EDUCACIÓN
IPSUMTEC
ISSN: 2594 - 2905

OCULTAR INFORMACIÓN DE TEXTO EN AUDIO .WAV USANDO LSB

HIDE TEXT INFORMATION IN .WAV AUDIO USING LSB

Carranza Díaz Wendy[1], Villatoro-Cruz María Concepción [2]
Jiménez Rasgado Guillermina[3], Ramírez González José Aurelio[4], Vivas Torres Pablo Francisco[5]

https://doi.org/10.61117/ipsumtec.v9i1.461

[1]Tecnológico Nacional de México /Instituto Tecnológico de Minatitlán, wendy.cd@minatitlan.tecnm.mx, https://orcid.org/0009-0009-4655-2816
[2]Tecnológico Nacional de México /Instituto Tecnológico de Minatitlán, maría.vc@minatitlan.tecnm.mx, https://orcid.org/0000-0002-8986-6219.
[3]Tecnológico Nacional de México /Instituto Tecnológico de Minatitlán, guillermina.jr@minatitlan.tecnm.mx, https://orcid.org/0000-0003-0163-0525,
[4]Tecnológico Nacional de México /Instituto Tecnológico Superior de Acayucan, aurelio.rg@acayucan.tecnm.mx, https://orcid.org/0009-0008-8986-0774
[5]Tecnológico Nacional de México /Instituto Tecnológico de Minatitlán, pablo.vt@minatitlan.tecnm.mx, https://orcid.org/0009-0001-5920-3668

Resumen -- El presente artículo describe el desarrollo de una librería en Python para la ocultación de información en archivos de audio mediante la técnica de esteganografía basada en el método Least Significant Bit (LSB). El objetivo fue diseñar un paquete reutilizable capaz de insertar y extraer archivos PDF dentro de archivos WAV sin afectar su calidad perceptible.
La investigación siguió un enfoque experimental dividido en cinco etapas: revisión del estado del arte, definición del algoritmo LSB, integración de cifrado simétrico con AES-256, implementación del código y validación de resultados. Se emplearon las bibliotecas PyDub para la manipulación del audio, NumPy para el manejo de muestras y Fernet para el cifrado. El proceso consistió en cifrar previamente el archivo PDF, modificar los bits menos significativos de las muestras de audio para insertar los datos y luego reconstruir el archivo WAV. La extracción se realizó de manera inversa, recuperando y descifrando el mensaje embebido.
Los resultados demostraron que el método LSB combinado con cifrado AES-256 permite ocultar información manteniendo la integridad del archivo original y sin alterar perceptiblemente el sonido. En todas las pruebas, tanto locales como en la plataforma PythonAnywhere, el archivo PDF fue recuperado de forma íntegra y el audio resultante conservó su calidad original.
Se concluye que la esteganografía en audio con Python es una alternativa efectiva para proteger información sensible, al combinar la seguridad del cifrado con la discreción del ocultamiento digital. Sin embargo, se reconoce la fragilidad del método frente a la compresión o manipulación del audio. Como trabajo futuro se propone optimizar la capacidad de almacenamiento y explorar técnicas más robustas de esteganografía.

Palabras Clave-- Cifrado AES, Esteganografía, LSB, Python, Seguridad Digital.

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 results validation. 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 inversely, 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 fragility against compression or audio manipulation is recognized. As future work, it is proposed to optimize data capacity and explore more robust steganographic techniques.

Key words **–** AES Encryption, Steganography, LSB, Python, Digital Security.

INTRODUCCIÓN

En la era digital actual, la protección de la información se ha convertido en una necesidad prioritaria debido al incremento de los ataques cibernéticos, la interceptación de datos y la vulnerabilidad de los sistemas de comunicación.

Hoy en día, la esteganografía tiene aplicaciones en la industria como el ocultamiento de la información de propiedad intelectual en archivos de audio e imágenes [1], así como en aplicaciones militares donde se utilizan los mensajes secretos.

Dentro del estado del arte, se puede mencionar que las técnicas tradicionales de seguridad, como la criptografía y la esteganografía protegen el contenido de la información, pero no impiden que terceros detecten su existencia.

Para comenzar, la criptografía se define como "el proceso de ocultar o codificar información mediante algoritmos y claves, de modo que los datos solo puedan ser interpretados por destinatarios previstos. Esto incluye técnicas como el cifrado y descifrado, que protegen información en tránsito o en reposo frente a accesos no autorizados"[2]. Por otra parte, "La esteganografía se compone de un método de embebido, incrustación, ocultación o codificación donde se ocultará la información dentro del archivo original al que también llamaremos portador"[3]. Además, en el ámbito del audio digital, la esteganografía ha adquirido relevancia por su capacidad de incorporar mensajes o archivos dentro de señales sonoras sin alterar perceptiblemente su calidad**.** En particular, el método del bit menos significativo (Least Significant Bit, LSB) se ha consolidado como una de las estrategias más utilizadas por su simplicidad, capacidad de almacenamiento y eficiencia computacional**.** De igual modo, se ha documentado variaciones del método clásico LSB y sus limitaciones en capacidad de datos y robustez [4]

En palabras de un estudio reciente, "El método del bit menos significativo (LSB) se utilizó como un algoritmo de transformación esteganografía. El bajo nivel de fiabilidad del método puede mejorarse mediante la introducción de una capa criptográfica."[5]. Este planteamiento respalda la necesidad de combinar esteganografía con técnicas de cifrado, como el AES-256, para garantizar la confidencialidad de los datos ocultos.

Sin embargo, la mayoría de las implementaciones existentes carecen de mecanismos integrados de cifrado**,** lo que las hace vulnerables si los datos ocultos son descubiertos.

Ante esta limitación, la esteganografía surge como una herramienta complementaria que permite ocultar información dentro de otros medios digitales, de modo que su presencia pase inadvertida para un observador no autorizado. La esteganografía en audio ha ganado relevancia como técnica para ocultar información sensible en archivos de sonido sin comprometer su calidad perceptible.

El presente trabajo aborda esta limitación mediante el desarrollo de una librería en Python que integra la técnica de esteganografía LSB con cifrado simétrico AES-256**,** permitiendo ocultar archivos PDF dentro de archivos de audio WAV**.** Esta combinación ofrece una capa adicional de seguridad, garantizando que el mensaje permanezca inaccesible sin la clave correcta, incluso si es detectado.

Este artículo presenta una implementación práctica en Python que combina el método LSB (Least Significant Bit) con cifrado AES para ocultar archivos PDF dentro de archivos WAV. La solución aprovecha bibliotecas como NumPy y PyDub para manipular muestras de audio, garantizando que los datos ocultos permanezcan inaccesibles sin la contraseña correcta.

Además de describir el diseño del código, se documenta sus resultados y recomendaciones.

El objetivo general de la investigación es diseñar e implementar una librería reutilizable en Python para ocultar y recuperar información en archivos de audio mediante el método LSB con cifrado AES, acompañada de su documentación técnica y validación experimental.

La justificación de este trabajo radica en la necesidad de contar con soluciones abiertas, seguras y accesibles para el ocultamiento de información sensible, aplicadas en áreas como la seguridad digital, la comunicación encubierta y la protección de datos. Así mismo, el uso de Python como lenguaje de implementación permite fomentar el aprendizaje y la replicabilidad del método en contextos educativos y de investigación.

MARCO TEÓRICO

Esteganografía:

"La esteganografía es el arte de ocultar información en un medio portador de tal manera que la presencia de dicha información sea desconocida" [6]

De manera similar Sion, la define así: "La esteganografía (del griego steganos, 'cubierto') es un término que denota mecanismos para ocultar información dentro de un 'medio portador' de modo que, por lo general, sólo el receptor previsto tenga conocimiento de su existencia y sea capaz de recuperarla de dicho medio". [7]

Por lo tanto, se puede decir que la esteganografía es la ciencia de ocultar información dentro de otros medios digitales, como imágenes, audio o video, de manera que su presencia pase desapercibida. A diferencia de la criptografía, que protege el contenido del mensaje, la esteganografía busca ocultar su existencia.

Acerca de la importancia de la esteganografía, y para resaltar la diferencia a la criptografía, se puede decir que: "la esteganografía difiere de la criptografía en que su objetivo no es proteger el contenido del mensaje, sino ocultar su existencia misma". [8]

Por otro lado, exploran cómo la fusión de esteganografía y criptografía en audio, utilizando LSB mejorado, fortalece la protección de datos sensibles. [9]

Asimismo,[10] introduce un enfoque que emplea algoritmos genéticos para optimizar la capacidad de ocultación y la calidad de la señal en la esteganografía en audio.
Por lo que, en un archivo de audio digital (como WAV) se almacenan muestras de amplitud en formato PCM (Pulse Code Modulation). Cada muestra es un valor numérico cuantizado (generalmente de 16 bits), donde el bit menos significativo (LSB) tiene un impacto mínimo en la percepción humana del sonido.
LSB (Least Significant Bit)
El método LSB consiste en reemplazar los bits menos significativos de las muestras de audio con los bits del mensaje secreto. Si se modifican solo 1 o 2 LSBs, la distorsión introducida es casi imperceptible al oído humano. Al respecto se puede comentar que el método LSB en audio es: "Modificar los bits menos significativos (LSB) en señales de audio que permiten ocultar datos con una distorsión mínima, siempre que se limite a 1-2 bits por muestra" [11].

Ejemplo:
• Muestra original (16 bits): 11011010 10101101
• Bit a ocultar: 1
• Muestra modificada: 11011010 1010110**1** (solo cambia el último bit)

Componentes del Sistema Propuesto:
El esquema implementado en el código analizado consta de tres etapas principales:

Cifrado del Mensaje (Seguridad Adicional):
Para el cifrado del mensaje, se destaca la eficacia de combinar esteganografía en audio con cifrado AES y LSB para mejorar la seguridad en comunicaciones militares [12], lo que resulta en verdad interesante, pues ofrece una multicapa de seguridad, por lo tanto y tomando en cuenta esta fuente, antes de ocultar los datos, se aplica cifrado simétrico (AES-256 mediante Fernet) usando una clave derivada de una contraseña mediante SHA-256. Esto garantiza que, incluso si el mensaje es detectado, no pueda ser leído sin la clave correcta.
Por otro lado, se propone una técnica avanzada que utiliza mapas caóticos para cifrar imágenes antes de insertarlas en señales de audio, mejorando la seguridad del proceso. [13]

Inserción Esteganografía (LSB)

- Entrada: Audio original (WAV) + Datos cifrados.
- Proceso:
  1. Se convierte el audio en un arreglo de muestras (usando pydub y numpy).
  2. Se reemplazan los *n* LSBs de cada muestra por los bits del mensaje cifrado.
  3. Se reconstruye el archivo WAV con las muestras modificadas.

Extracción y Descifrado

- Entrada: Audio estenografiado + Contraseña.
- Proceso:
  1.Se extraen los LSBs de las muestras de audio.
  2. Se reconstruyen los bytes del mensaje cifrado.
  3. Se descifran los datos usando la contraseña proporcionada.

Ventajas y Limitaciones

Ventajas:

- Baja perceptibilidad: Modificar 1-2 LSBs no afecta significativamente la

calidad del audio.

- Alta capacidad: Dependiendo del número de bits modificados, se pueden ocultar KBs de datos en segundos de audio. Seguridad mejorada: El cifrado previo evita el acceso al mensaje sin la clave.

Limitaciones:

Fragilidad: El método LSB es sensible a compresión, reprocesamiento o ruido. Detección posible: Análisis estadístico avanzado puede revelar la presencia de datos ocultos.

Dependencia del formato: Funciona mejor en archivos WAV sin compresión (PCM).

Aplicaciones en Tecnologías de la Información:
• Comunicaciones encubiertas: Transmisión de mensajes secretos en streaming de audio.
• Marcado digital: Inclusión de watermarks para derechos de autor.
• Seguridad ofensiva/defensiva: Uso en pentesting o protección de datos sensibles.

Definición de Librería (en el contexto de programación):
Una librería es un conjunto de funciones, métodos y clases predefinidas que permiten:

- Reutilizar código sin tener que programarlo desde cero.
- Estandarizar procesos (ej: cifrado, manejo de archivos, procesamiento de señales).
- Optimizar el rendimiento, ya que muchas librerías están altamente optimizadas (ej: NumPy para operaciones matemáticas rápidas).

DESARROLLO
El proceso esteganográfico se fundamenta en la tríada compuesta por el mensaje secreto (información a ocultar), el objeto portador o cover object(medio original) y el objeto estego (resultado del proceso que contiene la información embebida) [14]
Bajo este enfoque, la metodología aplicada para el desarrollo del trabajo se estructuró de la siguiente manera:

Metodología:

1. Búsqueda de información relevante en la web
2. Definición y análisis del algoritmo LSB(Least Significant Bit)
3. Selección de los archivos de origen, incluyendo audios y documentos en formato PDF.
4. Organización y ubicación de todos los archivos dentro de la estructura de la librería desarrollada.
5. Verificación del funcionamiento del sistema y depuración de errores detectados
6. Comprobación y validación de los resultados obtenidos.

Búsqueda de información relevante en la web:
En primer lugar, se llevó a cabo una investigación exhaustiva de ejemplos de esteganografía ya existentes, por medio de una revisión del estado del arte de esteganografía. Es importante comprender primero el método LSB. Así que el enfoque fue en el método LSB aplicado a archivos de audio. Se consultaron artículos científicos, libros y repositorios de código para entender las implementaciones existentes y sus limitaciones.
Investigaciones recientes han explorado la combinación de esteganografía con criptografía caótica, lo que incrementa la capacidad de ocultación y la resistencia frente a técnicas de detección. [15]

Definición y análisis del algoritmo LSB(Least Significant Bit):
La solución utiliza LSB (Least Significant Bit- Bit Menos Significativo) que es un método clásico de esteganografía. Esta técnica modifica los bits menos significativos de las muestras de audio. Pero, estos cambios son casi imperceptibles al oído humano. Se diseñó un algoritmo basado en el método LSB para modificar los bits menos significativos de las muestras de audio, asegurando que los cambios fueran imperceptibles al oído humano. Esto permite embeber datos binarios dentro del archivo de audio.

Selección de los archivos de origen, incluyendo audios y documentos en formato PDF:
Se buscó y seleccionó un archivo de audio y de PDF para utilizar como portador y datos secretos

Además, el lenguaje de programación elegido es Python. El IDE de desarrollo es: Visual Studio Code y para realizar las pruebas, se utilizó el compilador de pythonanywhere.com.
Python es un lenguaje de programación muy actual. Es un lenguaje de alto nivel, interpretado y de propósito general, famoso por ser claro, legible y fácil de aprender.

Conjuntamente, se eligieron las bibliotecas PyDub para manipulación de audio y NumPy para procesamiento eficiente de muestra.
En el caso de la herramienta Pythonanywhere se trata de una plataforma, es un servicio en la nube que permite alojar, ejecutar y desarrollar aplicaciones Python directamente desde un navegador web. Es especialmente útil para principiantes, para educadores y desarrolladores que quieren desplegar proyectos web sin tener que configurar servidores complejos.
Organización y ubicación de todos los archivos dentro de la estructura de la librería desarrollada:
Todos los archivos seleccionados, así como el código y archivos necesarios se ubicaron en una carpeta con el nombre de la librería desarrollada.

Es importante mencionar que todos los archivos seleccionados, así como el código se guarden dentro de la misma ubicación del archivo a ejecutar. Así mismo dentro de la misma ubicación también se pueden guardar los archivos generados.

Verificación del funcionamiento del sistema y depuración de errores detectados:

- Cifrado: Los datos del archivo PDF se cifraron antes de ser incrustados en el audio. En términos generales, para la ocultación de la información se pensó en cifrar el PDF con la contraseña, convertir el audio a muestras numéricas y ocultar los datos modificando los LSBs, y guardar el archivo de audio en audio_esteganografiado.wav y el archivo de salida mensaje_extraido.pdf
- Inserción: Las muestras de audio se modificaron para incluir los bits del mensaje cifrado.
- Extracción: Se recuperaron los bits LSB del audio y se descifraron para reconstruir el PDF original. El archivo que se utilizó para solucionarlo es el de mensaje.pdf y original.wav.

Comprobación y validación de los resultados obtenidos:
Después de ocultar, continúa el proceso de extracción, donde se leen las muestras de audio y extraen los bits menos significativos, usando la contraseña para descifrar los datos y se rehace el PDF original.
Se recomienda que los archivos mantengan el formato que se espera y concuerde con la extensión que maneja el código .wav o en su defecto .pdf
En el caso de la librería, se propone que quede organizado de la siguiente manera:

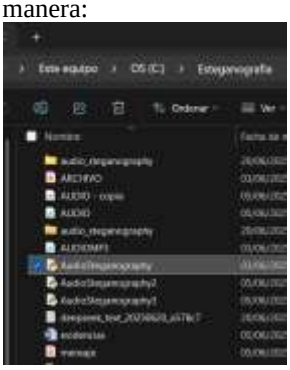

Figura 1. *Estructura de la librería audiosteganography*.
Fuente. *Elaboración propia* (2026).

Esteganografía (LSB): Se implementa manualmente, pero NumPy y PyDub facilitan el manejo de las muestras de audio.
Cifrado: Delegado completamente a Fernet (de la librería cryptography).
Manejo de archivos: PyDub se encarga de cargar/guardar el audio, mientras que Python maneja el PDF con open() nativo.
Este código implementa un sistema de esteganografía que oculta archivos PDF dentro de archivos de audio WAV utilizando técnicas de cifrado y manipulación de bits.
El enfoque de este trabajo es usar la técnica del bit menos significativo (LSB, Least Significant Bit) aplicada a señales de audio digital en formato WAV, debido a su simplicidad y eficiencia en el ocultamiento de datos.

Para la validación de los resultados obtenidos se llevaron a cabo diversas pruebas:

- ✓ Prueba 1:
  Ocultamiento y extracción exitosa en pythonanywhere.com, verificando la integridad del PDF.
- ✓ Prueba 2:
  Ejecutar de manera local en CMD y confirmando la generación correcta de los archivos de salida.
- ✓ Prueba 3:
  Análisis de calidad del audio, donde no se detectaron diferencias perceptibles al oído humano entre el original y el modificado.

En la validación se comprobó que el archivo PDF extraído fuera idéntico al original y que el audio mantuviera su calidad.

CÓDIGO DE EJEMPLO

El código implementado incluye funciones para cifrar, ocultar, extraer y descifrar datos. Se utilizó la clase **AudioSteganography3** para encapsular la lógica, facilitando su reutilización. Las pruebas demostraron que el sistema es capaz de manejar archivos de hasta varios KB sin comprometer la calidad de audio.

El siguiente Código se utilizó para probar la librería:

```
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 demasiado pequeño para los datos. Requiere {len(data_bits)} bits pero solo hay {max_bits} disponibles.")

        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 oculto en {output_path} con {lsb_bits} bits LSB.")
            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 al descifrar los datos: ¿la contraseña es
                    correcta?") from e
            with open(output_pdf_path, 'wb') as f:
            f.write(decrypted_data)
            print(f" PDF extraído correctamente a: {output_pdf_path}")
# ========================
# Código de prueba
# ========================
if __name__ == "__main__":
            stego = AudioSteganography3()
# Cambia estas rutas según donde tengas tus archivos pdf_path = "mensaje.pdf"
audio_path = "original.wav"
output_stego_audio = "audio_esteganografiado.wav" extracted_pdf_path = "mensaje_extraido.pdf" password = "miclave123"
lsb_bits = 2
# Ocultar PDF en audio
stego.hide_pdf_in_audio(pdf_path,audio_path, output_stego_audio, password, lsb_bits)
# Extraer PDF del audio esteganografiado
stego.extract_pdf_from_audio(output_stego_audio, extracted_pdf_path, password, lsb_bits)
```

En primer lugar, se utilizó la plataforma en línea de Pythonanywhere, en la cual se colocaron los archivos origen, con extensión .wav y .pdf, obteniéndose buenos resultados.

Ejecución:

Figura 2. *Prueba del código AudioSteganography3.py en pythonanywere*.
Fuente. *Elaboración propia* (2026).

ARCHIVOS GENERADOS:

- mensaje_extraido.pdf
- audio_esteganografiado.wav

A continuación, en la figura 3, se mostrará la lista de archivos que se cargaron en la plataforma de Pythonanywhere que es abierta para todo el público en general y no necesita que se instalen previamente en la computadora, ningún software para probar los códigos.
Su URL es la siguiente: https://www.pythonanywhere.com/ pythonanywhere es una plataforma en la nube (un IDE en línea y alojamiento en la web), eliminando la necesidad de instalar Python y configurar localmente.

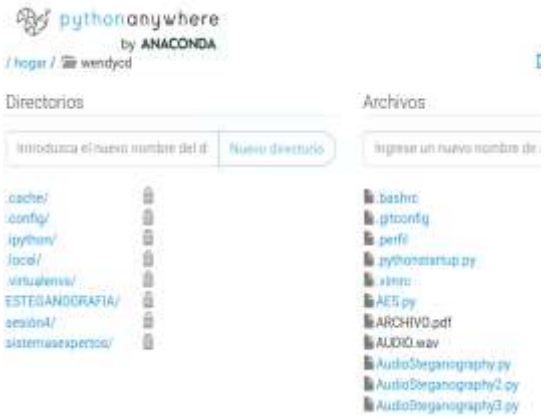

Figura 3. *Repositorio de Códigos en pythonanywhare.com*.
Fuente. *Elaboración propia* (2026).

Al abrir el archivo: mensaje_extraido.pdf, se observa lo que muestra la figura 4.

Figura 4. *Archivo del mensaje de prueba oculto en el audio*.
Fuente. *Elaboración propia* (2026).

Posteriormente, se realizó la prueba usando cmd en la computadora, de la siguiente manera:

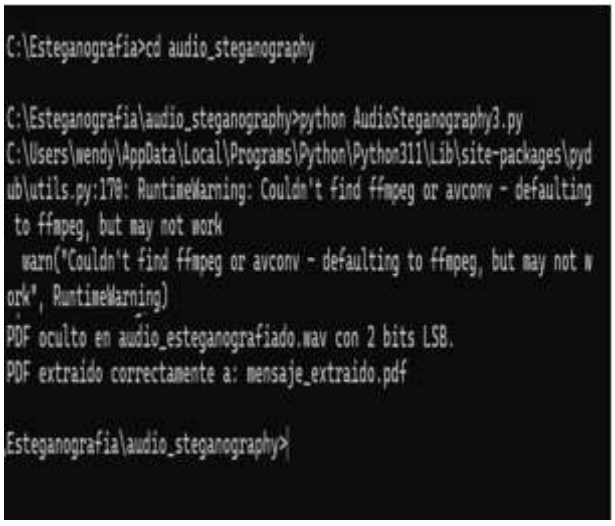

Figura 5. *Ejecución de la prueba usando la librería audio_steganography*.
Fuente: *Elaboración propia* (2026).

El resultado fue favorable y para comprobarlo se muestra la siguiente imagen donde ya figuran los archivos generados usando la librería.

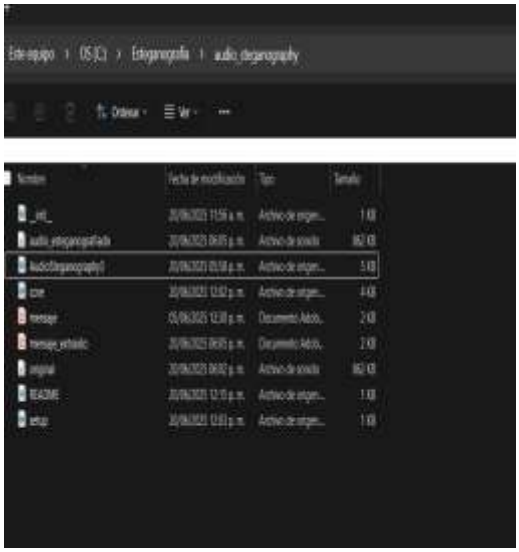

Figura 6. *Archivos de salida generados correctamente*.
Fuente. *Elaboración propia* (2026).

DISCUSIÓN Y ANÁLISIS DE RESULTADOS

Los resultados obtenidos en la presente investigación demuestran que la implementación de la técnica de esteganografía basada en LSB combinada con cifrado AES-256 es funcional y efectiva para el ocultamiento de la información en archivos de audio en formato WAV.

En todas las pruebas realizadas, tanto en entorno local como en la plataforma PythonAnywhere, se logró la inserción y posterior recuperación del archivo PDF sin pérdida de información, lo que confirma la integridad del proceso.

Se cumplió el objetivo de esta investigación al diseñar e implementar una librería reutilizable en Python para ocultar y recuperar información en archivos de audio mediante el método LSB con cifrado AES. En términos de calidad del audio, no se detectaron diferencias perceptibles entre el archivo original y el archivo estenografiado. Este comportamiento coincide con lo reportado por Sánchez Rinza et al. [16], quienes evidencian que el uso del método LSB en audio permite ocultar información manteniendo una baja distorsión de la señal. En ese sentido, los resultados obtenidos validan que esta técnica es eficiente para preservar la imperceptibilidad del mensaje oculto.

Por otra parte, en relación con la capacidad de ocultamiento, el sistema desarrollado permitió incrustar archivos de varios kilobytes dentro del audio sin comprometer su reproducción, tal que, la salida del archivo WAV (audio_esteganografiado.wav), suena prácticamente igual que el original, sin afectar la calidad. Este hallazgo es consistente con estudios en esteganografía de audio que señalan que la técnica LSB ofrece un equilibrio entre la capacidad de inserción y la calidad de la señal, siempre que se limite el número de bits modificados por muestra [11].

Así mismo, investigaciones más recientes confirman que los métodos de esteganografía buscan optimizar simultáneamente la capacidad de inserción y la imperceptibilidad de audio, minimizando la distorsión generada en la señal portadora [17].

Al ejecutar el programa, se genera el PDF cifrado incrustado en sus muestras de audio. Al extraer, se recupera el archivo original, muestra la salida como: mensaje_extraido.pdf el cual es idéntico al original visualmente.

El archivo original:

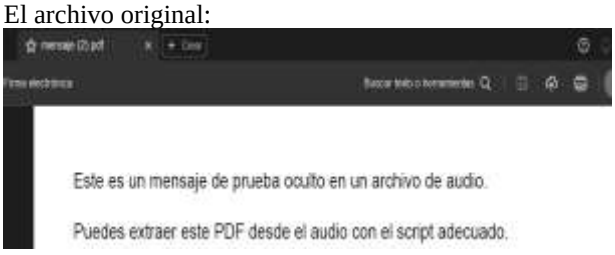

Figura 7. *Archivo original: mensaje.pdf*.
Fuente. *Elaboración propia* (2026).

El sonido del archivo de origen en comparación con el de salida, no presenta diferencia auditiva alguna, tal que se escuchan semejantes, y por lo tanto el usuario no identifica la ocultación de información en el mismo.
Un aspecto relevante de esta investigación es la incorporación de cifrado AES-256 previo al proceso de inserción, lo cual añade una capa adicional de seguridad. A diferencia de otros enfoques tradicionales que únicamente ocultan la información, el sistema propuesto garantiza que los datos no puedan ser interpretados sin clave correspondiente. Este resultado coincide con lo señalado por Modibbo et al. [12], quienes describen que la integración de esteganografía y criptografía constituye un enfoque de seguridad multicapa para la producción de información sensible.
Por lo anterior se valida que el diseño e implementación de los códigos de programación en Python fueron efectivos en el logro del objetivo planteado en el presente trabajo.

CONCLUSIONES

En esta investigación se realizó esteganografía utilizando Python, obteniéndose buenos resultados al lograr ocultar información mediante el método de bit menos significativo (LSB), en un archivo de audio en formato WAV. Posteriormente, mediante código en Python se programó la extracción del mensaje oculto, y la generación de un archivo de salida con extensión .pdf.
Sin embargo, existen dos desventajas principales asociadas con el uso de métodos como la codificación LSB. El oído humano es muy sensible y a menudo puede detectar el mínimo ruido introducido en un archivo de audio. La segunda desventaja es su falta de robustez. Si se demuestra un archivo de sonido integrado con un mensaje secreto mediante codificación LSB, se pierde la información incrustada[16]
Esto genera que los archivos ocultos pasen inadvertidos para los usuarios comunes y de esta forma proteger la información encriptada, datos de texto usando un simple archivo de audio. Por lo tanto, dentro de las conclusiones más destacables se pueden resumir de la siguiente manera:

1. Efectividad del método LSB: La técnica demostró ser eficaz para ocultar información en archivos de audio WAV, manteniendo la calidad del sonido y permitiendo la recuperación exacta de los datos ocultos.
2. Seguridad mejorada: La combinación de esteganografía LSB con cifrado AES-256 asegura que los datos permanezcan inaccesibles sin la contraseña correcta, incluso si son detectados.
3. Herramientas adecuadas: El uso de PyDub y NumPy facilitó el procesamiento de audio, mientras que Fernet proporcionó un cifrado robusto y fácil de implementar.
4. Limitaciones: El método es sensible a compresiones o reprocesamientos del audio, lo que limita su uso en entornos donde el archivo pueda ser alterado.
5. Aplicaciones prácticas: Este enfoque puede utilizarse en comunicaciones encubiertas, protección de datos sensibles y marcas de agua digitales, entre otros.

La presente investigación adquiere relevancia al aplicar la esteganografía en la actualidad para contribuir a la seguridad y privacidad de la información personal. Se espera que el trabajo contribuya significativamente en mayor medida para la comprensión y uso de las técnicas estenográficas y sus beneficios.
Futuras mejoras:
Se recomienda explorar técnicas para aumentar la capacidad de almacenamiento y la resistencia a manipulaciones de audio.
Como propuesta de trabajo futuro, se abordará la esteganografía en video para aumentar el nivel de seguridad de la información.

AGRADECIMIENTOS

Agradecemos al TecNM, en especial al Instituto Tecnológico de Minatitlán, y a los alumnos y maestros de la Carrera de Ingeniería en Sistemas Computacionales y a la Jefatura de Ingeniería en Sistemas Computacionales.

BIBLIOGRAFÍA

[1] Evsutin, A. Melman y 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. [En línea]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9187785.
[2] Fortinet. ¿Qué es la criptografía? Definición, importancia, tipos [Internet]. 2025 [citado 2026 Feb 13]. Disponible en: https://www.fortinet.com/lat/resources/cyberglossary/what-is-cryptography
[3] Arias A.G. Esteganografía en archivos digitales. [Tesis de Licenciatura]. Madrid: Universidad Complutense de Madrid;2021.
[4] Bahl J, Ramakishore R. Técnica LSB y sus variaciones utilizadas en la esteganografía de audio: una revisión. Rev Int Investig Ing Tecnol. 2013;2(4):2327-2332.
[5] Blanco SA, Budiman AA. *Esteganografía en archivos de audio MP3 para proteger mensajes utilizando los métodos del bit menos significativo (LSB) y del Estándar de Cifrado Avanzado (AES).* Rev TIFDA (Tecnología de la Información y Análisis de Datos). 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.* En: Liu L, Özsu MT, editores. *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. Disponible en: 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. Disponible en: https://www.researchgate.net/publication/391234555_Audio_Steganography_Using_Genetic_Algorithms
[11] Cvejic N, Seppänen T. *Increasing the capacity of LSB-based audio steganography.* En: *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. *Esteganografía de audio mediante cifrado AES y sustitución LSB para la protección de datos militares. 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. *Técnica robusta de esteganografía en audio basada en el cifrado de imágenes mediante diferentes mapas caóticos. Scientific Reports.* 2024;14(1):22054. doi:10.1038/s41598-024-70940-3.
[14] Pérez-Marín D, Pascual-Nieto. Fundamentos de seguridad informática. Madrid: Ediciones Paraninfo; 2021
[15] Zeballos Soruco JA. Esteganografía de archivos de audio WAV: inserción de ejecutables y análisis de evasión antivirus**.** Rev Ciencia Tecnol Digit. 2025;1(1). Disponible en: 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. *Computación y Sistemas*. 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.

Tabla de roles de contribución

Rol	Autor (es)
Administración del proyecto	Wendy Carranza Díaz
Conceptualización	Guillermina Jiménez Rasgado
Software	Wendy Carranza Díaz-Principal, José Aurelio Ramírez González-Apoya.
Redacción	María Concepción Villatoro Cruz
Visualización	Wendy Carranza Díaz-Principal, Pablo Francisco Vivas Torres-Apoya.

Esta obra está bajo una licencia internacional Creative Commons Atribución 4.0