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_attack_noise_stsm

Sinh viên thực hiện:	Trần Văn Chính
Mã sinh viên:	B21DCAT048
Nhóm:	04
Giảng viên hướng dẫn:	Đỗ Xuân Chợ

HÀ NỘI 2025

Bài lab Các kỹ thuật giấu tin: stego_attack_noise_stsm

1. Mục đích

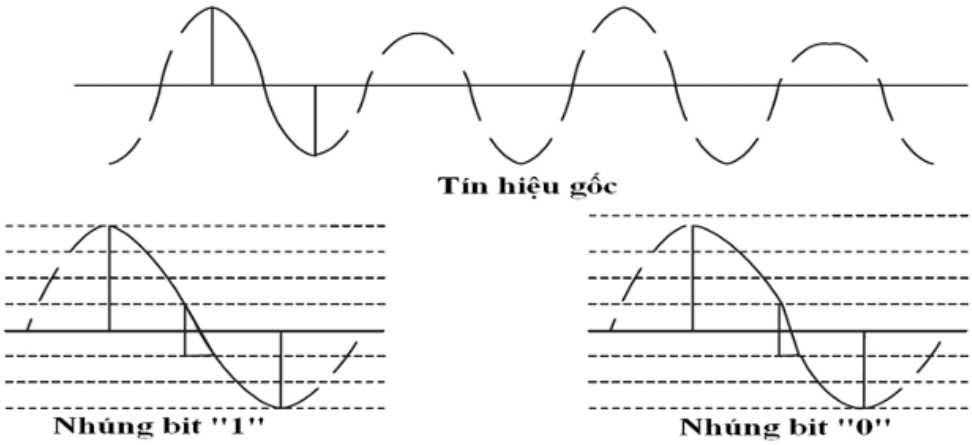
Giúp sinh viên hiểu được cách thức tấn công vào phương pháp giấu tin trong âm thanh sử dụng điều chỉnh tỉ lệ thời gian (Time scale modifications).

2. Yêu cầu đối với sinh viên

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

3. Nội dung lý thuyết

Kỹ thuật giấu tin sử dụng phương pháp điều chỉnh tỉ lệ thời gian (Steganography using Time-Scale Modification -STSM) là một phương pháp giấu tin trong đó thông tin được giấu vào giữa các khoảng thời gian của tín hiệu âm thanh. Cụ thể, thông tin được giấu bằng cách thay đổi khoảng thời gian giữa các mẫu âm thanh. Trong tỉ lệ thời gian điều chỉnh, các mẫu âm thanh được chuyển đổi theo một mẫu được xác định trước, với khoảng thời gian giữa các mẫu được điều chỉnh để giấu thông tin. Việc điều chỉnh khoảng thời gian này có thể được thực hiện bằng cách tăng hoặc giảm khoảng cách giữa các mẫu trong một tập tin âm thanh. Ý tưởng thực hiện là thay đổi tỉ lệ thời gian giữa hai cực là cực đại và cực tiểu. Khoảng cách giữa hai cực được chia thành N phân đoạn có biên độ bằng nhau. Lúc này sẽ thay đổi độ dốc của tín hiệu, tùy thuộc vào bit muốn nhúng.



Hình 1: Quy tắc giấu thông tin sử dụng phương pháp điều chỉnh tỉ lệ thời gian

Quy trình giấu thông tin trong tín hiệu âm thanh bằng phương pháp điều chỉnh tỉ lệ thời gian được tiến hành theo 2 bước như sau:

+ **Bước 1**: Mã hóa: Chuỗi bit M được chia thành các đoạn Mi có độ dài 4 bit. Mỗi đoạn bit thông tin này sẽ được mã hoá bằng phương pháp mã Hamming. Với phương pháp mã hóa Hamming thì các đoạn Mi được biến đổi từ 4 bit

thành từ mã có độ dài 7 bit. Ghép các chuỗi bit kết quả lại để được chuỗi bit M'. Độ dài chuỗi M' sẽ bằng (L/4) x7.

+ **Bước 2**: Giấu tin: Trước tiên vật chứa C sẽ được xử lý để trích phần header và phần dữ liệu. Người giấu tin sẽ tiến hành kiểm tra vật chứa C có đủ để giấu chuỗi bit M' không. Nếu không đủ thì dừng và báo không giấu được. Nếu đủ thì sẽ ghi header của C vào C' sau đó thực hiện giấu từng bit của chuỗi M' vào phần dữ liệu của C để ghi ra C'. Sau quá trình kiểm tra thỏa mãn thì tiến hành trích tuần tự 3 mẫu dữ liệu của C và tính tổng:
  - Nếu bit đang xét của M' là 1 mà tổng lẻ thì thỏa mãn điều kiện giấu, không cần điều chỉnh. Nếu tổng chẵn thì điều chỉnh mẫu số 2 của 3 mẫu đang xét để cho tổng là số lẻ.
  - Nếu bit đang xét của M' là 0 mà tổng chẵn thì đã thỏa mãn điều kiện giấu, không cần điều chỉnh, ngược lại điều chỉnh mẫu 1 hoặc mẫu thứ 3 trong 3 mẫu đang xét để cho tổng là số chẵn.
  - Ghi 3 mẫu đang xét ra tệp C'. Lặp lại quá trình kiểm tra trên cho đến khi toàn bộ các bit của chuỗi M' đã được giấu.

    Cuối cùng là công đoạn ghi các mẫu còn lại từ C vào C' và kết thúc.

Tấn công vào kỹ thuật giấu tin STSM bằng cách chèn nhiễu (noise attack) nhằm mục tiêu làm rối loạn cấu trúc tín hiệu âm thanh đã giấu tin, khiến cho tiến trình giải mã không thể thu được thông điệp ban đầu. Trong kỹ thuật STSM đã sử dụng, dữ liệu nhúng được mã hóa bằng mã Hamming và nhúng vào mẫu âm thanh theo từng cụm 3 mẫu dựa trên tính chẵn/lẻ của tổng các mẫu. Tuy nhiên, phương pháp này phụ thuộc chặt chẽ vào giá trị cụ thể và vị trí chính xác của từng mẫu trong chuỗi tín hiệu. Khi kẻ tấn công chèn một lượng nhiễu nhỏ (thường là Gaussian noise hoặc uniform noise mức thấp) vào toàn bộ tín hiệu, các mẫu này bị thay đổi giá trị một cách ngẫu nhiên nhưng đủ để phá vỡ tính đồng nhất của tổng 3 mẫu, dẫn đến lỗi bit khi giải mã. Do STSM không có cơ chế kiểm tra lỗi vượt quá khả năng sửa lỗi của mã Hamming, một khi quá nhiều bit bị sai lệch, toàn bộ chuỗi thông điệp sẽ trở nên không thể khôi phục, kể cả khi thuật toán và khóa nhúng là chính xác. Đây là kiểu tấn công thuộc loại phá huỷ tín hiệu giấu (destructive stego attack), rất hiệu quả với các kỹ thuật giấu phụ thuộc trực tiếp vào độ chính xác của mẫu âm thanh trong miền thời gian như STSM.

4. Nội dung thực hành

Khởi động bài lab:

Vào terminal, gõ:

labtainer stego_attack_noise_stsm

(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 STSM

- Sau khi khởi động xong bài lab, có 1 terminal của Alice
- Trong terminal Alice, đã có sẵn một file âm thanh input.wav, một file dữ liệu secret.txt, một tool dùng để giấu tin vào trong file âm thanh và một code python dùng để tấn công noise vào kỹ thuật giấu tin STSM
- Sinh viên tiến hành hiển thị thông tin về cách sử dụng công cụ:

python3 stsm_tool.py -h

- Sinh viên tiến hành sử dụng tool để giấu tin trong file secret.txt vào file âm thanh input.wav. Sinh viên sử dụng câu lệnh:

python3 stsm_tool.py –encode –i input.wav –m secret.txt

Nhiệm vụ 2: Tấn công noise vào file âm thanh đã được giấu tin sử dụng STSM

- Sinh viên tiến hành hiển thị thông tin về cách sử dụng công cụ tấn công noise vào STSM:

python3 attack_noise.py -h

- Thực hiện tấn công noise vào file âm thanh đã được giấu bằng câu lệnh:

python3 attack_noise.py -input output.wav -output attacked_noise.wav -level 0.001

Nhiệm vụ 3: Giải mã thông tin trong file âm thanh sử dụng STSM đã bị tấn công

- Sau khi thực hiện tấn công vào file âm thanh đã giấu tin, sinh viên thực hiện giải mã thông tin đã được giấu trong file này:

python3 stsm_tool.py –decode –i attacked_noise.wav

- Sinh viên sẽ bị lỗi khi thực hiện trích xuất thông tin được giấu trong file âm thanh này, chứng tỏ cuộc tấn công noise vào STSM đã thực hiện thành công

Nhiệm vụ 4: So sánh file âm thanh gốc với file âm thanh đã bị tấn công noise vào phương pháp giấu tin bằng STSM

- Mục đích của nhiệm vụ này là tìm hiểu sự khác biệt giữa 2 file wav trước và sau tấn công. Tại terminal của Alice, sử dụng công cụ soxi để xem thông tin kỹ thuật của 2 file âm thanh

soxi input.wav

soxi attacked_noise.wav

- Kết quả cho thấy 2 file đã có sự khác biệt thông số kỹ thuật. Tuy nhiên, điều đó không có nghĩa là nội dung âm thanh giống nhau, do đó cần so sánh raw data của 2 file âm thanh. Chúng ta thử xác định xem raw data của 2 file wav này khác nhau bắt đầu từ byte nào. Sử dụng lệnh

  cmp -l <(sox input.wav -t .raw -) <(sox attacked_noise.wav -t .raw -)

- Kết quả cho thấy ký tự thứ 5 của dòng 1 là nơi đầu tiên xảy ra sự khác biệt. Như vậy, mặc dù 2 file có cùng định dạng và thông số kỹ thuật nhưng nội dung 2 file khác nhau hoàn toàn.

Kết thúc bài lab:

Kiểm tra checkwork:

checkwork

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

stoplab