarios in diverse network environments.

The overall work establishes a relationship between the HCD size, the audio samples utilized, and the error bits introduced in the embedding of HCD. This insight allows for systematic planning of the entire healthcare communication process, minimizing the potential for steganalysis and enhancing the security of the transmitted data.

5 Conclusion and future work

This paper proposes a model for secure healthcare data transmission using audio steganography for telemedicine. The 24x8 compression algorithm optimally uses audio samples to increase embedding capacity by compressing three characters into one. The model utilizes three audio files. The first audio file transmits the run-length encoding total count, individual count values, and corresponding data values using the least significant bit (LSB) algorithm with Ks as reference values. The metrics supporting the algorithm are the PSNR: 30.1417 dB and BER: 0.001118. The second audio file transmits 8-bit healthcare data (HCD) as encoded values using the BCS-3 algorithm. The compressed 1 byte (8 bits) of healthcare data from 3 bytes (24 bits) is further embedded within three bytes of the audio file. The average MSE and PSNR for the BCS-3 algorithm are 6.5297E-14 and 42.1480 dB, respectively. The BCR-3 algorithm retrieves the hidden HCD, with its efficiency indicated by a BER of 7.0165E-5 and an NCC value of 0.999999. The third audio file transmits the index values of stego audio samples containing HCD as encoded values, again using the LSB algorithm with Ks as the reference. The efficiency of algorithms is proved with the metrics PSNR: 36.5499 dB and BER: 0.000258.

Healthcare data (HCD) is encrypted and decrypted using the I-OVA and D-RVA algorithms to ensure data security. When transmitting 15000B of HCD, the algorithms achieve a throughput of 8592.74 KB/s. The 24x8 compression algorithm reduces the audio samples required for embedding healthcare data (HCD), resulting in 5000 samples for 15000B of data compared to the least significant bit algorithm.

The prevalence of 3 differing bits in the BCS-3 algorithm reduces the complexity of the process and the computational iterations in computing audio samples for embedding healthcare data (HCD).

The comparative methodology of the proposed BCS-3 algorithm resulted in identical values. With an HCD comprising 15000B (15000 characters), the audio samples utilized for embedding are 5000 samples. The difference in bits between the cover and stego audio files containing concealed data of 15000B is 15000 bits. Future research could explore real-time transmission of healthcare data using an audio steganographic system integrated with IoT for telemedicine applications.

Acknowledgement

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

Data availability statement

Audio Files: https://mixkit.co/free-sound-effects/
HCD:https://www.kaggle.com/datasets/xuehaihe/covid-dialogue-dataset?select=COVID-Dialogue-Dataset-English.txt

References

[1] Rumi Chunara, Yuan Zhao, Ji Chen, Katharine Lawrence, Paul A Testa, Oded Nov, and Devin M Mann.Telemedicine and healthcare disparities: a cohort study in a large healthcare system in New York City during COVID-19.Journal of the American Medical Informatics Association. 28(1):33-41,2021. https://doi.org/10.1093/jamia/ocaa217

[2] Abid Haleem, Mohd Javaid, Ravi Pratap Singh, and Rajiv Suman. Telemedicine for healthcare: Capabilities, features, barriers, and applications. Sensors International,2(100117):1-12,2021. https://doi.org/10.1016/j.sintl.2021.100117

[3] A. S. Albahri, Jwan K. Alwan, Zahraa K. Taha, Sura F. Ismail, Rula A.Hamid, A. A. Zaidan, O. S. Albahri , B. B. Zaidan, A. H. Alamoodi, and M. A. Alsalem .IoT-based telemedicine for disease prevention and health promotion: State-of-the-Art.Journal of

Network and Computer Applications.173(102873),2020. https://doi.org/10.1016/j.jnca.2020.102873

[4] Malak Alkhudaydi, and Adnan Gutub. Securing Data via Cryptography and Arabic Text Steganography.SN Computer Science .2(46):1-18, 2021. https://doi.org/10.1007/s42979-020-00438-y

[5] Hamouda B.E.H.H. Comparative Study of Different Cryptographic Algorithms. Journal of Information Security.11(3):138-148,2020. https://doi.org/10.4236/jis.2020.113009

[6] Abdalbasit Mohammed Qadir, and Nurhayat Varol. A Review Paper on Cryptography .2019 7th International Symposium on Digital Forensics and Security (ISDFS).IEEE, 2019. https://doi.org/10.1109/isdfs.2019.8757514

[7] Sazeen T. Abdulrazzaq, Mohammed M. Siddeq, and Marcos A. Rodrigues. A Novel Steganography Approach for Audio Files.SN Computer Science. 1(97):1-13,2020. https://doi.org/10.1007/s42979-020-0080-2

[8] Hrishikesh Dutta, Rohan Kumar Das, Sukumar Nandi, and S. R. Mahadeva Prasanna. An Overview of Digital Audio Steganography. IETE Technical Review.37(6):632-650,2019. https://doi.org/10.1080/02564602.2019.1699454

[9] Dingwei Tan, Yuliang Lu, Xuehu Yan and Xiaoping Wang. A Simple Review of Audio Steganography. 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC).1409-1413,2019. https://doi.org/10.1109/ITNEC.2019.8729476

[10] Ujala Razaq, Xu Lizhong, Changli Li, and Muhammad Usman.Evolution and Advancement of Arithmetic Coding Over Four Decades. Open Journal of Science and Technology. 3(3):194-236, 2020. https://www.researchgate.net/publication/346441614_EVOLUTION_AND_ADVANCEMENT_OF_ARITHMETIC_CODING_OVER_FOUR_DECADES

[11] Uthayakumar Jayasankar, Vengattaraman Thirumal, and Dhavachelvan Ponnurangam.A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. Journal of King Saud University-Computer and Information Sciences.33(2):119-140,2021. https://doi.org/10.1016/j.jksuci.2018.05.006

[12] Tonny Hidayat, Mohd Hafiz Zakaria, and Ahmad Naim Che Pee .Comparison of Lossless Compression Schemes for WAV Audio Data 16-Bit between Huffman and Coding Arithmetic.International Journal of Simulation: Systems, Science & Technology .19(6):36.1-36.7,2019. https://doi.org/10.5013/ijssst.a.19.06.36

[13] 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).100316:1-27,2020. https://doi.org/10.1016/j.cosrev.2020.100316

[14] Indy Haverkamp, and Dipti K. Sarmah, Evaluating the merits and constraints of cryptography-steganography fusion: a systematic analysis.International Journal of Information Security.23:2607-2635,2024; https://doi.org/10.1007/s10207-024-00853-9

[15] Enas Wahab Abood, Abdulhssein M. Abdullah , Mustafa A. Al Sibahee , Zaid Ameen Abduljabbar, Vincent Omollo Nyangaresi , Saad Ahmad Ali Kalafy, and Mudhafar Jalil Jassim Ghrabta. Audio steganography with enhanced LSB method for securing encrypted text with bit cycling.Bulletin of Electrical Engineering and Informatics.11(1):185-194,2022. https://doi.org/10.11591/eei.v11i1.3279

[16] Dilip Kumar Sharma, Ningthoujam Chidananda Singh, Daneshwari A Noola, Amala Nirmal Doss , and Janaki Sivakumar. A review on various cryptographic techniques & algorithms. Materials Today:Proceedings.51(1):104-109,2022. https://doi.org/10.1016/j.matpr.2021.04.583

[17] Mohanad Sameer Jabbar, and Samer Saeed Issa. A crypto-steganography healthcare management: towards a secure communication channel for data COVID-19 updating. Indonesian Journal of Electrical Engineering and Computer Science.29(2):1102-1112,2023. https://doi.org/10.11591/ijeecs.v29.i2.pp1102-1112

[18] Reihane Saniei, and Karim Faez. The Security of Arithmetic Compression Based Text Steganography Method. International Journal of Electrical and Computer Engineering (IJECE).3(6):797-804,2013. https://ijece.iaescore.com/index.php/IJECE/article/view/5425/4932

[19] Wafaa Al-Chaab, Zaid Ameen Abduljabbar, Enas Wahab Abood, Vincent Omollo 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

[20] Nathaniel Fout, and Kwan-Liu Ma. An Adaptive Prediction-Based Approach to Lossless Compression of Floating-Point Volume Data. IEEE Transactions on Visualization and Computer Graphics,18(12):2295-2304,2012. https://doi.org/10.1109/TVCG.2012.194

[21] Mostafa A. Ahmad, Mourad Elloumi , Ahmed H. Samak , Ali M. Al-Sharafi, Ali Alqazzaz , Monir Abdullah Kaid ,and Costas Iliopoulos. Hiding patients' medical reports using an enhanced wavelet steganography algorithm in DICOM images. Alexandria Engineering Journal.61(12): 10577–10592,2022. https://doi.org/10.1016/j.aej.2022.03.056

[22] Anne Y. Ning, Claudia I. Cabrera, and Brian D Anza. Telemedicine in Otolaryngology: A Systematic Review of Image Quality, Diagnostic Concordance, and Patient and Provider Satisfaction. Annals of Otology, Rhinology & Laryngology.130(2):195–204,2020. https://doi.org/10.1177/0003489420939590

[23] Medapati Venkata Manga Naga Sravan, and K Venkata Rao.5G-Optimized Deep Learning Framework for Real-Time Multilingual Speech-to-Speech Translation in Telemedicine Systems. Informatica, 49 (2):279–298, 2025. https://doi.org/10.31449/inf.v49i2.7826

[24] Mahmoud Magdy, Khalid M. Hosny, Neveen I. Ghali, and Said Ghoniemy. Security of medical images for telemedicine: a systematic review. Multimedia Tools and Applications.81:25101–25145, 2022. https://doi.org/10.1007/s11042-022-11956-7

[25] K Revathi, S. Kaja Mohideen, and Latha Tamilselvan. Efficient Audio Steganography Using Bit Comparison and Substitution Algorithms for Improved Embedding Capacity. Informatica, 49 (21)**:** 179–198, 2025. https://doi.org/10.31449/inf.v49i21.6957

[26] Bambang Harjito, Beni Sulistyarso, and Esti Suryani. Audio steganography using two LSB modification and RSA for security data transmission. Journal of Telecommunication, Electronic and Computer Engineering (JTEC),10(2–4):107–111,2018. https://jtec.utem.edu.my/jtec/article/view/4326/3173

[27] 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

[28] Ratul Chowdhury, Debnath 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] Hemalatha S, U Dinesh Acharya, Renuka A, Deepthi S, and Jyothi Upadhya K.Audio steganography in discrete wavelet transform domain. International Journal of Applied Engineering Research. 10(16):36639–36644,2015. https://www.ripublication.com/ijaer10/ijaerv10n16_85.pdf

[30] 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

[31] Ahmed Hussain Ali, Mohd Rosmadi Mokhtar, and Loay Edwar 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

[32] Ahmed Hussain Ali , Loay Edwar George, A. A. Zaidan, and Mohd Rosmadi 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

[33] Haider Ismael Shahadi, Razali Jidin, and Wong Hung 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

[34] Xubao Zhang. Designs, experiments, and applications of multichannel structures for hearing aids. EJECE, European Journal of Electrical Engineering and Computer Science. 5(4): 46–55,2021. https://doi.org/10.24018/ejece.2021.5.4.347

[35] Mahmoud M. Mahmoud, and Huwaida 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

[36] Chakravorty, Subrato and He, Xuehai and Yang, Xingyi and Xie, Pengtao. 2020. Chakravorty2020COVIDDialogueDatasetEnglish.COVID-Dialogue-Dataset-English: an English medical dialogue dataset about COVID-19 and other types of, pneumonia. https://github.com/UCSD-AI4H/COVID-Dialogue.2020. https://www.kaggle.com/datasets/xuehaihe/covid-dialogue-dataset?select=COVID-Dialogue-Dataset-English.txt

[37] Thoyazan Sultan Algaradi, and B. Rama. A New Encryption Scheme For Performance Improvement In Big Data Environment Using Mapreduce. Journal of Engineering Science and Technology.16(5):3772 – 3791,2021.https://jestec.taylors.edu.my/Vol%2016%20Issue%205%20October%202021/16_5_11.pdf

[38] R. Ramya Devi, and V. Vijaya Chamundeeswari. Triple DES: Privacy Preserving in Big Data Healthcare. International Journal of Parallel Programming.48:515-533, 2018. https://doi.org/10.1007/s10766-018-0592-8

[39] Muhammad Yunus, Intan Sulistyaningrum Sakkinah, Ulfa Emi Rahmawati, Atma Deharja , and Maya Weka Santi. File Security Design in Electronic Health Record (EHR) System with Triple DES Algorithm (3DES) at Jember Family Health Home Clinic.1(1):1-8,2023. https://doi.org/10.47134/ijhis.v1i1.2

[40] Ahmed A. Alsabhany, Farida Ridzuan, and A. H. Azni, The Progressive Multilevel Embedding Method for Audio Steganography, Journal of Physics: Conference Series, 1551 012011,2020. https://doi.org/10.1088/1742-6596/1551/1/012011

[41] Dipankar Pal, and Nabin Ghoshal. A Robust Audio Steganographic Scheme in Time Domain (RASSTD). International Journal of Computer Applications (0975 8887), 80(15):1-8, 2013. https://doi.org/10.5120/13934-1803

.