# Steganografie Audio

**Autor: Ion Bianca-Andreea, 333AA**

**An: 2024**

# Cuprins

# Introducere

**Steganografia** reprezintă tehnica de ascundere a informațiilor într-un fișier obișnuit, non-secret, sau într-un mesaj, fără a putea fi detectate. Steganografia poate fi folosită pentru ascunderea conținutului text, imagine, video sau audio în fișierul gazdă, conținut care este apoi extras la destinație. Acesta poate fi criptat sau procesat înainte de a fi introdus în fișierul gazdă pentru a fi mai greu de interceptat.
Dintr-o perspectivă digitală, există următoarele tipuri de steganografie:

1. Text
2. Imagine
3. Video
4. Audio
5. Network

**Steganografia audio** implică integrarea mesajului secret în semnalul audio, fapt care implică alterarea secvenței binare a fișierului în cauză.

Steganografia reprezintă o cale prin care atacatorii încearcă să infecteze dispozitivele victimelor. Atacurile pot folosi fișiere infectate cu malware. Aceste atacuri sunt adesea combinate cu alte tipuri de atacuri, cum ar fi phising, ransomware sau skimming. Un exemplu de acest fel a fost descoperit de cercetătorii de la Sansec [1] care au descoperit un payload malițios ascuns într-un fișier SVG. Un alt atac SteganoAmor realizat de TA558 folosea o imagine JPG pentru a ascunde un payload. Acesta conținea un script cu un cod PowerShell care descărca payload-ul final ascuns într-un fișier text sub forma unui executabil criptat folosind base64[1].

**Tema de proiect** aleasă implică realizarea unei steganografii audio prin 2 metode, una folosind tehnica LSB[2] (least significant bit), iar cealaltă transmițând informația prin pauzele dintre sunete.

**Obiectivele acestui proiect sunt următoarele**:

- Realizarea steganografiei audio folosind tehnica LSB, dar folosind ultimii 2 biți în loc de ultimul
- Implementarea unei metode de steganografie audio care transmite informație (un cod PIN) prin pauzele dintre 2 sunete consecutive ale unui semnal audio
- Realizarea unei comparații între cele 2 tehnici
- Implementarea unei metode de criptare a mesajului secret pentru tehnica LSB

[1] Base64 - schemă de codificare binară a textului care reprezintă date binare într-un format de șir ASCII (American Standard Code for Information Interchange)
[2] LSB – metoda Least Significant Bit este o metodă de a ascunde un mesaj în bitul cel mai din dreapta a fișierului gazdă

# Suport tehnic

Există numeroase abordări de realizare a steganografiei audio folosind tehnica LSB. Acestea, de cele mai multe ori, sunt realizate folosind limbajul Python sau Matlab. Două exemple în acest sens sunt prezentate în [2] și în [3].

# Prezentare tehnică a etapei de implementare

## Metoda LSB

Metoda Least Significant Bit presupune ascunderea unui bit din reprezentarea mesajului care trebuie transmis, în ultimul bit din fiecare cadru al fișierului audio (fișierul audio trebuie convertit și el în binar). Această operație alterează fișierul audio, dar numărul de biți care sunt alterați nu schimbă, de obicei, semnalul audio în așa măsură încât să fie detectat de urechea umană. Cu cât numărul de biți alterați din fiecare cadru crește, se introduce zgmot în semnalul audio, iar mesajul secret poate fi mai ușor de interceptat.

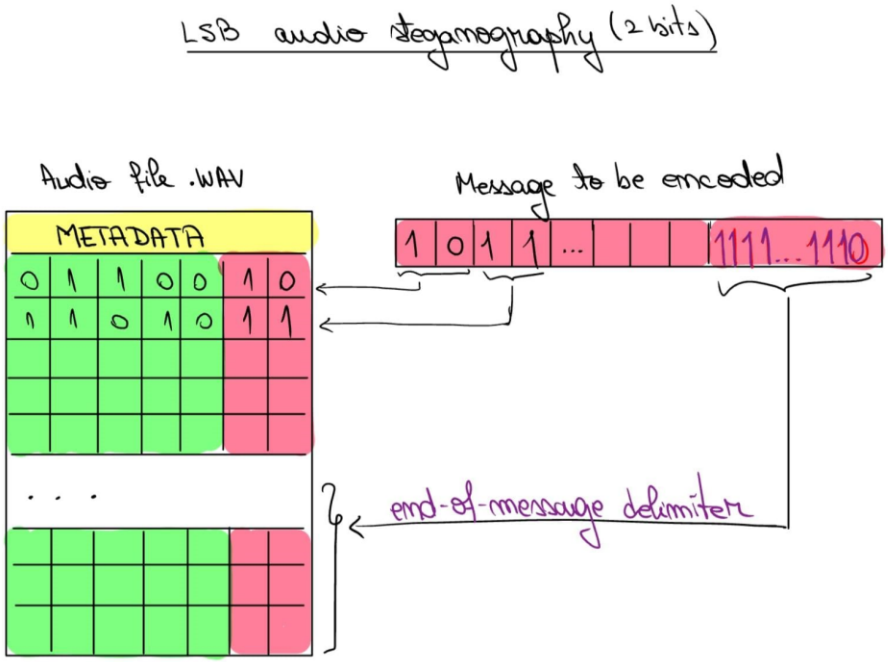


Fig. 1 Reprezentarea codificării mesajului în fișierul audio în care se alterează ultimii 2 biți

Pentru realizarea steganografiei audio cu tehnica LSB am folosit limbajul Python.

## Biblioteci

- random
- tkinter
- wave

Librăria **random** m-a ajutat în crearea funcțiilor de criptare și decriptare, variabilele care dau direcția inițială de mers și saltul fiind generate random.

Librăria **tkinter** m-a ajutat în implementarea interfeței pentru utilizator și a interacțiunii cu acesta prin mesajele de informare și de eroare.

Librăria **wave** m-a ajutat în deschiderea, citirea, scrierea și manipularea fișierelor .WAV [4].

## Algoritm

### Codificare

- Mesajul secret este criptat, apoi convertit în reprezentare binară. Se folosește reprezentarea binară pe 8 biți.
- Se adaugă secvența 1111111111111110, numită și delimitator de mesaj, la finalul reprezentării binare a mesajului pentru a marca sfârșitul acestuia. Secvența binară este utilă în procesul de decodificare, deoarece se poate marca sfârșitul mesajului și, prin urmare, la găsirea acestei secvențe se poate opri procesul de decodificare.
- Se citește informația utilă din fișierul audio și se stochează sub formă de byte în variabila *frames*.
- Se iterează prin fiecare cadru (*frame*) și, cu ajutorul operațiilor pe biți, se realizează următoarele operații:
    - Se folosește operația *& (și logic)* cu o mască ($252_{10} = 11111100_2$) pentru a păstra neschimbați primii 6 biți din reprezentarea binară a cadrului curent
    - Se modifică ultimii doi biți ai cadrului cu 2 biți din reprezentarea binară a mesajului secret printr-o operație de | *(sau logic)*
- Se creează fișierul *stegano_audio.wav* cu mesajul integrat

### Decodificare

- Se citește informația utilă din fișierul audio și se stochează sub formă de byte în variabila *frames*.
- Se extrag ultimii doi biți, cu ajutorul unei măști = 3, din reprezentarea binară a fiecărui cadru *(frame)* și se adaugă variabilei *extracted_message*
- La fiecare pas se verifică dacă am terminat de citit mesajul prin compararea ultimilor 16 biți găsiți cu cei din secvența 1111111111111110
- Se extrage câte un byte din variabila *extracted_message* și se convertește în caracterul ASCII corespunzător
- Se decriptează mesajul
- Se creează fișierul *SecretMessage.txt* care conține mesajul secret din fișierul .WAV

**Criptare**

Metoda de criptare este inspirată de codul Caesar Cipher care funcționează după cum urmează:

- Cifrul Caesar, folosit de Gaius Julius Caesar, este un cifru de substituție în care fiecare literă din textul inițial este înlocuită cu o literă care se află în alfabet la o distanță fixă față de cea înlocuită.

Metoda de **criptare** utilizează 4 variabile:

ascii _characters

- listă de caractere cu codul ASCII între 0 și 126

pad

- reprezintă saltul/deplasarea pe care urmează să îl/o facem pentru criptare
- poate avea valori între 1 si 9
- valoarea sa este aleatorie (generată cu *random.randrange()*)

fb

- provine de la forward-backward (înainte – înapoi)
- reprezintă direcția de rotație curentă (înainte - 1 sau înapoi - 2, în funcție de paritate)
- la început valoarea sa este 1 sau 2 (aleatorie - generată cu *random.randrange()*)
- valoarea sa este incrementată pentru fiecare caracter din mesaj care urmează să fie criptat atunci când îl iterăm

initial_fb

- păstrează prima valoare a lui fb (utilă pentru decriptare)

Metoda de criptare pe care am implementat-o schimbă literele din mesaj pentru a fi criptate după cum urmează:

1. este generată valoarea *pad*-ului
2. este generată valoarea *fb*
3. *initial_fb* este egal cu *fb*
4. se iterează prin mesajul care trebuie să fie criptat
   4.1. indexul caracterului curent este căutat în șirul *ascii_characters*
   4.2. se calculează indexul noului caracter, pe baza parității *fb*
   4.3. *pad* este adăugat la indexul găsit (*fb* = 1) sau scăzut din acesta (*fb* = 2)
   4.4. dacă noua valoare a indexului este mai mare decât zero, calculăm restul împărțirii la lungimea listei care conține caracterele ASCII *(ascii_characters)* pentru a fi siguri că parcurgem corect șirul

4.5. dacă noul indice este nu este mai mic decât zero, nu mai calculăm restul împărțirii la lista cu caracterele ASCII *(ascii_characters)*
5. adăugăm valoares *pad* la mesajul criptat
6. adăugăm valoarea *initial_fb* la mesajul criptat
7. returnăm mesajul criptat

Mesajul criptat va avea întotdeauna cu 2 caractere mai mult decât mesajul original, ultimul este *fb* și al doilea este *pad* (util pentru decriptare)

Metoda de **decriptare** funcționează similar, dar direcțiile rotațiilor sunt inversat, în comparație cu metoda de criptare (înainte - 2, înapoi - 1).

## Metoda Miau

**Biblioteci**

- tkinter
- pygame
- time
- sounddevice
- numpy
- matplotlib

Librăria **tkinter** m-a ajutat în implementarea interfeței pentru utilizator și a interacțiunii cu acesta prin mesajele de eroare care pot apărea.

Librăria **pygame** m-a ajutat în redarea fișierului audio cu conținutul sunetului *miau*, folosind funcțiile pygame.mixer.music.load() pentru a încărca fișierul audio și pygame.mixer.music.play() pentru a da play acestuia. [5]

Librăria **time** m-a ajutat în implementarea pauzelor dintre două sunete *miau* prin care se transmite informația (se așteaptă 4 secunde pentru a transmite 1 și 5 secunde pentru a transmite 0).

Librăria **sounddevice** m-a ajutat la înregistrarea sunetelor *miau* emise. [6]

Librăria **numpy** m-a ajutat în convertirea înregistrării de la stereo la mono prin calcularea mediei valorilor de pe cele 2 canale pe care s-a făcut înregistrarea.

Librăria **matplotlib** m-a ajutat în vizualizarea pauzelor din înregistrare prin crearea unui grafic pe baza înregistrării obținute.

## Algoritm

### Emitere

- Se introduce un PIN din 4 cifre și se verifică dacă acesta este valid, altfel se primește un mesaj de eroare
- PIN-ul este convertit în reprezentare binară pe 4 biți pentru a putea fi transmis prin pauzele dintre 2 sunete
- Se redă fișierul audio după cum urmează:
    - Dacă trebuie transmis un bit = 1, pauza dintre 2 redări este de 4 secunde
    - Dacă trebuie transmis un bit = 0, pauza dintre 2 redări este de 5 secunde

### Recepționare

- Se înregistrează sunetul emis de aplicație.
- Prin trasarea semnalului înregistrat se observă vârfurile corespunzătoare sunetului pisicii. Din cauza frecvenței de eșantionare, există multe mostre reprezentând același lucru.
- Se selectează un număr mai mic de probe și se măsoară diferența de poziție între vârfurile adiacente. Pentru un „1” se numără în jur de 149000 de mostre și 195000 pentru un „0”.
- După această prelucrare se reconstruiește PIN-ul transmis.

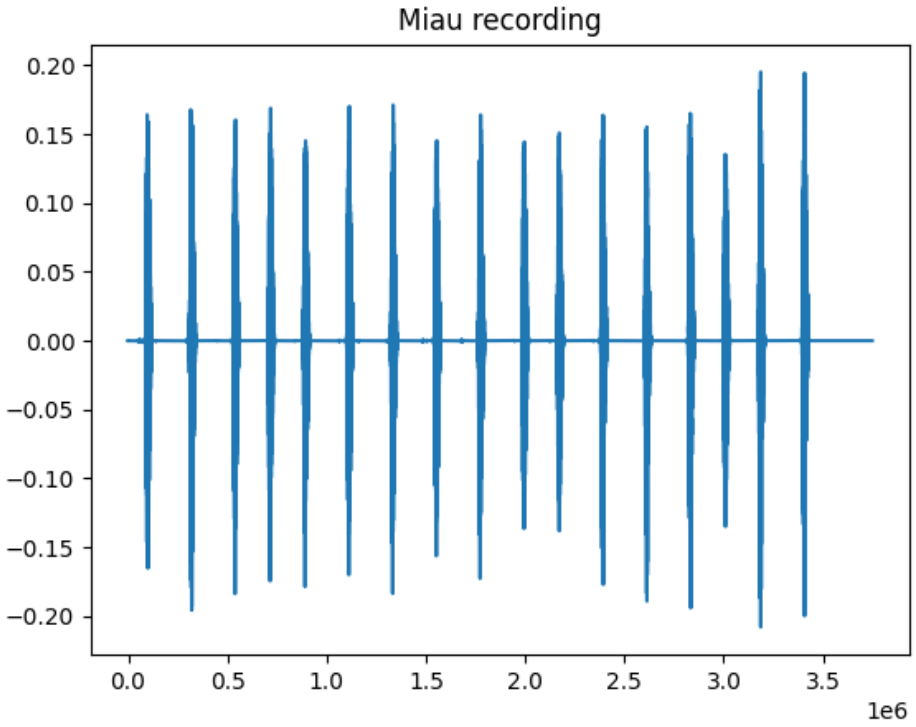


Fig.2 Evidențiere durate pauze pentru PIN-ul $3046_{10} = 0011000001000110_2$

# Interacțiune cu utilizatorul

**Metoda LSB**

- Pentru implementarea cu LSB interfața cu utilizatorul constituie o pagină care permite utilizatorului sa aleagă opțiunea dorită: codificare (Encode) sau decodificare (Decode).
- Pentru cele 2 opțiuni utilizatorul trebuie să aleagă fișierul gazdă al mesajului secret. În cazul în care acesta nu alege un fișier corespunzător (.WAV) va primi un mesaj de eroare. Calea către fișierul ales este afișată pentru ca utilizatorul să verifice dacă a ales fișierul corect.
- În cazul **codificării** este necesară introducerea mesajului care va fi integrat în fișierul audio.
- Dacă nu este introdus un mesaj sau nu este ales un fișier, în ambele cazuri, codificare sau decodificare, se va afișa un mesaj de eroare corespunzător.
- În cazul **decodificării**, mesajul secret se va afișa într-o căsuță text și se va scrie și într-un fișier text.
- Dacă cele două operații se realizează cu succes se va afișa un mesaj corespunzător care va conține și calea către fișierele create (*stegano_audio.wav* pentru codificare și *SecretMessage.txt* pentru decodificare).

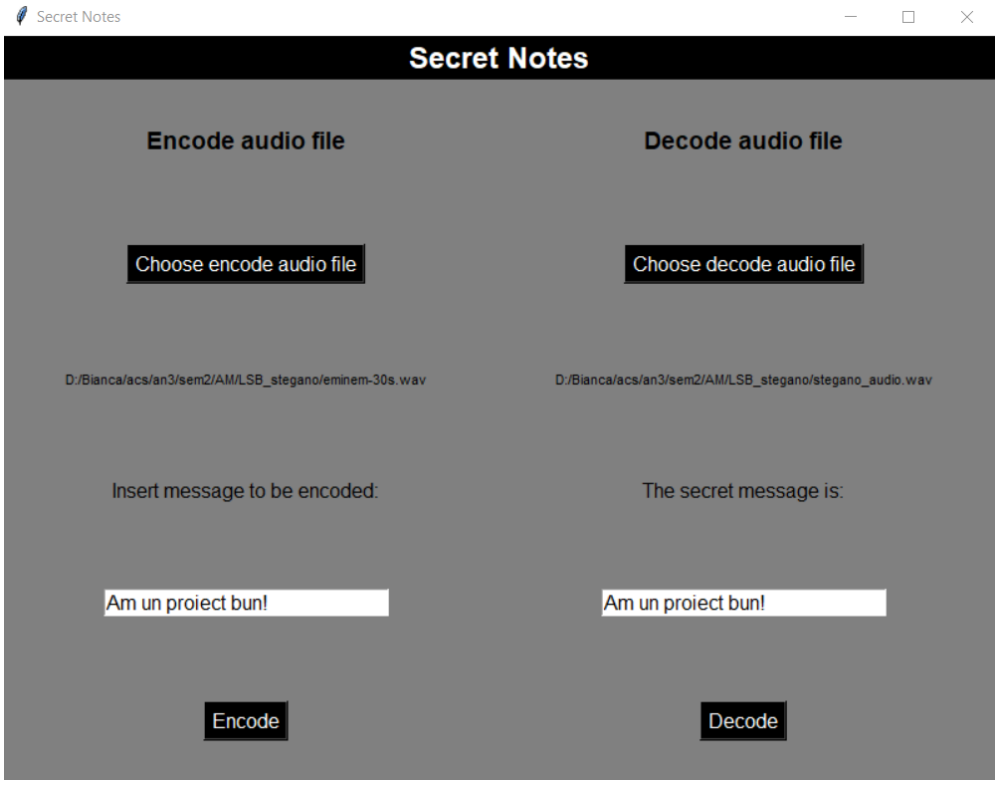


Fig. 3 Interfața cu utilizatorul - LSB

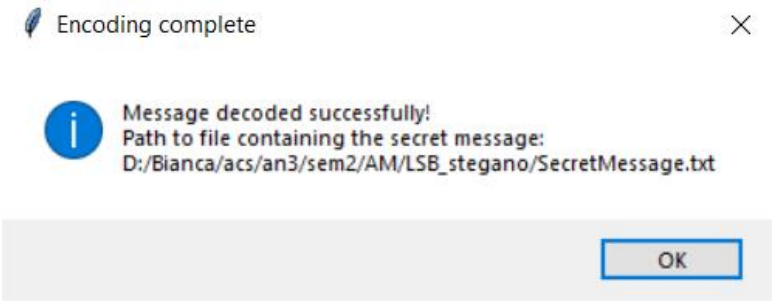


Fig. 4 Mesaj de informare – operație de codificare realizată cu succes

**Metoda Miau**

- Metoda Miau are o interfață care permite introducerea unui cod PIN de 4 cifre, acesta fiind ascuns cu caracterul '*'.
- În cazul în care se introduce un cod necorespunzător, se afișează un mesaj de eroare.
- După apăsarea butonului 'Confirm' sunetul *miau* începe să se audă până când se transmit toți cei 16 biți ai codului PIN introdus.
- La finalizarea acestei acțiuni, sunt afișate codul PIN și reprezentarea lui binară în interfață.

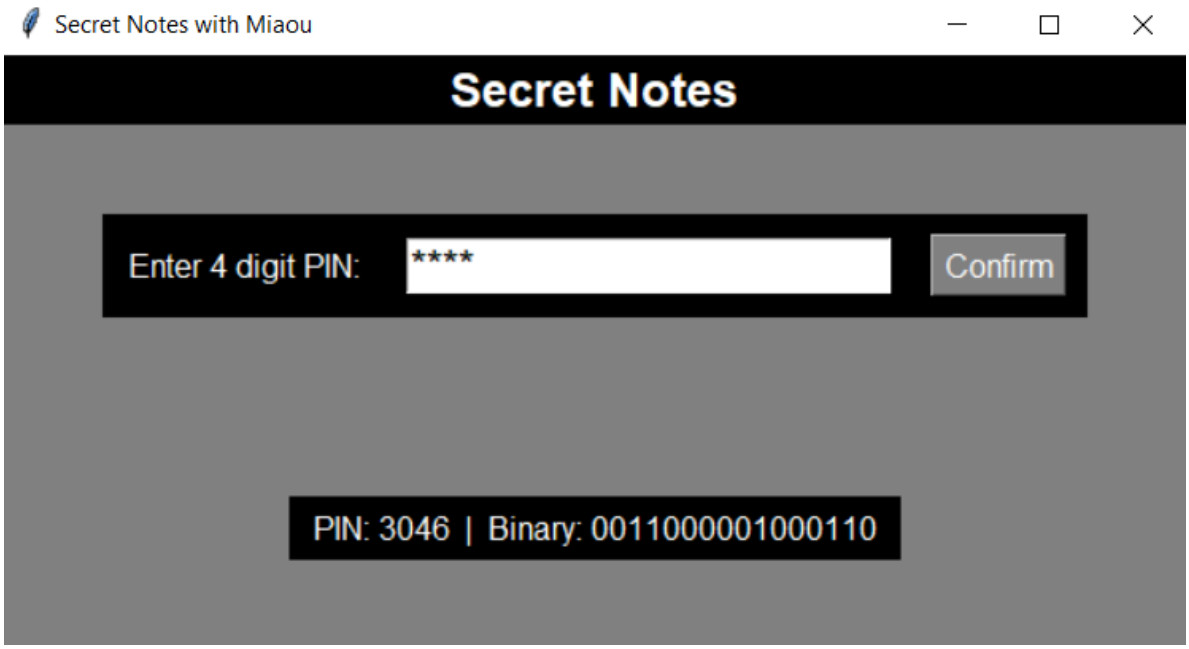


Fig. 5 Interfață cu utilizatorul – Emisie sunet pisică

# Concluzii

Cele 2 metode de steganografie ascund eficient informații. Tehnica LSB este mai bună în cazul în care se transmit mesaje mai lungi, deoarece ascunderea lor în pauzele din redarea unor sunete poate fi un proces de lungă durată.

Steganografia este o metodă utilă de a ascunde informațiile la vedere și de aceea este o tehnică utilizată de atacatori pentru a infecta dispozitivele victimelor.

Steganografia este folosită și pentru watermarking, dar poate avea aplicabilitate și în domeniul militar. [7] [8]

Obiectivele setate au fost atinse prin:

- Crearea unei metode originale de criptare
- Realizarea steganografiei folosind metoda LSB fără ca schimbarea să fie detectată de urechea umană
- Implementarea metodei de ascundere a unui mesaj în pauzele dintre două redări ale unui sunet

# Bibliografie


[1] Sansec Forensics Team, „Payment skimmer hides in social media buttons," https://sansec.io/research/svg-malware, 2020.

[2] S. K. Arora, „Audio Steganography : The art of hiding secrets within earshot (part 2 of 2)," https://sumit-arora.medium.com/, 2018.

[3] A. Ashraf, „AudioSteganography," GitHub, 2020.

[4] Python Software Foundation, „Python libraries," https://docs.python.org/, 2024.

[5] R. H. Seth Kenlon (Team, „Add sound to your Python game," https://opensource.com/, 2020.

[6] spatialaudio.net, „Play and Record Sound with Python," https://python-sounddevice.readthedocs.io/en/0.4.6/, 2019.

[7] Umadevi Ramamoorthy și Aruna Loganathan, „https://www.iajit.org/portal/images/Year2022/No.6/20821.pdf," The International Arab Journal of Information Technology, 2022.

[8] M.Ananthi, R.Bharathi, K. Karthika, N. Sasireka și S. Yogarni, „Secure Sharing Military Information Using Image, Audio and Video With," https://www.riverpublishers.com/.

[9] B. Toulas, „New SteganoAmor attacks use steganography to target 320 orgs globally," https://www.bleepingcomputer.com/news/security/new-steganoamor-attacks-use-steganography-to-target-320-orgs-globally/, 2024.