HIDE TEXT INFORMATION IN .WAV AUDIO USING LSB

HIDE TEXT INFORMATION IN .WAV AUDIO USING LSB

Wendy Carranza Díaz [1], María Concepción Villatoro-Cruz [2]
Guillermina Jiménez Rasgado[3], José Aurelio Ramírez González[4], Pablo Francisco Vivas Torres[5]https://doi.org/10.61117/ipsumtec.v9i1.461

[1]National National de Mexico /Institute Technological of Minatitlán, wendy.cd@minatitlan.tecnm.mx, https://orcid.org/0009-0009-4655-2816
[2] National National of Mexico /Institute Technological of Minatitlán, maría.vc@minatitlan.tecnm.mx, https://orcid.org/0000-0002-8986-6219.
[3] National Technological Institute of Mexico /Minatitlán Institute of Technology,guillermina.jr@minatitlan.tecnm.mx , https://orcid.org/0000-0003-0163-0525,
[4] National Technological Institute of Mexico / Acayucan Higher Technological Institute,aurelio.rg@acayucan.tecnm.mx , https://orcid.org/0009-0008-8986-0774
(5) National Technological National de Mexico /Higher Technological of Minatitlán, pablo.vt@minatitlan.tecnm.mx, https://orcid.org/0009-0001-5920-3668

Abstract -- This article describes the development of a Python library for hiding information in audio files using the Least Significant Bit (LSB) steganography technique. The objective was to design a reusable package capable of embedding and extracting PDF files within WAV files without affecting their perceptible quality.
The research followed an experimental approach divided into five stages: review of the state of the art, definition of the LSB algorithm, integration of symmetric encryption using AES-256, code implementation, and validation of results. The PyDub libraries were used for audio manipulation, NumPy for sample handling, and Fernet for encryption. The process consisted of first encrypting the PDF file, modifying the least significant bits of the audio samples to insert the data, and then reconstructing the WAV file. Extraction was performed in reverse, by retrieving and decrypting the embedded message.
The results demonstrated that the LSB method combined with AES-256 encryption allows information to be hidden while maintaining the integrity of the original file and without perceptibly altering the sound. In all tests, both locally and on the PythonAnywhere platform, the PDF file was recovered in its entirety, and the resulting audio retained its original quality.
It is concluded that audio steganography using Python is an effective alternative for protecting sensitive information, as it combines the security of encryption with the discretion of digital concealment. However, the method's vulnerability to audio compression or manipulation is acknowledged. Future work will focus on optimizing storage capacity and exploring more robust steganography techniques.

Keywords—AES Encryption, Steganography, LSB, Python, Digital Security.

Abstract -- This article describes the development of a Python library for hiding information in audio files using the *Least Significant Bit* (LSB) steganographic method. The objective was to design a reusable package capable of embedding and extracting PDF files within WAV audio files without perceptibly affecting their quality.
The research followed an experimental approach divided into five stages: review of the state of the art, definition of the LSB algorithm, integration of symmetric encryption with AES-256, code implementation, and validation of results. The PyDub library was used for audio manipulation, NumPy for sample processing, and Fernet for encryption. The process consisted of encrypting the PDF file, modifying the least significant bits of the audio samples to insert the data, and reconstructing the WAV file. Extraction was performed in reverse, by recovering and decrypting the embedded message.
The results demonstrated that the LSB method combined with AES-256 encryption effectively hides information while maintaining the integrity of the original file and preserving sound quality. In all tests, both locally and on the PythonAnywhere platform, the PDF file was fully recovered, and the audio remained perceptually identical to the original.
It is concluded that audio steganography using Python is an effective alternative for protecting sensitive information by combining encryption security with the discretion of digital hiding. However, the method's vulnerability to compression or audio manipulation is acknowledged. As future work, it is proposed to optimize data capacity and explore more robust

steganographic techniques.

Keywords **–** AES Encryption, Steganography, LSB, Python, Digital Security.

INTRODUCTION

In today's digital age, information security has become a top priority due to the rise in cyberattacks, data interception, and the vulnerability of communication systems.

Today, steganography has applications in industry, such as hiding intellectual property information in audio and image files [1], as well as in military applications where secret messages are used.

In the current state of the art, it is worth noting that traditional security techniques, such as cryptography and steganography, protect the content of the information but do not prevent third parties from detecting its existence.

To begin with, cryptography is defined as "the process of hiding or encoding information using algorithms and keys, so that the data can only be interpreted by intended recipients. This includes techniques such as encryption and decryption, which protect information in transit or at rest from unauthorized access"[2]. On the other hand, "Steganography consists of a method of embedding, concealing, or encoding in which information is hidden within the original file, which we will also refer to as the carrier"[3]. Furthermore, in the field of digital audio, steganography has gained prominence due to its ability to embed messages or files within audio signals without noticeably altering their quality**.** In particular, the Least Significant Bit (LSB) method has established itself as one of the most widely used strategies due to its simplicity, storage capacity, and computational efficiency**.** Similarly, variations of the classic LSB method and their limitations in terms of data capacity and robustness have been documented [4]

As stated in a recent study, "The least significant bit (LSB) method was used as a steganographic transformation algorithm. The method's low level of reliability can be improved by introducing a cryptographic layer."[5]. This approach underscores the need to combine steganography with encryption techniques, such as AES-256, to ensure the confidentiality of the hidden data. However, most existing implementations lack built-in encryption mechanisms**,** making them vulnerable if the hidden data is discovered.

Given this limitation, steganography emerges as a complementary tool that allows information to be hidden within other digital media so that its presence goes unnoticed by an unauthorized observer. Audio steganography has gained prominence as a technique for hiding in sound files without compromising their perceptible quality.

This paper addresses this limitation by developing a Python library that integrates the LSB (Least Significant Bit) steganography technique with AES-256 symmetric encryption**,** allowing PDF files to be hidden within WAV audio files**.** This combination provides an additional layer of security, ensuring that the message remains inaccessible without the correct key, even if it is detected. This article presents a practical implementation in Python that combines the LSB (Least Significant Bit) method with AES encryption to hide PDF files within WAV files. The solution leverages libraries such as NumPy and PyDub to manipulate audio samples, ensuring that the hidden data remains inaccessible without the correct password.

In addition to describing the code design, the results and recommendations are documented.

The overall objective of the research is to design and implement a reusable Python library for hiding and retrieving information in audio files using the LSB method with AES encryption, accompanied by technical documentation and experimental validation. The rationale for this work lies in the need for open, secure, and accessible solutions for hiding sensitive information, applicable in areas such as digital security, covert communication, and data protection. Furthermore, the use of Python as the implementation language promotes learning and the replicability of the method in educational and research contexts.

THEORETICAL FRAMEWORK

Steganography:

"Steganography is the art of hiding information in a carrier medium in such a way that the presence of that information remains undetected" [6]

Similarly, Sion defines it as follows: "Steganography (from the Greek steganos, 'hidden') is a term denoting mechanisms for hiding information within a 'carrier medium' so that, generally, only the intended recipient is aware of its existence and is able to retrieve it from that medium." [7]

Therefore, it can be said that steganography is the science of hiding information within other digital media—such as images, audio, or video—in a way that its presence goes unnoticed. Unlike cryptography, which protects the content of the message, steganography seeks to conceal its very existence.

Regarding the importance of steganography, and to highlight the difference from cryptography, it can be said that: "steganography differs from cryptography in that its objective is not to protect the content of the message, but to conceal its very existence." [8]

Furthermore, they explore how the fusion of steganography and cryptography in audio, using enhanced LSB, strengthens the protection of sensitive data. [9]

Likewise, [10] introduces an approach that employs genetic algorithms to optimize the hiding capacity and signal quality in audio steganography.
Thus, in a digital audio file (such as WAV), amplitude samples are stored in PCM (Pulse Code Modulation) format. Each sample is a quantized numerical value (typically 16 bits), where the least significant bit (LSB) has a minimal impact on human perception of sound.
LSB (Least Significant Bit)
The LSB method involves replacing the least significant bits of the audio samples with bits from the secret message. If only 1 or 2 LSBs are modified, the resulting distortion is almost imperceptible to the human ear. In this regard, the LSB method in audio can be described as: "Modifying the least significant bits (LSB) in audio signals to hide data with minimal distortion, provided that the modification is limited to 1–2 bits per sample" [11].

Example:
• Original sample (16 bits): 11011010 10101101
• Bit to be hidden: 1
• Modified sample: 11011010 1010110**1** (only the last bit changes)

Components of the Proposed System:
The scheme implemented in the analyzed code consists of three main stages:

Message Encryption (Additional Security):
For message encryption, the effectiveness of combining audio steganography with AES and LSB encryption to enhance security in military communications is noteworthy [12], which is truly interesting, as it provides a multi-layered security approach. Therefore, based on this source, before hiding the data, symmetric encryption (AES-256 via Fernet) is applied using a key derived from a password via SHA-256. This ensures that, even if the message is detected, it cannot be read without the correct key.
Furthermore, an advanced technique is proposed that uses chaotic maps to encrypt images before embedding them in audio signals, thereby enhancing the security of the process. [13]

Steganographic Insertion (LSB)
- Input: Original audio (WAV) + Encrypted data.
- Process:
  1. The audio is converted into an array of samples (using pydub and numpy).
  2. The *n* LSBs of each sample are replaced with the bits of the encrypted message.
  3. The WAV file is reconstructed using the modified samples.

Extraction and Decryption
- Input: Steganographic audio + Password.
- Process:
  1. The LSBs are extracted from the audio samples.
  2. The bytes of the encrypted message are reconstructed.
  3. The data is decrypted using the provided password.

Advantages and Limitations

Advantages:
- Low perceptibility: Modifying 1–2 LSBs does not significantly affect the audio quality.
- High capacity: Depending on the number of bits modified, KBs of data can be hidden in seconds of audio. Enhanced security: Pre-encryption prevents access to the message without the key.

Limitations:
Vulnerability: The LSB method is sensitive to compression, reprocessing, or noise. Detection possible: Advanced statistical analysis can reveal the presence of hidden data.

Format Dependency: Works best on uncompressed WAV files (PCM).

Applications in Information Technology:
• Covert communications: Transmission of secret messages in audio streams.
• Digital watermarking: Inclusion of watermarks for copyright protection.
• Offensive/defensive security: Use in penetration testing or the protection of sensitive data.

Definition of a Library (in the context of programming):
A library is a set of predefined functions, methods, and classes that allow you to:
- Reuse code without having to write it from scratch.
- Standardize processes (e.g., encryption, file handling, signal processing).
- Optimize performance, since many libraries are highly optimized (e.g., NumPy for fast mathematical operations).

DEVELOPMENT
The steganographic process is based on the triad consisting of the secret message (information to be hidden), the cover object (original medium), and the stego object (result of the process containing the embedded information) [14]
Under this approach, the methodology applied for the development of this work was structured as follows:

Methodology:

1. Search for relevant information on the web
2. Definition and analysis of the LSB (Least Significant Bit) algorithm
3. Selection of source files, including audio files and PDF documents
4. Organization and placement of all files within the structure of the developed library
5. Verification of system functionality and debugging of detected errors
6. Checking and validating the results obtained

Search for relevant information on the web:
First, an exhaustive investigation of existing examples of steganography was conducted through a review of the state of the art in steganography. It is important to first understand the LSB method. Therefore, the focus was on the LSB method applied to audio files. Scientific articles, books, and code repositories were consulted to understand existing implementations and their limitations.
Recent research has explored combining steganography with chaotic cryptography, which increases the hiding capacity and resistance to detection techniques. [15]

Definition and Analysis of the LSB (Least Significant Bit) Algorithm:
The solution uses LSB (Least Significant Bit), which is a classic steganography method. This technique modifies the least significant bits of the audio samples. However, these changes are nearly imperceptible to the human ear. An algorithm based on the LSB method was designed to modify the least significant bits of the audio samples, ensuring that the changes were imperceptible to the human ear. This allows binary data to be embedded within the audio file.

Selection of source files, including audio files and PDF documents:
An audio file and a PDF file were searched for and selected to serve as the carrier and secret data, respectively.

In addition, Python was chosen as the programming language. The development IDE is Visual Studio Code, and the compiler from pythonanywhere.com was used for testing.
Python is a very modern programming language. It is a high-level, interpreted, general-purpose language, known for being clear, readable, and easy to learn.

In addition, the PyDub library was chosen for audio manipulation and NumPy for efficient sample processing.
Pythonanywhere is a platform—a cloud service that allows you to host, run, and develop Python applications directly from a web browser. It is especially useful for beginners, educators, and developers who want to deploy web projects without having to configure complex servers.
Organization and location of all files within the structure of the developed library:
All selected files, as well as the necessary code and files, were placed in a folder named after the developed library.

It is important to note that all selected files, as well as the code, should be saved in the same location as the executable file. The generated files can also be saved in the same location.

Verification of system functionality and debugging of detected errors:

- Encryption: The data in the PDF file was encrypted before being embedded in the audio. Generally speaking, to conceal the information, the approach involved encrypting the PDF with a password, converting the audio to digital samples, hiding the data by modifying the LSBs, and saving the audio file as audio_esteganografiado.wav and the output file as mensaje_extraido.pdf
- Insertion: The audio samples were modified to include the bits of the encrypted message.
- Extraction: The LSBs from the audio were retrieved and decrypted to reconstruct the original PDF. The files used for this process were mensaje.pdf and original.wav.

Verification and validation of the results: After hiding the data, the extraction process continues: the audio samples are read, the least significant bits are extracted, the password is used to decrypt the data, and the original PDF is reconstructed.
It is recommended that the files maintain the expected format and match the file extension handled by the code: .wav or, failing that, .pdf
In the case of the library, it is proposed that it be organized as follows:

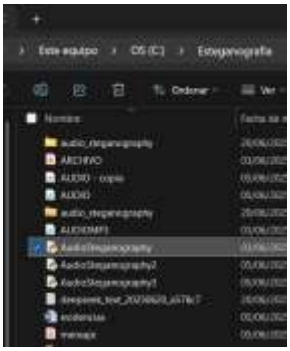

Figure 1. *Structure of the audiosteganography library*.
Source: Author's *own work* (2026).

Steganography (LSB): Implemented manually, but NumPy and PyDub facilitate the handling of audio samples.
Encryption: Fully delegated to Fernet (from the cryptography library).
File handling: PyDub is responsible for loading/saving the audio, while Python handles the PDF using the native `open()` function.
This code implements a steganography system that hides PDF files within WAV audio files using encryption and bit manipulation techniques.
The focus of this work is on using the least significant bit (LSB) technique applied to digital audio signals in WAV format, due to its simplicity and efficiency in hiding data.

To validate the results obtained, several tests were conducted:

- ✓ Test 1:
  Successful hiding and extraction on pythonanywhere.com, verifying the integrity of the PDF.
- ✓ Test 2:
  Local execution in CMD and confirmation of the correct generation of the output files.
- ✓ Test 3:
  Audio quality analysis, where no differences perceptible to the human ear were detected between the original and the modified version.

Validation confirmed that the extracted PDF file was identical to the original and that the audio retained its quality.

EXAMPLE CODE

The implemented code includes functions for encrypting, hiding, extracting, and decrypting data. The **AudioSteganography3** class was used to encapsulate the logic, making it easier to reuse. Tests showed that the system is capable of handling files up to several KB without compromising audio quality.

The following code was used to test the library:

```
import numpy as np
from pydub import AudioSegment
from cryptography.fernet import Fernet from hashlib import
sha256
from base64 import urlsafe_b64encode
class AudioSteganography3:
        def init(self): pass
        def _generate_key(self, password: str) -> bytes:
        return sha256(password.encode()).digest()

        def _encrypt_data(self, data: bytes, password: str) ->
        bytes: key =
        urlsafe_b64encode(self._generate_key(password))
        fernet = Fernet(key)
        return fernet.encrypt(data)
def _decrypt_data(self, encrypted_data: bytes, password: str) ->
bytes:
key = urlsafe_b64encode(self._generate_key(password))
fernet = Fernet(key)
return fernet.decrypt(encrypted_data)
def _audio_to_samples(self, audio_path: str) -> np.ndarray:
audio = AudioSegment.from_file(audio_path)
if audio.channels > 1:
        audio = audio.set_channels(1)
return
np.array(audio.get_array_of_samples()).astype(np.int16)

def _samples_to_audio(self, samples: np.ndarray, output_path:
str):
audio = AudioSegment(
        samples.tobytes(),
        frame_rate=44100,
        sample_width=2,
        channels=1
)
audio.export(output_path, format="wav")
def _hide_data_in_audio(self, samples: np.ndarray, data: bytes,
lsb_bits: int = 2) -> np.ndarray:
        samples = samples.astype(np.int16)
        data_bits = ''.join(format(byte, '08b') for byte in data)
+ '00000000' # EOF marker

        max_bits = len(samples) * lsb_bits
        if len(data_bits) > max_bits:
          raise ValueError(f"Audio is too short for the data.
It requires {len(data_bits)} bits, but only
{max_bits} available.")

        modified_samples = samples.copy()
        bit_idx = 0

        for i in range(len(samples)):
          if bit_idx >= len(data_bits):
            break

        sample = int(samples[i])
        cleared_sample = sample & (~((1 << lsb_bits) - 1))
        bits_to_embed = data_bits[bit_idx:bit_idx +
        lsb_bits].ljust(lsb_bits, '0')
         bits_value = int(bits_to_embed, 2)
        new_sample = cleared_sample | bits_value

        new_sample = max(-32768, min(32767,
        new_sample))
        modified_samples[i] = np.int16(new_sample)

        bit_idx += lsb_bits
return modified_samples
def _extract_data_from_audio(self, samples: np.ndarray,
lsb_bits: int = 2) -> bytes:
        bits = ''
        for sample in samples:

        bits += format(sample & ((1 << lsb_bits) - 1),
f'0{lsb_bits}b')

byte_chunks = [bits[i:i+8] for i in range(0, len(bits), 8)]

data_bytes = bytearray()
for chunk in byte_chunks:
```

```
            if chunk == '00000000': # EOF marker
              break
            data_bytes.append(int(chunk, 2))
    return bytes(data_bytes)
    def hide_pdf_in_audio(self, pdf_path: str, audio_path: str,
    output_path: str, password: str, lsb_bits: int = 2):
            with open(pdf_path, 'rb') as f:
                 pdf_data = f.read()
            encrypted_data = self._encrypt_data(pdf_data,
            password)
            samples = self._audio_to_samples(audio_path)
            stego_samples = self._hide_data_in_audio(samples,
            encrypted_data,
            lsb_bits)
            self._samples_to_audio(stego_samples, output_path)
            print(f" PDF hidden in {output_path} with {lsb_bits}
            LSB bits.")
            def extract_pdf_from_audio(self, audio_path: str,
    output_pdf_path: str, password: str, lsb_bits: int = 2):
            samples = self._audio_to_samples(audio_path)
            encrypted_data =
            self._extract_data_from_audio(samples, lsb_bits)
            try:
                decrypted_data =
                self._decrypt_data(encrypted_data, password)
                except Exception as e:
                        raise ValueError("Error decrypting the
                        data: Is the password
                        correct?") from e
            with open(output_pdf_path, 'wb') as f:
            f.write(decrypted_data)
            print(f" PDF extracted successfully to:
            {output_pdf_path}")
    # ========================
    # Test code
    # ========================
    if_ name_ == "_ main_ ":
            stego = AudioSteganography3()
    # Change these paths depending on where your files are located
    pdf_path
    = "message.pdf"
    audio_path = "original.wav"
    output_stego_audio = "steganized_audio.wav"
    extracted_pdf_path = "extracted_message.pdf" password =
    "mypassword123"
    lsb_bits = 2
    # Hide PDF in audio
    stego.hide_pdf_in_audio(pdf_path, audio_path,
    output_stego_audio, password, lsb_bits)
    # Extract PDF from steganographic audio
    stego.extract_pdf_from_audio(output_stego_audio,
    extracted_pdf_path, password, lsb_bits)
```

First, the PythonAnywhere online platform was used, where the source files with .wav and .pdf extensions were uploaded, yielding good results.

Execution:

Figure 2. *Testing the AudioSteganography3.py code on PythonAnywhere*.
Source: Author's *own work* (2026).

GENERATED FILES:

- extracted_message.pdf
- steganized_audio.wav

Next, Figure 3 shows the list of files uploaded to the PythonAnywhere platform, which is open to the general public and does not require any software to be installed on the computer beforehand to test the code. Its URL is the following: https://www.pythonanywhere.com/ PythonAnywhere is a cloud-based platform (an online IDE and web hosting service), eliminating the need to install Python and configure it locally.

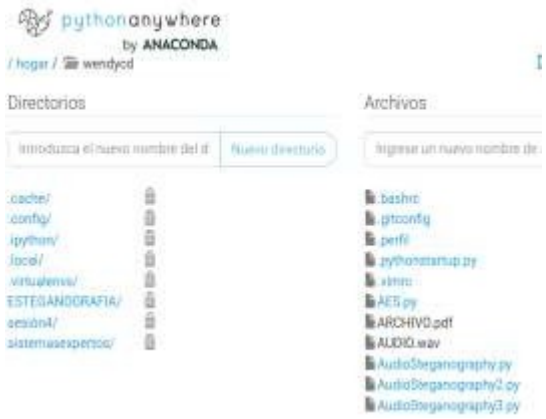

Figure 3. *Repository of Code on pythonanywhere.com*.
Source: Author's *own work* (2026).

When opening the file: mensaje_extraido.pdf, you will see what is shown in Figure 4.

Figure 4. *Test message file hidden in the audio*.
Source: Author's *own work* (2026).

Subsequently, the test was performed using cmd on the computer, as follows:

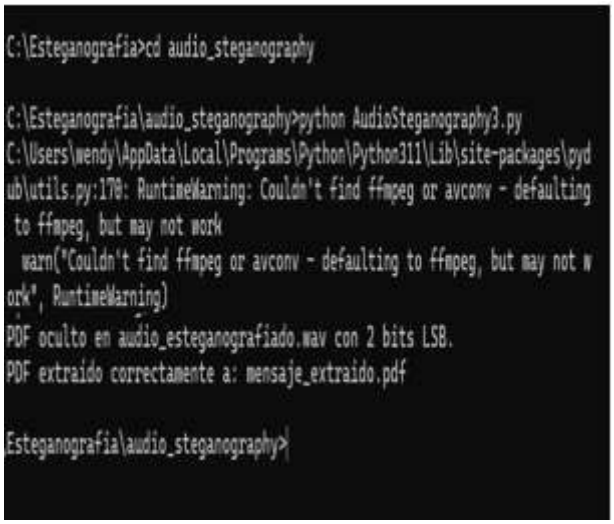

Figure 5. *Running the test using the audio_steganography library*.
Source: Author's *own work* (2026).

The result was successful, and to verify this, the following image is shown, which displays the files generated using the library.

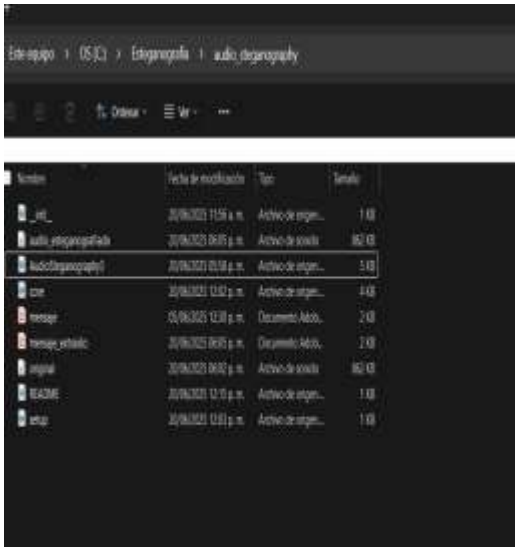

Figure 6. *Correctly generated output files*.
Source: Author's *own work* (2026).

DISCUSSION AND ANALYSIS OF RESULTS

The results obtained in this research demonstrate that the implementation of the LSB-based steganography technique combined with AES-256 encryption is functional and effective for hiding information in WAV-format audio files. In all tests conducted, both in a local environment and on the PythonAnywhere platform, the PDF file was successfully embedded and subsequently retrieved without any loss of information, confirming the integrity of the process.

The objective of this research was achieved by designing and implementing a reusable Python library to hide and retrieve information in audio files using the LSB method with AES encryption. In terms of audio quality, no perceptible differences were detected between the original file and the steganographic file. This behavior is consistent with the findings reported by Sánchez Rinza et al. [16], who demonstrate that using the LSB method in audio allows information to be hidden while maintaining low signal distortion. In this regard, the results obtained validate that this technique is efficient for preserving the imperceptibility of the hidden message.

Furthermore, regarding hiding capacity, the developed system allowed for the embedding of files several kilobytes in size within the audio without compromising playback; thus, the output WAV file (audio_esteganografiado.wav) sounds virtually identical to the original, with no loss of quality. This finding is consistent with studies in audio steganography that indicate that the LSB technique offers a balance between insertion capacity and signal quality, provided that the number of modified bits per sample is limited [11].

Similarly, more recent research confirms that steganography methods seek to simultaneously optimize the insertion capacity and the imperceptibility of the audio, while minimizing the distortion generated in the carrier signal [17].

When the program is run, it generates an encrypted PDF embedded within its audio samples. Upon extraction, the original file is recovered, and the output is displayed as: extracted_message.pdf, which is visually identical to the original.

The original file:

Figure 7. *Original file: mensaje.pdf*.
Source: Author's *own work* (2026).

The sound of the source file, when compared to that of the output file, shows no audible difference; they sound identical, and therefore the user cannot detect that information has been hidden within it.

A key aspect of this research is the incorporation of AES-256 encryption prior to the embedding process, which adds an additional layer of security. Unlike other traditional approaches that merely hide information, the proposed system ensures that the data cannot be interpreted without the corresponding key. This result aligns with the findings of Modibbo et al. [12], who describe how the integration of steganography and cryptography constitutes a multilayer security approach for handling sensitive information.

Therefore, it is confirmed that the design and implementation of the Python program codes were effective in achieving the objective set forth in this paper.

CONCLUSIONS

In this research, steganography was performed using Python, yielding good results by successfully hiding information using the least significant bit (LSB) method in a WAV-format audio file. Subsequently, Python code was used to extract the hidden message and generate an output file with a .pdf extension.

However, there are two main disadvantages associated with the use of methods such as LSB encoding. The human ear is very sensitive and can often detect even the slightest noise introduced into an audio file. The second disadvantage is its lack of robustness. If an audio file embedded with a secret message using LSB encoding is analyzed, the embedded information is lost[16]

This ensures that the hidden files go unnoticed by ordinary users, thereby protecting the encrypted information—text data—using a simple audio file. Therefore, the most notable conclusions can be summarized as follows:

1. Effectiveness of the LSB method: The technique proved effective for hiding information in WAV audio files, maintaining sound quality and allowing for the exact recovery of the hidden data.
2. Enhanced security: The combination of LSB steganography with AES-256 encryption ensures that the data remains inaccessible without the correct password, even if it is detected.
3. Appropriate Tools: The use of PyDub and NumPy facilitated audio processing, while Fernet provided robust and easy-to-implement encryption.
4. Limitations: The method is sensitive to audio compression or reprocessing, which which limits its use in environments where the file might be altered.
5. Practical applications: This approach can be used in covert communications, the protection of sensitive data, and digital watermarking, among other applications.

This research is relevant because it applies steganography in today's world to contribute to the security and privacy of personal information. It is expected that this work will contribute significantly to a better understanding and use of steganographic techniques and their benefits.

Future improvements:

It is recommended to explore techniques to increase storage capacity and resistance to audio manipulation.

As a proposal for future work, video steganography will be addressed to enhance information security.

ACKNOWLEDGMENTS

We would like to thank TecNM, especially the Minatitlán Institute of Technology, as well as the students and faculty of the Computer Systems Engineering program and the Department of Computer Systems Engineering.

REFERENCES

[1] Evsutin, A., Melman, and R. Meshcheryakov, "'Digital Steganography and Watermarking for Digital Images: A Review of Current Research Directions,'" IEEE Access, vol. 8, pp. 166589–166611, doi: 10.1109/access.2020.3022779., 2020. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9187785.

[2] Fortinet. What Is Cryptography? Definition, Importance, Types [Internet]. 2025 [cited 2026 Feb 13]. Available at: https://www.fortinet.com/lat/resources/cyberglossary/what-is-cryptography

[3] Arias A.G. Steganography in Digital Files. [Bachelor's Thesis]. Madrid: Complutense University of Madrid; 2021.

[4] Bahl J, Ramakishore R. The LSB technique and its variations used in audio steganography: a review. Int J Eng Technol Res. 2013;2(4):2327-2332.

[5] Blanco SA, Budiman AA. *Steganography in MP3 audio files to protect messages using the least significant bit (LSB) and Advanced Encryption Standard (AES) methods.* TIFDA Journal (Information Technology and Data Analysis). 2025;2(1):42–5.

[6] Digital Image Steganography. *A literature survey. Information Sciences.* 2022;606:32–47. doi:10.1016/j.ins.2022.05.053.

[7] Sion R. *Steganography.* In: Liu L, Özsu MT, eds. *Encyclopedia of Database Systems.* New York: Springer; 2018.

[8] Johnson NF, Duric Z, Jajodia S. *Information Hiding: Steganography and Watermarking—Attacks and Countermeasures.* Boston: Kluwer Academic/Plenum; 2001. p. 15.
[9] Ahmed MA, Hossain MS, Rahman MM. *Audio steganography with intensified security and hiding capacity. Int J Comput Appl.* 2025;2(1):1–10. Available at: https://www.researchgate.net/publication/389751245_Audio_Steganography_with_Intensified_Security_and_Hiding_Capacity
[10] Hossain MT. *Audio steganography using genetic algorithms for optimized signal-to-noise ratio and data capacity. J Eng Technol.* 2023;2(1):1–10. Available at: https://www.researchgate.net/publication/391234555_Audio_Steganography_Using_Genetic_Algorithms
[11] Cvejic N, Seppänen T. *Increasing the capacity of LSB-based audio steganography.* In: *Proceedings of the IEEE Workshop on Multimedia Signal Processing.* Siena: IEEE; 2004. p. 336–338. doi:10.1109/MMSP.2004.1436568.
[12] Modibbo AA, Ngene CU, Bulus B. *Audio steganography using AES encryption and LSB substitution for the protection of military data. Int J Pure Appl Sci.* 2025;8(9):004. doi:10.70382/nijpas.v8i9.004.
[13] Nasr MA, El-Shafai W, El-Rabaie EM, El-Fishawy AS, El-Hoseny HM, Abd El-Samie FE, Abdel-Salam N. *A robust audio steganography technique based on image encryption using different chaotic maps. Scientific Reports.* 2024;14(1):22054. doi:10.1038/s41598-024-70940-3.
[14] Pérez-Marín D, Pascual-Nieto. Fundamentals of Computer Security. Madrid: Ediciones Paraninfo; 2021
[15] Zeballos Soruco JA. Steganography of WAV audio files: insertion of executables and analysis of antivirus evasion**.** Rev Ciencia Tecnol Digit. 2025;1(1). Available at: https://revistas.usfx.bo/index.php/rctd/article/view/2068
[16] Sánchez Rinza BE, Munive Morales LG, Jaramillo Núñez A**.** LSB Algorithm to Hide Text in an Audio Signal. *Computing and Systems*. 2022;26(1):39-44.
[17] Hamdi AA, Eyssa AA, Abdalla MI, ElAffendi M, AlQahtani AA, Ateya AA, et al. Improving audio steganography transmission over various wireless channels. J Sens Actuator Netw. 2025;14(6):106.

Table of Contribution Roles

Role	Author(s)
Project Management	Wendy Carranza Díaz
Conceptualization	Guillermina Jiménez Rasgado
Software	Wendy Carranza Díaz—Lead, José Aurelio Ramírez González—Support.
Writing	María Concepción Villatoro Cruz
View	Wendy Carranza Díaz-Principal, Pablo Francisco Vivas Torres-Supporting.