

# An adaptive audio wave steganography using simulated annealing algorithm

**Atef Ahmed Obeidat, Mohammed Jazi Bawaneh, Sawsan Yousef Abu Shqair,  
Hamdi A. Al-Omari, Emad Fawzi Alshalabi**

Information Technology Department, Al-Huson University College, Al-Balqa Applied University, Al-Huson, Jordan

---

## Article Info

### Article history:

Received Jul 3, 2024

Revised Dec 5, 2024

Accepted Dec 14, 2024

---

### Keywords:

Audio wave

Encryption

Least significant bit

Security

Simulated annealing

Steganography

---

## ABSTRACT

The science of information security has increased in importance to encounter the espionage and information theft. This research proposes a new steganography framework that utilizes simulated annealing (SA) as an artificial intelligence algorithm to support the process of hiding a binary secret message file within an audio wave file. The best path for embedding the secret data inside the audio file is determined through SA that searches for the preferred path according to the content of the host audio file and secret message to be hidden. The least significant bit (LSB) technique was employed to hide message bytes, in which each audio-chosen byte will hold one bit from a secret message byte. The hiding process constructs the stego audio file and extraction key that will be required in an extraction process. The authorized user requires an extraction key and a decryption key to retrieve the hidden message. On the other hand, the attacker requires knowledge of the aforementioned keys and working algorithms that were employed in the hidden process. Robustness against data extraction, detection, imperceptibility (phonological hearing), security, peak signal to noise ratio (PSNR), mean square error (MSE) and capacity as security performance measures were used to evaluate the system. The maximum size of the data to be hidden may reach 12.5% of the data size of the host audio file, in which the average value of MSE and PSNR are (0.0041, 74.73), respectively.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



---

## Corresponding Author:

Mohammed Jazi Bawaneh

Information Technology Department, Al-Balqa Applied University, Al-Huson University College

Al-Huson, 21510, Jordan

Email: dr\_mjab@bau.edu.jo

---

## 1. INTRODUCTION

The internet is considered a shared channel to transmit and receive data between different parties, where data across this network is often violated and attacked. Much of the data that we deal with and exchange through the internet needs some kind of privacy and security, especially the sensitive ones. To address and reduce data violation problems, attacks, deprivations, data blocking, denial of services, and any other problems related to cyberspace security several methods and technologies have been invented such as cryptography, steganography, and watermarking. They are considered the most common and important techniques used in cybersecurity to maintain data and information security, as they all seek to achieve the same goal but by processing data in different ways.

Cryptography is the process of converting data into a visible form that can be read only by authenticated user who has the reconverting key. Cryptography method depends on a single key shared between the sender and the receiver, called the public key, or it can depend on more than one key, so that

there is a private key for each of the sender and the receiver in addition to a public key shared between them. Methods of cryptography range from the simple, which require a little period before the attacker can decrypt data, to the complex, which depend on more than one key and mathematical calculations, which makes the decryption process require a long period. In all cases, the attacker can see the data, but in an encrypted form that cannot be read until it is decrypted. Therefore, the encryption process gives the attacker the impression that there is communication between the sender and the receiver, giving him the opportunity to try to access the data [1].

Digital watermarking inserts visible or covert code or images inside multimedia files in an aim to keep the of copyright for data. The attacker or thief can see the watermark within the data and can sometimes get rid of it by adding some data that works to hide it, especially if this mark is in a neutral or peripheral area and does not interfere with the data. Therefore, the owner of the right often works to enter more than one watermark and sometimes makes some of them interfere with the original data for the purpose of protection [2].

Steganography is considered an old and new technology for transmitting invisible messages. In the past, wax, wood, animal skins, and slave heads were used to hide information in a secret way and then transfer them to different geographical areas [1]–[3]. These days, in the digital world that contains different types of data files, such as images, audio, video, and text, the process has shifted from using hard media to soft ones where any type of file can be used as a host for secret messages. The process of hiding messages can be classified in different ways depending on host file type or hiding method. If we look at the host of data, it can be classified into text, image, audio, video, or IP steganography, while methods used for hiding could be injection, replacement, distortion, or generation [4]–[6].

The main problem in the Internet is how to transfer and exchange data securely between the sender and the receiver. Data can be transferred in a visible but encrypted form (cryptography), in a hidden form (steganography), or in a hidden and encrypted form (steganography and cryptography). This research focused on the third form, which is a combination of steganography and cryptography with the aim of participating in providing a solution for secure communication.

This paper proposes a reliable and secure framework that combines artificial intelligence, cryptography, and steganography for embedding secret messages within a wave audio file. The framework uses the simulated annealing (SA) algorithm, which is an optimization search algorithm that is used to find the solution to linear and non-linear problems [7]. It depends on cooling, heating, malleability, and processing metals, especially iron. The ductility of metal is high and can be easily formed at high temperatures, while the opposite is true at low temperatures. In the initial state at high temperature (worst case), the SA method works on choosing a random solution or path and considering it as the best solution and then works to reduce temperature and replace the current solution if a better one occurs. The process continues with cooling, comparison, and selection until the best solution is reached at minimum temperature (best case), where the comparison process between solutions is managed through a specific conjunction called a fitness function. The SA algorithm searches the bytes of the host audio file in order to find the suitable places for the embedding process. The path of least significant change to content will be selected for storing the encrypted secret message bits by using the least significant bit (LSB) method. LSB as an embedding method works by replacing the last bit on the right side, as it has the least effect on the value of the byte that is used to hide part of the secret message. Message bytes are encrypted by using a replacement algorithm, which bases on predefined common table between sender and receiver. It works in a manner similar to Caesar algorithms that uses shifting and replacement process for data encryption and decryption [4], [5].

The proposed system can handle any type of secret messages because the message data is read and processed as binary data; therefore, the secret message can be text, image, sound, or any other type of data. Hidden and extraction processes require a set of predefined inputs to accomplish their tasks. Secret message file, host wave file, encryption key and SA parameters are the main requirements for hidden process. On the other hand, extraction process needs for stego wave file, extraction key, decryption key, and SA parameters as inputs.

## 2. RELATED WORKS

Many studies have been published in the field of steganography, whether images, audio, video or texts. Previous studies have included working on the topic of steganography separately or integrating the topic of steganography with cryptography or integrating the topics of machine learning and artificial intelligence for improvement and acceleration. Several researches have been reviewed, whether similar to us or different in terms of the mechanism of work and host files of various types. Below, a group of these studies will be reviewed, which focus specifically on the topic of audio steganography.

Bawaneh [7] proposed a steganography framework that uses the SA algorithm and linear congruent generator (LCG) to find the best for embedding a binary message inside a host image. The system works to find the path that is least affected by the modifications that occur as a result of embedding the secret message inside the image, where it works to use the SA algorithm to find this path and then embedding using the LSB algorithm. As shown in the results, the system was robust against intruders because the decoding process requires knowledge about extraction keys and used algorithms.

Manjunath *et al.* [8] proposed a new method for hiding audio information using discrete waves, which works to determine the appropriate locations to hide data after encrypting it to be more secure and reduce the used storage space. The feasibility of the proposed model was measured through a set of metrics, such as peak signal to noise ratio (PSNR) and mean square error (MSE). According to the evaluation measures used, the proposed system has good results in terms of performance and feasibility compared to other methods.

Nasr *et al.* [9] proposed an innovative method for hiding information in audio files, which filters sound waves before including the message in them and uses Wiener filtering to reduce noise on the audio signal. The secret message was filtered and converted it into a form that suits the medium used. Feasibility and effectiveness of the proposed method were measured through MSE and another group of benchmarks. After comparing results with previous methods, it was revealed that the proposed technique was successful in hiding data.

Mohammed [10] presented a new method for hiding data based on hiding the text inside an image and then hiding it in the audio file. The author worked on encrypting text by taking the reverse code of each letter and then replacing each bit from the right side of the text, and finally embedding the text in an image. The image was used as a text host and the audio file as an image host, where this work is characterized using two stages of masking. The proposed system was robust against data extraction due to multi stages of hiding, but slow in the process of hiding as shown in the results.

Wu *et al.* [11] proposed a new steganography method for audio based on the idea of negative assaults, which firstly uses a neural network to train the proposed system and secondly constructs stego audio files. The proposed system merged artificial intelligence (neural network) with steganography in order to produce robust and high-performance system. The results showed that the proposed system gave good results compared to other systems, as quality and safety measures were used to examine the system.

Jiang *et al.* [12] proposed an audio steganography method that uses three neural networks: the first one was utilized for embedding a secret message in the host file, the second for extracting and removing the secret message, and the third for determining the secret message host. The proposed system integrates neural network with steganography to protect secret message from external or internal attacks, which results in a robust system. The system works through several layers in the neural network, whether at the level of hiding or extraction. To measure the accuracy of the system, related security criteria such as MSE and PSNR were used, and the results showed the quality of the system compared to other systems.

Yi *et al.* [13] proposed and adapted the framework of Huffman Code, which is a method that constructs the code space checking data inserting process and utilizes an entropy code with the same length. It employed a stego key for modifying Huffman code mapping to manage statistical undetectability. The key was used on the data to be hidden in order to build a special structure for the data that in turn re-arranges the data, which will reduce the size of the data, which reduces the chance of suspicion in the file carrying the hidden data. The results showed the quality of the system compared to other systems using security standards related to the size of the hidden data as well as the chance of extraction and tracking it.

Wazirali *et al.* [14] proposed a steganography technique that uses a genetic algorithm and fitness function to improve similarity between the host file and secret message through the different stages of the embedding process. In this paper, the authors tried to find the possibility of similarity within the host by selecting random vectors and then comparing them with the data to be hidden in order to obtain the least possible variations within the host medium, which in turn will reduce the suspicion of the presence of hidden data. The results indicate through the use of MSE and PSNR good results and high ability to store raw messages with minimal modifications.

Bawaneh [15] proposed a steganography framework that uses the intelligent water drop (IWD) algorithm and LCG to search for the best hosting path inside a grayscale image. The best path was utilized to store the secret message data inside the host image. LCG algorithm determines the paths randomly for IWD. The comparison between path data and secret message data is done by IWD in aim to find the minimum modified one. The system was robust against intruders because the decoding process requires knowledge about extraction keys and used algorithms.

Ali *et al.* [16] presented an audio steganography framework based on the idea of embedding a secret message audio within uniform input audio. They used fractal coding for data compression and produced a high level of it. The fractal coding was used to compress the secret message bytes and then reordering them inside the host audio file according to this coding. The proposed system, as stated in the results, succeeded in

reducing the size of the secret message data through the compression process, which led to a reduction in the percentage of changes in the host file, as well as making the process of extracting the message by the attacker more complicated, due to its reliance on the fractal coding.

Bawaneh [17] proposed a random steganography system for images that use LCG to distribute data randomly inside the host file. The constructed system inserts data of secret messages inside colored images according to locations selected via LCG. Seed, multiplier and non-common factor refer to parameters of LCG algorithm, which are considered as the hidden and extraction keys for secret message. Results indicated that system was robust in terms of extraction and detection, due to the randomly selected locations and satisfying the main security requirements.

Bharti *et al.* [18] proposed a novel approach for audio steganography that uses a secret message and host file as audio ones. They used the samples of host file to embed the bytes of secret message and suggested inserting one bit from the secret message into each sample from the host file. The approach was robust against LSB removal and resampling attacks, and in term of the perceptual evaluation of speech quality (PESQ) and mean opinion score (MOS) the proposed method gave a good score as mentioned in the paper result.

Kanaujiya *et al.* [19] proposed a method for steganography that combines between steganography and cryptography. They focused on audio wave as a host file and LSB for embedding secret message data. Before hiding the message data, it is encrypted using the Rivest-Shamir-Adleman (RSA) algorithm and then embedded in the audio file in a sequential manner using the LSB algorithm, where the hiding process starts after the audio file header. The strength of this model lies in the encryption method and not the hiding method, as it is sequential. The combination in the proposed framework provides two levels of security for secret messages in term of encryption and steganography as mentioned in the paper result.

Krishnan *et al.* [20] proposed an audio steganography method to improve LSB via the RSA encryption algorithm. They applied stego keys and cryptography keys to increase the security of the steganography process. Hidden and extraction processes are controlled by steganography and cryptography keys. A secret message was hidden inside an audio signal and modified in significant binary ordering of the audio file. The result shows that the proposed system was robust against attacks that might be carried out on the LSB algorithm due to the strength of the encryption algorithm, which is RSA, which is considered one of the strongest algorithms.

Shanthakumari *et al.* [21] proposed an audio steganography based on the LSB approach to hide data inside wave file, by proposing an algorithm that searches for less modification rate locations inside the wave file. The proposed algorithm uses artificial intelligence (heuristic search) and backtracking to determine the best path for hiding data. Each selected path given a rank or priority value that will be utilized in next step to specify the best one. Secret message bytes will be hidden inside the best path bytes in order to achieve the minimum modification rate. They utilized PSNR and MSE measures for result analysis that display the dependency between the host audio file and the length of the used secret message.

### 3. MATERIALS AND METHOD

The main goal of the proposed framework is to build a stego wave audio file that is robust and secure against data detection and extraction. The most appropriate solution to embed the secret data within the host wave audio file is found through the SA algorithm. Each created solution consists of a set of sequential bytes chosen from the host audio wave file such that the starting byte is selected randomly via the system. The role of the SA algorithm is to examine the constructed solutions and then determine the best one between them. To achieve the main goal, there is a set of steps that must be carried out for hidden and extraction processes, which represent the basic requirements for those processes as shown in ulterior subsections.

#### 3.1. Hidden process

Hidden process requires a set of initial information that will be distributed among their sub-units. The host wave audio file, secret message file, encryption key, and parameters of the SA algorithm represent the required information. It starts by ensuring the compatibility of the wave audio file and its ability to accommodate the secret message, constructing the new secret message file, encrypting the file to be hidden, sending the encrypted and host file to the SA algorithm that works to build the best solution of hosting bytes, embedding data within the host file in an aim to construct the stego wave audio file, and finally returning the stego file and extraction key that will be used later by the recipient. Figure 1 shows the framework of the hidden process and how tasks are distributed among their units, which will be discussed in detail in subsequent sections.

### 3.1.1. Wave decoder unit

It receives a wave audio file through the interface and then works to ensure that the file is valid for use in the data hiding process. The wave file architecture is a subset of the Microsoft resource interchange file format (RIFF) specification for storing multimedia data [22], which begins with a file header followed by a set of chunk data. Most often, the wave file consists of a set of sections, where each part gives specific information about the file. These sections are usually divided into three types:

- a. RIFF section: this part has a signature at the beginning of the file that contains four letters, "RIFF," in an aim to distinguish the file type, file size, and data type used within the file through a set of fields as follows:
  - ChunkSize, it ChunkID, it stores the signature "RIFF" in ASCII form.
  - refers to file size, except the size ChunkID and ChunkSize of "fmt" and data.
  - Format, it stores the signature "WAVE."

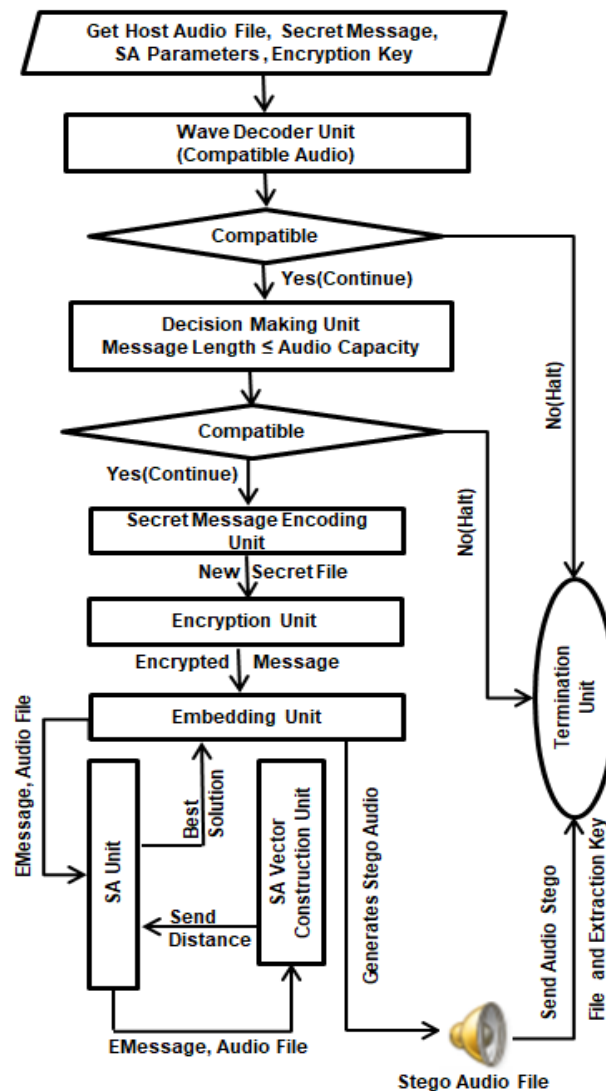


Figure 1. Framework of hidden process

- b. "fmt" section, this part defines the organization and data format. It starts with the signature "fmt" to determine the beginning of the current section and also holds information about the section size, audio format, sample rate, number of used channels inside the file, bit rate, block alignment, and bits per sample as follows:
  - Subchunk1ID stores the signature "fmt"
  - Sub Chunk1 Size, it refers to the total size of fields which follow this one in the same section.

- AudioFormat stores value 1 to indicate linear quantization or any other value to indicate compression.
  - NumChannels determines the number of channels (Mono = 1, Stereo = 2).
  - SampleRate defines the rate of data sampling in the file, such as 8000, 44100, etc.
  - ByteRate is computed from sample rate, number of channels, and bits per sample.
  - BlockAlign determines the number of bytes for one data sample involving all channels of data within the file, which is computed through the number of channels and bits per sample.
  - BitsPerSample determines the number of bits in a data sample (8 bits=8, 16 bits=16).
- c. Data section, this part stores the actual audio data, and it starts with signature “data” to determine the beginning of current section, and it also holds information about section size and audio data; as follow:
- Subchunk2ID stores the signature “data.”
  - Sub Chunk2 Size determines the data size (number of bytes) in the data field.
  - Data stores the real sound data of the audio file.

There is more than one type of wave file that works with different lengths, such as PCM-8 bits, PCM-16 bits, PCM-24bits and PCM-32. Dealing with the different lengths of wave file requires a lot of effort; therefore, we focused on manipulating and processing only PCM-16 bits in which the header size is 44 bytes. Figure 2 shows the structure of required the host wave file, which was used to construct the WAVE unit decoder.

Byte Offset	Field Name	Field Size(Bytes)	Section Descriptor
0	Chunk ID	4	RIFF Chunk
4	Chunk Size	4	
8	Format	4	
12	Sub Chunk1 ID	4	fmt Sub Chunk
16	Sub Chunk1 Size	4	
20	Audio format	2	
22	Number of Channels	2	
24	Sample Rate	4	
28	Byte Rate	4	
32	Block Align	2	
34	Bits Per Sample	2	Data sub Chunk
36	Sub Chunk2 ID	4	
40	Sub Chunk2 Size	4	
44	Data	Sub Chunk2 Size	

Figure 2. Wave file format

### 3.1.2. Decision unit

Based on results returned from the Wave Decoder unit, the Decision-Making unit works to determine the continuity of work whenever the host file is compatible. If work continues, the Decision-Making Unit works to calculate the number of available bytes in the host audio file, as well as the number of bytes needed for the secret message in order to perform the hiding process. The available number of bytes in the host file is calculated after excluding the host file header (HFH), which is (44) bytes, such that the remaining bytes are computed and divided by (8) because each byte from the secret message file requires (8) bytes from the host wave audio file as shown in (1). On the other hand, the number of bytes needed for the secret message file is calculated by determining the number of bytes in the file itself, then reserving an additional (13) bytes for signature, length, and extension (SLE) that will also be embedded in the host file, in which (3) bytes for message signature, (5) bytes for message length, and (5) bytes are for message extension. The result of the sum is multiplied by (8) because each byte from the secret message file, as mentioned previously needs, (8) bytes from the host wave audio file to hide it. Equation (2) shows how the required bytes are computed for the secret message file to be hidden. After that, the number of calculated bytes in the wave audio file and secret message file are compared to check suitability for continuity or not.

$$\text{Available Bytes} = (\text{Length Host File} - \text{HFH})/8 \quad (1)$$

$$\text{Required Bytes} = (\text{Length of Secret Message File} + \text{SLE}) * 8 \quad (2)$$

### 3.1.3. Secret message encoding unit

The role of this unit is to build the new secret message file, which consists of the secret message signature, length of the secret message, extension of the secret message, and actual data of the secret message as shown in Figure 3. The signature, length, and extension of the secret message file will be embedded in the new file and used in the extraction process. The signature has letters "SMF" that is utilized to check if the inserted audio file contains a secret message or not. Moreover, the extension of secret is employed to build the secret message file, while the length of the message determines the number of bytes to be extracted from the stego audio file.

Byte Offset	Field Name	Field Size(Bytes)
0	Message Signature	3
3	Message Length	5
8	Message Type	5
13	Message Data	Message Length

Figure 3. Format of new secret message file

### 3.1.4. Encryption unit

It receives the new secret message file after ensuring that the host wave audio file is able to hide it. The encryption process depends on an algorithm similar to the way the Caesar algorithm works [23], which is considered one of the simplest, most famous, and fastest algorithms. It relies on the principle of replacing values from a pre-prepared table with values shared between the sender and receiver, where the encryption key is a specific shift amount agreed upon between the two parties. The encryption algorithm used in current research works to build a shared table between the sender and receiver that stores random numbers in the period (0-255) without repeating the values. After that, each byte of the secret message is replaced within its corresponding value in the chosen location. Algorithm 1 shows how the encryption unit accomplishes their task and the mechanism in which the encryption algorithm works in terms of how to replace the values of the secret message.

#### Algorithm 1. Main steps of encryption unit

```

Step 1: Get SecretMessageFile
Step 2: Set Index=0
Step 3: If Index >= NewSecretFile.Length Then Goto Step8
Step 4: Set X=ReadByte()
Step 5: Set Encrypted_Byte= SharedTable[X]
Step 6: EncryptedFile.WriteByte(Encrypted_Byte)
Step 7: Goto Step3
Step 8: Return EncryptedFile

```

### 3.1.5. SA unit

SA unit receives the secret message file that was encrypted, as well as the host audio file, in order to find out the best path within the host file that will store the hidden message. The SA algorithm relies on a set of parameters that are considered essential in the process of determining the best solution [7] and summarized in maximum initial temperature (MaxTemperature), minimum stopping temperature (AbsoluteTemperature), and cooling coefficient (CoolingRate). The values of these parameters can be variable, obtained through the communication interface, or can be fixed, which was done in the current research. After obtaining the basic requirements for work to be done, the SA algorithm builds the initial vector that selects sequential bytes starting from a random location within the host audio file, which in turn forms the path that will contain the secret message. Distances between constructed solutions and secret messages will be computed and kept in aim to determine the best solution according to minimum distance value. It may come to mind that the best path sent to the hidden unit is a series of locations from the host audio file, but only the seed that created the best path will be sent to be used in building the best path again within the hidden unit. Subsequent sections explain how the SA algorithm works in terms of building solutions and determining the distance between the encrypted secret message and the solution. Algorithm 2 shows the main steps that the SA Unit will take to determine the best path that will be used later in the hidden unit.

**Algorithm 2. Main steps of SA unit**

```

Step1: Get HostFile, EncryptedFile and MaxIterations
Step2: Set MaxTemperature=10000
Step3: Set CoolingRate=0.99
Step4: Set AbsoluteTemperature=0.00001
Step5: Set Seed=Integer_Random_Value
Step6: Set BestSeed=Seed;
Step7: Set Distance=BuildSAVector(Seed, HostFile, EncryptedFile )
Step8: Set BestDistance=Distance
Step9: Set OldDistance=Distance
Step10: Set deltaDistance=0
Step11: Set Iteration=0
Step12: Set Temperature=MaxTemperature
Step13: If Temperature<=AbsoluteTemperature Then Goto Step 30
Step14: Set Seed=Integer_Random_Value
Step15: Set HostFile.Position=0;
Step16: Set EncryptedFile.Position=0;
Step17: Set NewDistance=BuildSAVector(Seed, HostFile, EncryptedFile )
Step18: Set deltaDistance=NewDistance-OldDistance
Step19: Set R=Random.Real (0,1)
Step20: If Not (deltaDistance<0 and BestDistance>NewDistance) Then Goto Step 24
Step21: Set BestDistance=NewDistance
Step22: Set BestSeed=Seed;
Step23: Goto Step 26
Step24: If Not ((OldDistance>0 && Math.Exp(-deltaDistance/Temperature)>R)) Then Goto Step 26
Step25: Set OldDistance=deltaDistance+OldDistance
Step26: Set Iteration=Iteration+1
Step27: Set Temperature=Temperature*CoolingRate
Step28: If Iteration>=MaxIterations Then Goto Step 30
Step29: Goto Step 13
Step30: Return BestSeed

```

SA vector construction unit gets the host audio file, the encrypted message file, and the seed value, which represents the starting position inside the host audio file. The vector construction process begins with a loop that reads the encrypted message byte by byte, in which each one is separated into (8) bits. After that, it reads (8) bytes sequentially from the host file starting from the seed position and then extracts the rightmost bits of them. The distance between extracted host bits and message bits will be computed and kept. The same process is repeated with all bytes of the encrypted message and host audio file, where each time the new distance is added to the retained distance, considering that the selected bytes cannot be repeated. Finally, the resulting distance is returned back to the SA unit. Algorithm 3 shows the complete process of vector construction and distance computation.

**Algorithm 3. Main steps of SA vector construction unit**

```

Step1: Get HostFile, EncryptedFile
Step2: Set Distance=0
Step3: Set Location=Seed
Step4: Define List SecretMessageBytes[8]
Step5: If End Of EncryptedFile Then Goto Step 18
Step6: Set SecretByte=EncryptedFile.ReadByte()
Step7: Set SecretMessageBytes=FindBits(SecretByte)
Step8: Set K=0
Step9: If K>=8 Then Goto Step 17
Step10: Set Location=(Location+1) mod LengthOfHostFile
Step11: Set HostFile.Position=Location+44
Step12: Set HostByte=HostFile.ReadByte()
Step13: Set HostBit=HostByte & 1
Step14: If SecretMessageBytes[K] != HostBit Then Distance=Distance+1
Step15: Set K=K+1
Step16: Goto Step 9
Step17: Goto Step 5
Step18: Return Distance

```

**3.1.6. Embedding unit**

After obtaining the encrypted secret message file, the host audio file, and the best solution seed, the unit starts the task of hiding. The embedding unit utilizes the least significant bit (LSB) technique to hide the bits of a given secret message byte inside the host audio file bytes. The LSB method is based on the idea of minimal modification impact on the content; therefore, it uses the rightmost bit of every byte, and therefore, the change is either by zero or one value to the host byte [17]. The embedding unit reads the encrypted secret message file byte by byte. Each byte that is read will be separated into its bits, and then each one replaces the



rightmost bit of host-selected byte. Host bytes are chosen sequentially starting from the best solution seed, taking into account avoiding host file header data. The process continues in reading message and host bytes until the entire message has been read and included in the stego audio file. In the end, the resulting stego audio file and the best solution seed, which is used as an extraction key for the hidden message are returned to the interface. Algorithm 4 shows the main steps of the embedding unit and how it works to accomplish its task.

#### Algorithm 4. Main steps of embedding unit

```

Step1: Get HostFile, EncryptedFile
Step2: Call SA Unit to get BestSeed
Step3: Set Location=BestSeed
Step4: Copy HostFile To stegoFile
Step5: Define List SecretMessageBytes[8]
Step6: If End Of EncryptedFile Then Goto Step 21
Step7: Set SecretByte=EncryptedFile.ReadByte()
Step8: Set SecretMessageBytes=FindBits(SecretByte)
Step9: Set K=0
Step10: If K>=8 Then Goto Step 20
Step11: Set Location=(Location+1) mod LengthOfHostFile
Step12: Set stegoFile.Position=Location+44
Step13: Set HostByte=stegotFile.ReadByte()
Step14: Set HostByte=(HostByte & 254)
Step15: Set HostByte=HostByte+SecretMessageBytes[K]
Step16: Set K=K+1
Step17: Set stegoFile.Position=Location+44
Step18: stegotFile.WriteByte(HostByte);
Step19: Goto Step 10
Step20: Goto Step 6
Step21: Return StegoFile, BestSeed

```

### 3.2. Extraction process

After ensuring that the entered audio file is suitable for work, the user must enter the decryption key, as well as the extraction key, where those keys consider the basic requirements to be able to extract the message via an authorized user. The extractor selects and reads (8) bytes from the stego file starting at the byte that is determined by the extraction key, extracts rightmost bits of them, builds one byte from those bits, sends the resulting byte to the decryption unit, and then returns the result to the extraction unit in the aim to be utilized in building the signature, extension, length, and data of the secret message. The extraction unit concatenates the first three returned bytes in order to build the character's secret message signature. If the constructed signature is not equivalent to the "SMF" letters, the process as a whole is canceled, while if they are equivalent, the unit continues to extract the length, extension, and data of the secret message from the same file. Embedded data within the stego audio file is divided into four ordered parts, which are the signature of the secret message, message length, message extension, and message data as shown in Figure 4. The embedded signature of the message is distributed between (24) bytes from the stego audio file, and the same applies to the length and extension of the message that require (40) bytes for each of them. Thereafter, determining the length and extension of the message, the process of extracting the message data begins by selecting the (8) bytes sequentially starting from the end byte of the message extension. The extracted decrypted byte will be stored in the secret message file that is created using the extracted message extension. The process of reading bytes, extracting the rightmost bits, building a byte, decrypting a byte and storing the resulting byte requires repeating along the retrieved message length value. Finally, after completing all the iterations, the secret message will be retrieved and sent to the interface as shown in Figure 5.

Fields Order and Size inside Stego Audio File	
Part	Size
Message Signature	3*8 Bytes = 24 Bytes
Message Length(ML)	5*8 Bytes = 40 Bytes
Message Extension	5*8 Bytes = 40 Bytes
Message Data	ML*8 Bytes

Figure 4. Size and order of embedded data

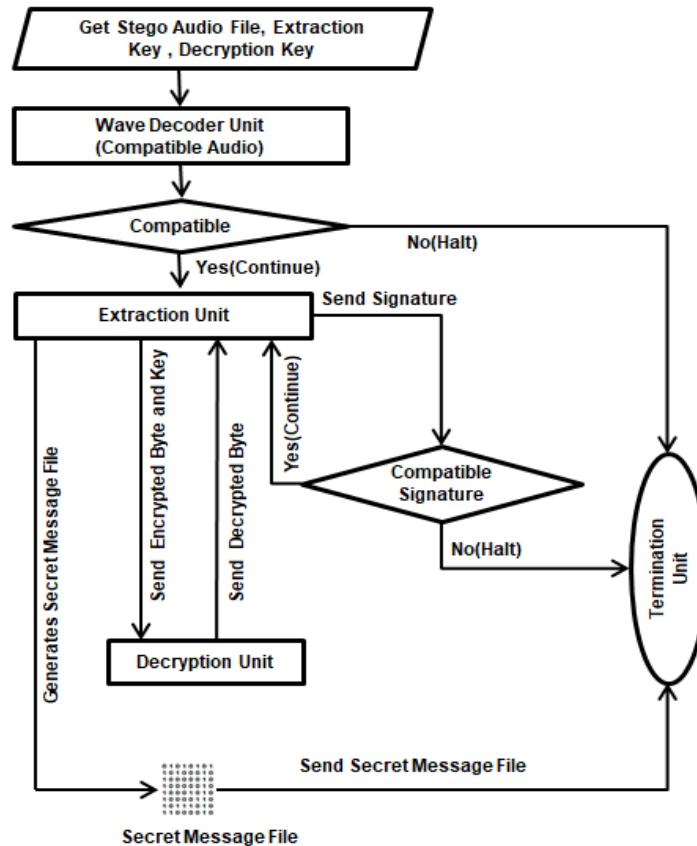


Figure 5. Framework of extraction process

The extraction process uses the decryption unit in order to decrypt the message bytes that were encrypted in a hidden process. In the initial state, the decryption unit builds the shared decoding table that must be common between sender and receiver, where the values are limited to the range (0-255), taking into account that the process of building this table takes place only one time, when this unit is called for the first time. After successfully completing the process of filling out the decoding table, it becomes possible to decode the sent values by comparing the sent value with the values stored in the shared decoding table, and the index of the corresponding value is returned to represent the decrypted value. Algorithm 5 illustrates the main steps of how an encrypted value will be decrypted.

#### Algorithm 5. Main steps of decrypting a given encrypted value

```

Step1: Get Decryption_Key, EncryptedByte
Step2: If FirstTimeCall Then Build SharedDecodingTable
Step3: Set Index=0
Step4: If EncryptedByte==DecodingTable[Index] Then Goto Step 7
Step5: Index=Index+1
Step6: Goto Step 4
Step7: Return Index
  
```

## 4. RESULTS AND ANALYSIS

The proposed framework was tested and evaluated through a selected set of audio files and secret message files of different sizes because there is no standard data set that has been relied upon in previous research. Table 1 displays the size of different secret message files (SMF) and the required bytes for each file; on the other hand, Table 2 presents the size of different host wave files (HWF) and available bytes in each one.

In order to evaluate the constructed steganography system, a set of common criteria's were used for verification and evaluation, which are robustness, detection, imperceptibility (phonological hearing), security, PSNR, MSE, and capacity [4], [24], [25]. In terms of robustness, the proposed framework is considered robust against data extraction because the extraction process requires knowledge about keys and

algorithms that were utilized to accomplish the hiding process; therefore, the extraction process without this knowledge is very difficult or impossible. On the other hand, any changes to the stego file in terms of size, data compression, contents modification, or formatting will cause the embedded secret message to be lost. In general, the constructed system and most existing steganography systems will be robust against data extraction due to the need for prior knowledge about the hidden process mechanism, but it is not resistant to the changes that may occur to the stego audio file.

The LSB method was used to embed secret message data inside the audio file so that each byte of the audio file contains only one bit from the secret message which is stored in rightmost bit; therefore, the differences will be non-existent in the case of similarity between the secret message bit and the audio bit to be replaced. Furthermore, the worst case will happen whenever there is a difference between the bit to be hidden and the one to be replaced; in this state, the value of the audio host byte will be increased by one. Accordingly, the resulting stego audio file will be closely identical to the host audio file as shown in Figure 6, which displays the data area representation for host and stego audio files when hiding SMF4 inside HWF4. The great similarity between host and stego audio files makes the process of detecting or doubting data to be sent almost non-existent or unavailable, thus undetectability as a measure was satisfied in the framework being built.

Table 1. Set of secret message files

Secret Message File	Size	Required Bytes
SMF 1	121 Bytes	1072 Bytes
SMF 2	621 Bytes	5072 Bytes
SMF 3	1246 Bytes	10072 Bytes
SMF 4	2496 Bytes	20072 Bytes

Table 2. Set of audio wave files

Host Wave File	Size	Available Bytes
HWF 1	563756 Bytes	70464 Bytes
HWF 2	1127468 Bytes	140928 Bytes
HWF 3	1691180 Bytes	211392 Bytes
HWF 4	2254892 Bytes	281856 Bytes

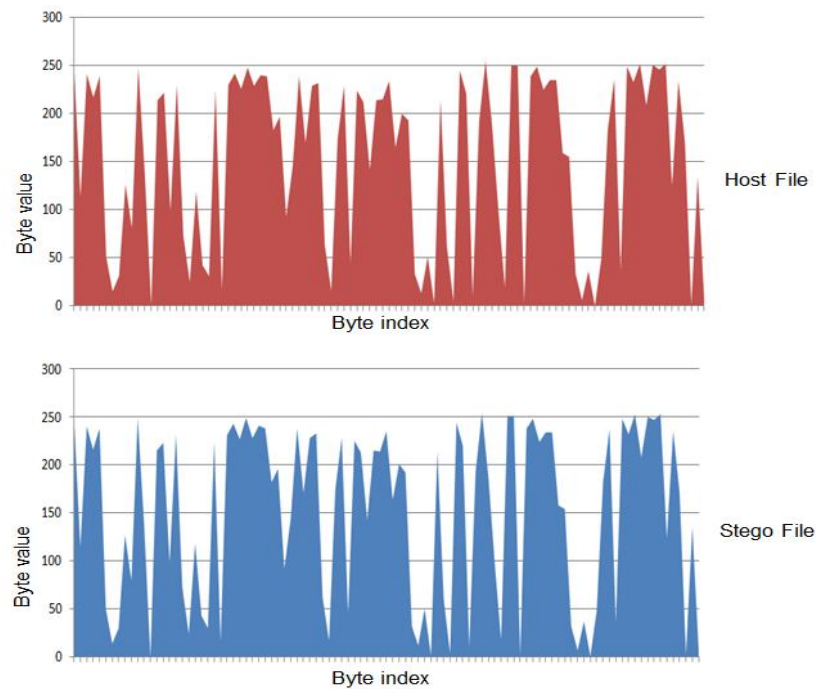


Figure 6. Data area representation for host and stego files

In addition, in case of comparing the resulting stego audio file with the host audio file, the human ear will not be able to distinguish sound modifications due to the small change that occurs between the stego and host audio files, as shown in Figures 7(a) and 7(b), which illustrate the waveforms of the host and stego files when hiding SMF4 inside HWF1 and HWF4. The few changes in the waveform of the stego audio file are achieved by the SA algorithm, which is considered the main part of the proposed model. The SA algorithm, as mentioned previously, works to determine the best path with the least modification rate within the host file, thus the imperceptibility (phonological hearing) as a measure was also satisfied in the constructed framework.

The built framework is considered secure because extracting a message requires knowing the keys that were used, whether for encryption or distribution, as well as the need to know the algorithms that were utilized for constructing secret message file, data encryption, best path determination, and embedding techniques inside the stego audio file. Moreover, there are needs to know the length of message data, type of hidden data, order of extracted data, and ways to decrypt this data. All of these challenges indicate and confirm the high security and strength of the proposed system.

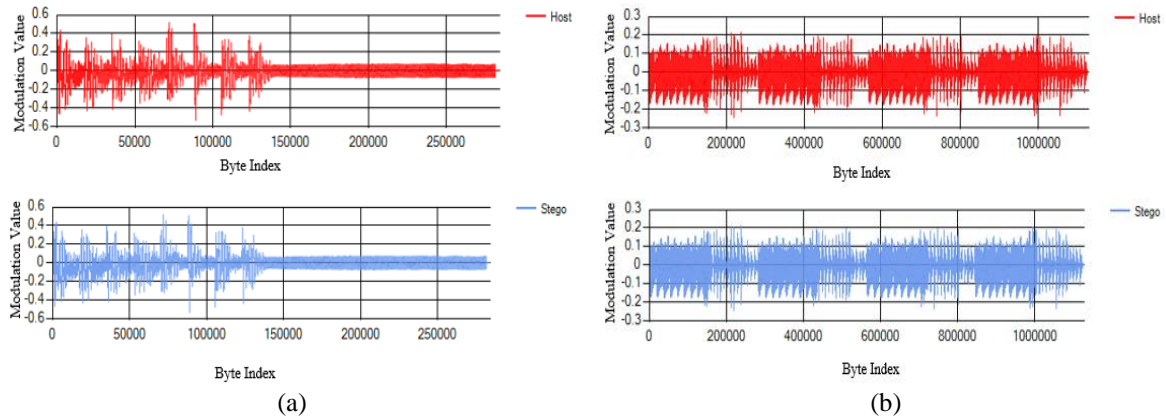


Figure 7. Wave form for host and stego files (a) wave form of SMF4 inside HWF1 and (b) wave form SMF4 inside HWF4

MSE and PSNR are used as detection quality measures [26]–[28] and employed to build a comparison between host and Steg audio files. Values of MSE and PSNR are computed by using (3) and (4), respectively. MSE calculates the average distance between the bytes of the host file (Bytex) and bytes of the stego file (Bytey) by reading each byte and its counterpart in the same location from both files, subtracting from each other, and then taking the square for the result of the subtraction. The sum of squares is divided by the total number of bytes (N), which should be equal in both files. The modification rate in the stego audio file is measured through the MSE, where the lower MSE resulting value is the lower modification rate, and therefore the lower MSE value is the more preferred one. In other words, the lower MSE value is the lower noise percentage in the stego audio file. On the contrary, the higher value of the PSNR is considered better than a lower one, due to the inverse relationship between MSE and PSNR.

$$MSE = \left( \sum_{i=1}^N (Byte_x(i) - Byte_y(i))^2 \right) / N \quad (3)$$

$$PSNR = 20 * \log_{10} \left( \frac{255}{\sqrt{MSE}} \right) \quad (4)$$

In order to identify the system efficiency that was built, values of MSE and PSNR were calculated for two systems. The first one used the SA algorithm to search for the best solution, and the second one worked sequentially (Seq) without using the SA algorithm, where the data to be hidden would be encrypted and embedded sequentially within the host audio file, starting from byte number (44). Files in Table 1 and Table 2 were used to build the comparison process between host audio and stego audio files that result from applying the two mentioned cases; the results of MSE and PSNR are shown in Table 3.

Results in Table 3 show an increase in the MSE values and decrease in the PSNR values in both cases whenever the messages (SMF1, SMF2, SMF3 and SMF4) were hidden in the same file. Increasing and decreasing in the values of MSE and PSNR result from differences in the size of embedded secret message files, where the smallest is SMF1 and the largest is SMF4. Figure 8 shows the extent of change in MSE and PSNR values when hiding the secret message files (SMF1, SMF2, SMF3, and SMF4) in the audio file HWF1 by using SA and Seq systems.

Figure 9 shows the inverse relationship between the size of the secret message file and the size of the host audio file. As the size of the secret message file increases and the size of the host audio file decreases, the MSE value increases, and vice versa. This can be proven by looking at the size of the audio file (HWF1, HWF2, HWF3, and HWF4) that were employed to hide the secret message SMF1, where the

MSE value was the highest possible with the HWF1 file, which has the smallest size, and the MSE value was the lowest with the HWF4 file, which has the largest size of the host audio files.

When comparing the SA system that was built with the Seq system, we find an advantage for the proposed system in terms of MSE, as it is clear in Figure 10, especially whenever the size of the message to be hidden is small compared to the size of the host audio file due to the availability of a huge number of sites as well as the availability of the opportunity to large distribution among these sites. The deduction appears clearly when hiding the SMF1 file using the SA method inside different host wave files.

Table 3. MSE and PSNR values for SA and sequential frameworks

Method	Used Host Wave File	Used Secret Message File	MSE	PSNR
Simulate Annealing Algorithm	HWF 1	SMF 1	0.00086	78.784
	HWF 1	SMF 2	0.004287	71.809
	HWF 1	SMF 3	0.008693	68.739
	HWF 1	SMF 4	0.017467	65.709
	HWF 2	SMF 1	0.000421	81.885
	HWF 2	SMF 2	0.002161	74.783
	HWF 2	SMF 3	0.004338	71.758
	HWF 2	SMF 4	0.008756	68.708
	HWF 3	SMF 1	0.000273	83.776
	HWF 3	SMF 2	0.001436	76.558
	HWF 3	SMF 3	0.00287	73.552
	HWF 3	SMF 4	0.005808	70.491
	HWF 4	SMF 1	0.000211	84.886
	HWF 4	SMF 2	0.001083	77.786
	HWF 4	SMF 3	0.002164	74.779
	Sequential Algorithm	HWF 4	SMF 4	0.004373
HWF 1		SMF 1	0.000899	78.592
HWF 1		SMF 2	0.004454	71.643
HWF 1		SMF 3	0.008949	68.613
HWF 1		SMF 4	0.01772	65.646
HWF 2		SMF 1	0.000484	81.280
HWF 2		SMF 2	0.002258	74.593
HWF 2		SMF 3	0.004433	71.664
HWF 2		SMF 4	0.008895	68.639
HWF 3		SMF 1	0.000305	83.295
HWF 3		SMF 2	0.001482	76.421
HWF 3		SMF 3	0.002941	73.446
HWF 3		SMF 4	0.005878	70.438
HWF 4		SMF 1	0.000213	84.850
HWF 4		SMF 2	0.001107	77.690
HWF 4		SMF 3	0.002215	74.677
HWF 4	SMF 4	0.004457	71.640	

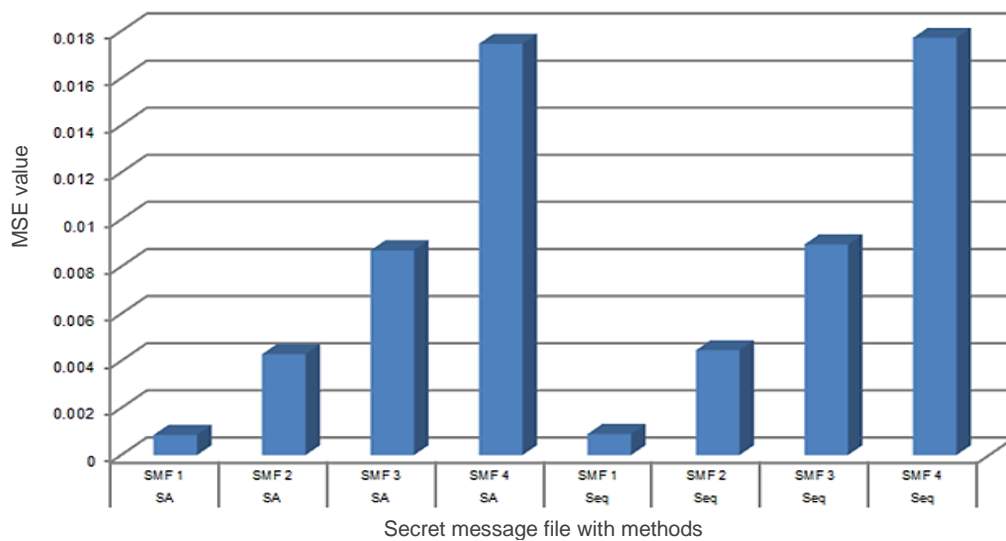


Figure 8. MSE values for hiding secret message files in HWF1

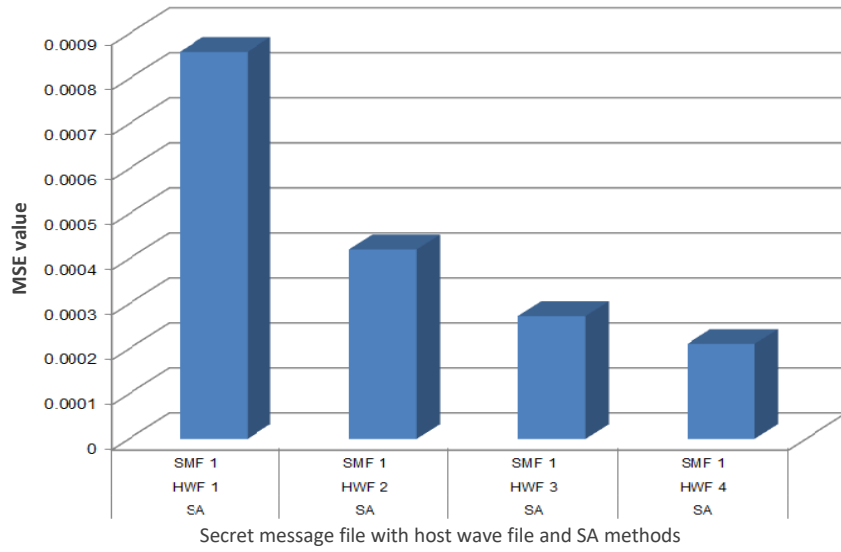


Figure 9. MSE value for hiding SMF1 in host wave files

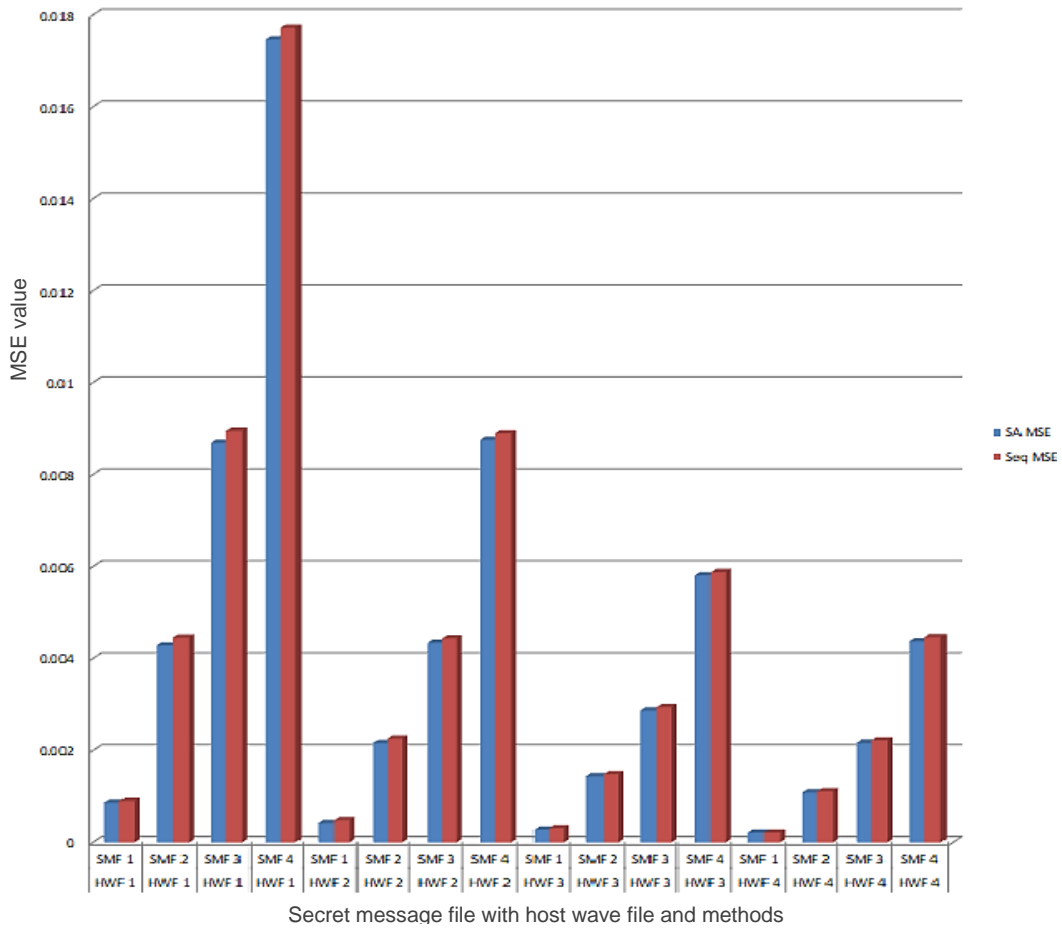


Figure 10. MSE value for hiding via SA and Seq systems

The built system deals with data at the byte level; therefore, before starting the hiding process, it compares the data to be hidden with the hosting data. Accordingly, the similarity between secret and host data in terms of formats may lead to a reduction in MSE value. For example, the MSE value is small if audio

data is hidden within an audio file due to the similarity or closeness between the values at the binary level, but due to the large size of the secret audio files, text or image files are more preferred to be hidden and sent.

By comparing the average MSE value for SA and Seq systems, which were (0.0041) and (0.0042), respectively, we find that there is an improvement, but not significant, using the SA algorithm. This is also clear when calculating the average PSNR for each of them, which are 74.73 for the SA system and 74.57 for the Seq system. The matter is attributed to the construction method of the SA path, which is based on sequential selection of sites as well as possibly redundancy in sites.

The host audio file can hide a binary message that is eight times smaller than its size because each byte of the secret message requires eight bytes from the host audio file. Whenever a binary secret message file with a size of (70448) bytes was hidden inside the HWF1, it was found that the values of MSE and PSNR were (0.061713 and 60.23), respectively, taking into account that the resulting file still works and the sound can be heard without distortion. Based on the above, the system that was built succeeded in capacity evaluation.

## 5. CONCLUSION

The idea of merging artificial intelligence, cryptography, and steganography in one framework is considered a development and challenge from a security standpoint, as well as an innovative one. Artificial intelligence improves the steganography process in terms of time, space, and modification rate. Moreover, it makes the process of data extraction from attackers more complex due to variability in used methods and their arguments. The presented method is characterized by strength against unauthorized users and effectiveness in terms of performance and changes to the audio file due to the process of combination between different sciences that were mentioned formerly. Results and analysis showed that the constructed system satisfied the main requirements for steganography, which are represented by robustness, detection, imperceptibility (phonological hearing), security, PSNR, MSE, and capacity as discussed previously. The main problem of the most existing steganography systems is the robustness against data modification, whether in content, size, or formats, and currently there is no solution for that. Replacing the SA algorithm with another artificial intelligence algorithm, such as IWD may improve the performance more, and this will be a future modification for the current system. The results of the new suggested system will be extracted and compared with the current ones in order to ensure whether there is an improvement or not.




## REFERENCES

- [1] M. Bansal and R. Ratan, "Designing a novel technique for multi-level security system for digital data by combined DSSS audio steganography & random permutation cryptography," *Tuijin Jishu/Journal of Propulsion Technology*, vol. 44, no. 4, 2023, doi: 10.52783/tjpt.v44.i4.980.
- [2] M. J. Bawaneh and A. A. Obeidat, "A secure robust gray scale image steganography using image segmentation," *Journal of Information Security*, vol. 07, no. 03, pp. 152–164, 2016, doi: 10.4236/jis.2016.73011.
- [3] N. M. Al-Saidie and T. A. Kadhim, "Using fractals in information hiding," *Engineering and Technology Journal*, vol. 27, no. 16, pp. 3093–3111, Dec. 2009, doi: 10.30684/etj.27.16.15.
- [4] M. C. Kasapbasi, "A new chaotic image steganography technique based on Huffman compression of Turkish texts and fractal encryption with post-quantum security," *IEEE Access*, vol. 7, pp. 148495–148510, 2019, doi: 10.1109/access.2019.2946807.
- [5] J. Bahl and R. Ramakishore, "LSB technique and its variations used in audio steganography: a survey," *International Journal of Engineering Research & Technology*, vol. 2, no. 4, pp. 2327–2332, 2013.
- [6] H. Dutta, R. K. Das, S. Nandi, and S. R. M. Prasanna, "An overview of digital audio steganography," *IETE Technical Review*, vol. 37, no. 6, pp. 632–650, Dec. 2019, doi: 10.1080/02564602.2019.1699454.
- [7] M. J. Bawaneh, "An adaptive virtual gray scale image steganography using simulated annealing," *International Journal of Computer Science and Information Security*, vol. 14, no. 9, pp. 612–621, 2016.
- [8] K. Manjunath, G. N. K. Ramaiah, and M. N. G. Prasad, "Optimal secure XOR encryption with dynamic key for effective audio steganography," *International Journal of Speech Technology*, vol. 26, no. 3, pp. 589–598, Nov. 2021, doi: 10.1007/s10772-021-09945-6.
- [9] M. A. Nasr *et al.*, "A robust technique for steganography of enhanced audio signals," in *2023 3rd International Conference on Electronic Engineering (ICEEM)*, Oct. 2023, pp. 1–6. doi: 10.1109/iceem58740.2023.10319573.
- [10] M. H. Mohammed, "A new approach to hide texts into images and audio files using steganography and cryptography techniques," *International Journal of Advances in Applied Sciences (IJAAS)*, vol. 11, no. 4, Dec. 2022, doi: 10.11591/ijaas.v11.i4.pp312-323.
- [11] J. Wu, B. Chen, W. Luo, and Y. Fang, "Audio steganography based on iterative adversarial attacks against convolutional neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2282–2294, 2020, doi: 10.1109/tifs.2019.2963764.
- [12] S. Jiang, D. Ye, J. Huang, Y. Shang, and Z. Zheng, "SmartSteganography: light-weight generative audio steganography model for smart embedding application," *Journal of Network and Computer Applications*, vol. 165, Sep. 2020, doi: 10.1016/j.jnca.2020.102689.
- [13] X. Yi, K. Yang, X. Zhao, Y. Wang, and H. Yu, "AHCM: adaptive Huffman code mapping for audio steganography based on psychoacoustic model," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 8, pp. 2217–2231, Aug. 2019, doi: 10.1109/tifs.2019.2895200.
- [14] R. Wazirali, W. Alasmay, M. M. E. A. Mahmoud, and A. Alhindi, "An optimized steganography hiding capacity and imperceptibly using genetic algorithms," *IEEE Access*, vol. 7, pp. 133496–133508, 2019, doi: 10.1109/access.2019.2941440.
- [15] M. J. Bawaneh, "A preferential virtual gray scale image steganography using intelligent water drop," *International Journal of Computer Science and Information Security*, vol. 14, no. 11, pp. 538–546, 2016.




- [16] A. H. Ali, L. E. George, A. A. Zaidan, and M. R. Mokhtar, "High capacity, transparent and secure audio steganography model based on fractal coding and chaotic map in temporal domain," *Multimedia Tools and Applications*, vol. 77, no. 23, pp. 31487–31516, Jun. 2018, doi: 10.1007/s11042-018-6213-0.
- [17] M. J. Bawaneh, "A novel approach for image steganography using LCG," *International Journal of Computer Applications*, vol. 102, no. 10, pp. 34–38, Sep. 2014, doi: 10.5120/17853-8826.
- [18] S. S. Bharti, M. Gupta, and S. Agarwal, "A novel approach for audio steganography by processing of amplitudes and signs of secret audio separately," *Multimedia Tools and Applications*, vol. 78, no. 16, pp. 23179–23201, Apr. 2019, doi: 10.1007/s11042-019-7630-4.
- [19] S. Kanaujiya, B. Chaudhari, V. Kannaujiya, and P. Madhura Vyawahare, "Hiding data into audio using steganographic methods," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 10, pp. 2320–2882, 2022.
- [20] S. Krishnan and M. S. Abdullah, "Enhanced security audio steganography by using," *Journal of Advanced Research in Computing and Applications*, vol. 2, no. 1, pp. 39–54, 2016.
- [21] R. Shanthakumari, E. M. R. Devi, R. Rajadevi, and B. Bharaneeshwar, "Information hiding in audio steganography using LSB matching revisited," *Journal of Physics: Conference Series*, vol. 1911, no. 1, p. 12027, May 2021, doi: 10.1088/1742-6596/1911/1/012027.
- [22] G. Bhatnagar, S. Mehta, and S. Mitra, "The WAV file format," in *Introduction to Multimedia Systems*, Elsevier, 2004, pp. 55–60. doi: 10.1016/b978-012500453-4/50011-0.
- [23] W. Stallings, "Cryptography and network security: principles and practices," *Cryptography and Network Security*, p. 592, 2005.
- [24] S. Hemalatha, U. D. Acharya, and A. Renuka, "Wavelet transform based steganography technique to hide audio signals in image," *Procedia Computer Science*, vol. 47, pp. 272–281, 2015, doi: 10.1016/j.procs.2015.03.207.
- [25] K. Dasgupta, J. K. Mandal, and P. Dutta, "Hash based least significant bit technique for video steganography (HLSB)," *International Journal of Security, Privacy and Trust Management (IJSPTM)*, vol. 1, no. 2, 2012, doi: 10.5121/ijspmt.2012.2201.
- [26] M. J. Bawaneh, A. A. Obeidat, O. M. Al-Hazaimeh, M. M. Al-Nawashi, and A. R. Shorman, "An adaptive fractal image steganography using Mandelbrot and linear congruent generator," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 15, no. 3, pp. 99–112, Dec. 2023, doi: 10.17762/ijcnis.v15i3.6240.
- [27] M. Kaur and A. Kaur, "Improved security mechanism of text in video using steganographic technique: A review," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 7782, no. 5, pp. 44–51, 2014.
- [28] P. Shinde and T. B. Rehman, "A novel video steganography technique," *International Journal of Advanced Research in Computer Science and Software Engineering Research*, vol. 5, no. 12, pp. 676–684, 2015.

## BIOGRAPHIES OF AUTHORS






**Atef Ahmed Obeidat**    received the B.Sc. degree in computer science from Yarmouk University, Irbid, Jordan, in 1991, the M.Sc. degree in computer science from the University of Jordan, Amman, Jordan, in 2001, and the Ph.D. degree in computer science from Novosibirsk State Technical University, Novosibirsk, Russia, in 2006. He has been an associate professor of computer science at Al Balqa Applied University/Al-Huson university college since 2018. He is currently the head of the Department of Information Technology, Al-Huson university college in Irbid, Jordan. He has authored or coauthored more than 40 refereed journal and conference papers. His research interests include the applications of artificial intelligence and the security of computers and networks. He can be contacted at email: Dr.atefob@bau.edu.jo.






**Mohammed Jazi Bawaneh**    received the B.S. degree in computer science from Yarmouk University in 2001 and M.S. degrees in computer science from Yarmouk University in 2004. He received the PhD degree in computer information systems from the Arab Academy of Banking and Financial Science in 2010. He can be contacted at email: dr\_mjab@bau.edu.jo.






**Sawsan Yousef Abu Shqair**    received the bachelor's and master's degrees in computer science from Yarmouk University, Jordan, in 2001 and 2004 respectively; she has been working at Al Balqa Applied University in Jordan in the Department of Information Technology since 2004 until now. Her research interests include artificial intelligence and communication. She can be contacted at email: sawsan.abushuqair@bau.edu.jo.





**Hamdi A. Al-Omari**    received the B.Sc. degree in computer science from Yarmouk University, Jordan, in 2001 and the M.S. degree in computer information systems from Yarmouk University in 2005. Currently, he is a computer science instructor at the Department of Information Technology, Huson University College, Al-Balqa Applied University (BAU), Jordan. He was head of the computer center at Al-Huson university college from 2016 to 2022. His research interests include computer security, steganography, big data, database security, data mining, and management information systems. He can be contacted at email: Hamdi.omari@bau.edu.jo.



**Emad Fawzi Alshalabi**    received the B.S. degree in information technology, management information systems from Philadelphia University, Jordan in, 2004, and the MSc in computer information systems from Yarmouk University (YU), Jordan, in 2009. His research interests include computer vision, information retrieval, and embedded systems. He can be contacted at email: emad@bau.edu.jo.