**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA AN TOÀN THÔNG TIN**

****

**HỌC PHẦN: CÁC KỸ THUẬT GIẤU TIN**

**MÃ HỌC PHẦN: INT14102**

**Chủ đề: Giấu tin trong âm thanh**

**Lab: stego\_audio\_dwt\_attack\_noise**

Sinh viên thực hiện: Hoàng Văn Thái

Mã sinh viên: B21DCAT172

Nhóm: 04

Giảng viên hướng dẫn: Đỗ Xuân Chợ

**HÀ NỘI 2025**

**Bài lab Các kỹ thuật giấu tin: stego\_audio\_dwt\_attack\_noise**

1. **Mục đích**

Giúp sinh viên hiểu được cách thức tấn công vào phương pháp giấu tin trong âm thanh sử dụng DWT.

1. **Yêu cầu đối với sinh viên**

Quen thuộc với hệ điều hành Linux và có kiến thức về kỹ thuật giấu tin.

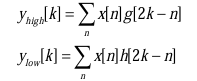
1. **Nội dung lý thuyết**

Biến đổi Wavelet rời rạc (DWT) là một kỹ thuật trong xử lý tín hiệu dùng để phân tách tín hiệu thành các thành phần tần số khác nhau, cho phép phân tích dữ liệu theo cả **miền thời gian** và **miền tần số** đồng thời. Trong phương pháp giấu tin trong âm thanh bằng DWT, thông tin bí mật được nhúng vào thành phần tần số cao (hay còn gọi là Detail Coefficients) của tín hiệu âm thanh.

Quy trình giấu thông tin trong tín hiệu âm thanh bằng phương pháp DWT được tiến hành theo các bước như sau:

* **Bước 1**: Biến đổi DWT

DWT thực hiện phân tích đa phân giải một tín hiệu audio x thành 2 thành phần: thành phần tín hiệu thô A ( approximation) tương ứng với thành phần tần số thấp và thành phần tín hiệu chi tiết D (detail) tương ứng với thành phần tần số cao. Sau đó, thành phần tín hiệu thô tiếp tục được phân tích tương tự. Như vậy, một tín hiệu có thể được biểu diễn dưới dạng tổng của thành phần tín hiệu thô và các thành phần tín hiệu chi tiết. Quá trình phân tích này được thực hiện bởi các bộ lọc băng tần cao và thấp đối với tín hiệu x



Trong đó: yhigh[k]: thành phần tần số cao; ylow[k]: thành phần tần số thấp; x[n]: tín hiệu audio; g[n]: bộ lọc băng tần cao; h[n]: bộ lọc băng tần thấp.

+ **Bước 2**: Xử lý thông điệp bí mật

Xác định độ dài (số bit) của thông điệp bí mật để nhúng chính xác vào âm thanh.

+ **Bước 3:** Nhúng dữ liệu vào hệ số CD

Thay thế hệ số CD đầu tiên bằng kích thước thông điệp bí mật. Điều này giúp cho quá trình trích xuất sau này biết được độ dài chính xác của dữ liệu cần giải mã. Thay thế các hệ số CD tiếp theo bằng thông điệp bí mật đã mã hóa. Điều này đảm bảo dữ liệu được ẩn đi một cách khéo léo mà không làm méo âm thanh quá nhiều.

+ **Bước 4:** Dựng lại tín hiệu âm thanh bằng phép biến đổi DWT ngược (IDWT)

Kết hợp hệ số CA (không bị thay đổi) với hệ số CD đã sửa đổi để tái tạo lại tín hiệu âm thanh. Kết quả là một file âm thanh mới chứa dữ liệu bí mật được gọi là Stego Audio.

Phương pháp thêm nhiễu là một kỹ thuật được sử dụng phổ biến trong lĩnh vực tấn công giấu tin, với mục tiêu phá hủy hoặc làm sai lệch thông tin đã được nhúng trong vật chứa như âm thanh, hình ảnh hoặc video. Kỹ thuật này hoạt động bằng cách chèn thêm một tín hiệu nhiễu – thường là nhiễu trắng (white noise), nhiễu hồng (pink noise), hoặc nhiễu Gauss – vào tín hiệu gốc đã giấu tin. Do dữ liệu ẩn thường được nhúng ở các vị trí nhạy cảm như các bit ít quan trọng (LSB) hoặc hệ số tần số cao trong miền biến đổi (như DWT, DCT), nên việc thêm nhiễu sẽ làm biến dạng hoặc phá vỡ chính xác các giá trị này, khiến việc trích xuất thông điệp trở nên không chính xác hoặc không thể thực hiện được. Phương pháp thêm nhiễu có thể được triển khai bằng các công cụ như SoX, FFmpeg hoặc thông qua lập trình Python

1. **Nội dung thực hành**

Khởi động bài lab:

Vào terminal, gõ:

***labtainer stego\_audio\_dwt\_attack\_noise***

(Chú ý: Sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)

**Nhiệm vụ 1: Giấu tin vào file âm thanh sử dụng DWT**

* Sau khi khởi động xong bài lab, có 1 terminal của Alice
* Trong terminal đã có sẵn một file âm thanh sound.wav, một file python cho phép giấu và tách tin, một file requirements.txt chứa các thư viện cần thiết cho file code
* Sinh viên tiến hành cài đặt các thư viện cần thiết trong file requirements.txt

***pip install -r requirements.txt***

* Sinh viên tiến hành hiển thị thông tin về cách sử dụng công cụ:

***python3 audio\_stego.py -h***

* Sinh viên tiến hành sử dụng tool để giấu thông điệp “PTIT{audio\_dwt\_steganography}” vào file âm thanh sound.wav. Sinh viên sử dụng câu lệnh:

***python3 audio\_stego.py –s “PTIT{audio\_dwt\_steganography}” –i sound.wav –o hidden.wav***

**Nhiệm vụ 2: Giải mã thông điệp**

* Trích xuất thông điệp từ file chứa thông tin ẩn đã tạo ở bước trước đó, ở bước này ta thấy thông điệp vẫn có thể trích xuất bình thường:

***python3 audio\_stego.py -e hidden.wav***

**Nhiệm vụ 3: Tấn công vào file chứa thông điệp bằng cách thêm nhiễu**

* Sử dụng sox để tạo ra một bản sao của hidden.wav nhưng đã bị trộn nhiễu trắng để phá thông tin giấu trong đó. Lệnh sau sẽ tạo nhiễu trắng có thời lượng bằng với file ban đầu, trong đó vol 0.02 là biên độ của tín hiệu nhiễu, mức 0.02 là nhẹ:

***sox -m hidden.wav "|sox -n -r 44100 -p synth $(soxi -D hidden.wav) whitenoise vol 0.02" attacked.wav***

**Nhiệm vụ 4: Trích xuất thông tin từ file đã bị tấn công**

* Sử dụng script python để trích xuất lại thông tin, lần này ta có thể thấy thông tin giấu trong file không thể tách được nữa:

***python3 audio\_stego.py -e attacked.wav***

**Nhiệm vụ 5: So sánh file gốc với file đã bị tấn công**

* Sử dụng sox để phân tích thông tin của 2 file:

***sox hidden.wav -n stat***

***sox attacked.wav -n stat***

Kết quả cho thấy sự khác biệt rõ ràng giữa 2 file, đặc biệt ở các chỉ số thống kê của sóng âm. Sự giảm RMS và biên độ (RMS amplitude và Max amplitude) cho thấy nhiễu đã làm mất tính sắc nét và làm méo nội dung gốc, đúng với mục tiêu phá giải thông tin giấu.

**Kết thúc bài lab:**

Kiểm tra checkwork:

***checkwork***

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

***stoplab***