Bài Thực Hành: Giấu Tin Trong Âm Thanh Bằng Parity Coding

1. Mục tiêu

- Hiểu cấu trúc file âm thanh WAV và cách dữ liệu được biểu diễn dưới dạng bit.
- Nắm rõ nguyên lý của kỹ thuật **Parity Coding** trong steganography.
- Trích xuất thông điệp ẩn từ file WAV đã được nhúng tin..

2. Yêu cầu thực hành

Parity Coding là một kỹ thuật giấu tin dựa trên việc điều chỉnh các bit chẵn lẻ (parity bits) trong dữ liệu âm thanh để mã hóa thông điệp bí mật. Phương pháp này hoạt động bằng cách thay đổi giá trị của bit ít quan trọng nhất (LSB - Least Significant Bit) trong các mẫu âm thanh sao cho tổng số bit "1" trong một nhóm dữ liệu phản ánh giá trị bit của thông điệp cần giấu (chẵn hoặc lẻ). Điều này cho phép nhúng thông tin mà không làm thay đổi đáng kể chất lượng âm thanh, vì thay đổi chỉ xảy ra ở mức bit nhỏ nhất, khó nhận biết bằng tai người. Bài lab sẽ mô phỏng việc kẻ tấn công sử dụng âm thanh để giấu thông tin nhạy cảm và nhà nghiên cứu bảo mật tìm cách phát hiện nó.

Kịch bản

- **Kẻ tấn công**: Giấu một đoạn mã độc hoặc thông điệp chiến lược trong file âm thanh để gửi qua mạng.
- **Nhà nghiên cứu bảo mật**: Phân tích file âm thanh để trích xuất dữ liệu ẩn và ngăn chặn mối đe dọa.

Yêu cầu

- Kiến thức cơ bản về steganography và cách hoạt động của bit chẵn lẻ.

3. Thực hành

Khởi động bài lab: *labtainer stego_code_audio_parity*

Bước 1: Chuẩn bị âm thanh WAV

Giả sử đang làm việc trên cửa sổ của Alice

Chuẩn bị một file âm thanh WAV để thực hành. Nếu bạn chỉ có file ở định dạng khác (ví dụ: MP3), chuyển đổi sang WAV bằng lệnh:

ffmpeg -i sample.mp3 sample.wav

Tạo file chứa thông điệp bí mật bằng lệnh:

echo "root" > secret.txt

Bước 2: Nhúng dữ liệu

Trên cửa sổ của Alice giấu thông điệp bí mật vào metadata sử dụng lệnh:

python3 hide.py

Giải thích ngắn gọn các bước trong code nhúng:

- **Đọc thông điệp**: Đọc từ file secret.txt và thêm dấu hiệu kết thúc <<END>>
- **Chuyển đổi thông điệp**: Biến đổi thông điệp thành chuỗi các bit nhị phân (mỗi ký tự = 8 bit)
- **Đọc file âm thanh**: Đọc samples từ file WAV đầu vào
- **Chia âm thanh thành nhóm**: Chia samples thành các nhóm nhỏ, mỗi nhóm 32 mẫu để mã hóa 1 bit
- **Mã hóa bit bằng parity**:
    - Với mỗi bit cần giấu:
    - Tính tổng chẵn lẻ (parity) của nhóm mẫu (0 nếu có số chẵn giá trị lẻ, 1 nếu có số lẻ)
    - So sánh parity hiện tại với bit cần giấu
    - Nếu khác nhau, chỉ sửa đổi mẫu cuối cùng của nhóm (+1 hoặc -1)
- **Lưu kết quả**: Ghi file âm thanh đã giấu thông điệp ra stego_audio.wav
- **Lưu cấu hình**: Lưu các tham số (kích thước nhóm, độ dài thông điệp) vào file parity_config.npy để phục vụ việc trích xuất sau này

Bước 3: Kiểm tra file dữ liệu được nhúng

So sánh dữ liệu trước và sau khi giấu để xem sự khác biệt về nội dung và dung lượng bằng các lệnh sau:

ls -l sample.wav stego_audio.wav

cmp sample.wav stego_audio.wav

Sau đó thực hiện chuyển video chứa thông điệp đã giấu sang cho Bob

scp *stego_audio.wav* bob:/home/ubuntu/

Password: password123

Bước 4: Trích xuất dữ liệu

Phía Bob thực hiện giải mã

python3 extract.py

Giải thích ngắn gọn các bước trong code trích xuất:

- **Tải cấu hình**: Đọc file parity_config.npy để lấy kích thước nhóm và độ dài thông điệp
- **Đọc file âm thanh**: Mở file WAV đã giấu thông điệp, đọc tất cả các mẫu
- **Trích xuất bit**:
  - Chia audio thành các nhóm mẫu (mỗi nhóm có group_size mẫu)
  - Với mỗi nhóm:
  - Tính tổng chẵn lẻ (parity) của nhóm
  - Giá trị parity (0 hoặc 1) chính là bit đã được giấu
- **Chuyển từ bit sang văn bản**:
  - Gom các bit thành nhóm 8 bit (1 byte)
  - Chuyển mỗi byte thành ký tự ASCII
- **Tìm marker kết thúc**: Tìm chuỗi <<END>> và loại bỏ nó cùng mọi dữ liệu sau đó
- **Trả về thông điệp**: Hiển thị thông điệp đã trích xuất

4. Kết quả cần đạt được

• Hoàn thành các bước: Chuẩn bị file, nhúng tin, chuyển file, trích xuất tin, và xóa metadata.

• Cần nộp 1 file: trong thư mục: /home/student/labtainer_xfer/TÊN_BÀI_LAB (tên tài khoản.*TÊN_BÀI_LAB.lab)*

• Kết thúc bài lab:

o Trên terminal đầu tiên sử dụng câu lênh sau để kết thúc bài lab:

stoplab stego_code_audio_parity

• Khi bài lab kết thúc, một tệp lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.

• Sinh viên cần nộp file *.lab* để chấm điểm.

• Để kiểm tra kết quả khi trong khi làm bài thực hành sử dụng lệnh:

checkwork <tên bài thực hành>

• Khởi động lại bài lab: Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

labtainer –r stego_code_audio_parity