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

**KHOA AN TOÀN THÔNG TIN**

****

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN: CÁC KỸ THUẬT GIẤU TIN**

**MÃ HỌC PHẦN: INT14102**

**CHỦ ĐỀ: THỰC HIỆN TẤN CÔNG VÀO TỆP ÂM THANH ĐƯỢC GIẤU TIN BẰNG PHƯƠNG PHÁP FHSS**

**Lab: attack\_fhss**

| Sinh viên thực hiện | Đặng Minh Hiếu |
| --- | --- |
| Mã sinh viên | B21DCAT087 |

Tên nhóm: Nhóm 10

Tên lớp: Nhóm 3

Giảng viên hướng dẫn: TS. Đỗ Xuân Chợ

**HÀ NỘI 2025**

# Bài lab : Thực hiện tấn công vào tệp âm thanh được giấu tin bằng phương pháp FHSS

## Mục đích

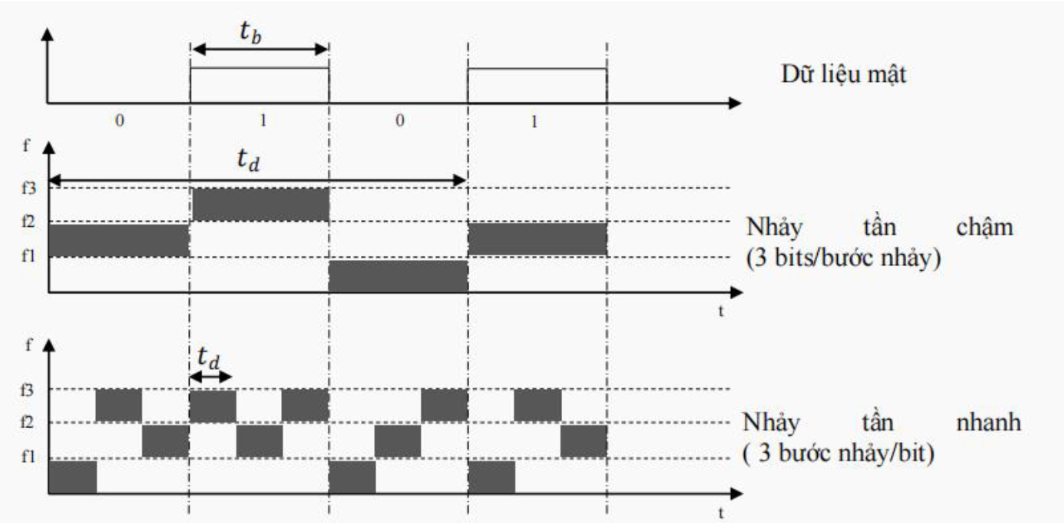
* Giúp sinh viên hiểu được thuật toán giấu tin và tách tin trong âm thanh sử dụng phương pháp FHSS
* Sinh viên làm quen với các loại tấn công như tấn công tệp âm thanh bằng cách thêm nhiễu, tấn công tệp âm thanh bằng cách lọc tần số, tấn công tệp âm thanh bằng cách nén bằng Mp3

## Yêu cầu đối với sinh viên

* Hiểu biết về các loại nhiễu thường gặp trong âm thanh (ví dụ: nhiễu trắng cộng Gauss - AWGN) và cách chúng ảnh hưởng đến tín hiệu.
* Nắm vững lý thuyết về giấu tin (Steganography) và các phương pháp tấn công tin giấu (Steganalysis).
* Hiểu rõ nguyên lý hoạt động của kỹ thuật trải phổ nhảy tần (FHSS) và cách ứng dụng nó trong giấu tin âm thanh để tăng tính bền vững.
* Nắm được nguyên lý của các bộ lọc tần số và tác động của chúng lên phổ tín hiệu âm thanh.
* Hiểu rõ cơ chế hoạt động của chuẩn nén mất dữ liệu MP3, các yếu tố ảnh hưởng đến chất lượng âm thanh và khả năng phá hủy thông tin ẩn khi nén.

## Nội dung lý thuyết

Trải phổ nhảy tần là một công nghệ sử dụng bộ phát tần số và có thể thay đổi tần số truyền một cách đột ngột trong dãy băng tần sử dụng. Trong trải phổ nhảy tần, độ rộng băng thông sẵn có sẽ được chia thành một số lớn các khe tần không lấn lên nhau. Tại bất kì khoảng thời gian nào, tín hiệu truyền đi đều chiếm một hay nhiều hơn một khe tần số nói trên. Việc chọn một khe hay nhiều khe tần số trong một khoảng thời gian truyền tín hiệu đều được thực hiện một cách giả ngẫu nhiên theo tín hiệu ra của một bộ tạo chuỗi giả ngẫu nhiên. Dựa trên tốc độ nhảy của tần số thì phương pháp trải phổ nhảy tần được chia làm 2 loại đó là trải phổ nhảy tần nhanh và trải phổ nhảy tầm chậm. Trong đó, trải phổ nhảy tần nhanh có đặc điểm là tốc độ nhảy nhanh hơn tốc độ dữ liệu. Còn trải phổ nhảy tần chậm thì có tốc độ nhảy chậm hơn tốc độ dữ liệu. Hình 3.17 mô tả ví dụ của trải phổ nhảy tần nhanh và trải phổ nhảy tần chậm.



#### Quy trình Giấu tin (Phía Máy phát):

***Bước 1: Chuẩn bị thông tin và Mã hóa (Tùy chọn):***

* Thông tin cần giấu (Chuỗi thông tin mật) được đưa vào.
* Thông tin này có thể được đưa qua **Bộ mã hóa**, nơi nó có thể được mã hóa bằng một khóa riêng để tăng tính bảo mật. Bước này là tùy chọn.

***Bước 2: Điều chế FSK:***

* Tín hiệu (đã mã hóa hoặc chưa) được đưa vào **Bộ điều chế FSK (Frequency Shift Keying)**.
* Bộ điều chế FSK chuyển đổi thông tin thành tín hiệu nhị phân x(t), trong đó mỗi bit (0 hoặc 1) tương ứng với một trong hai tần số cụ thể (ví dụ: f′ cho bit 0 và f′+Δf cho bit 1).

***Bước 3: Tạo chuỗi nhảy tần:***

* Một **Bộ tạo chuỗi giả ngẫu nhiên (PN - Pseudorandom Number)** tạo ra một đoạn mã (ví dụ: m bit).
* Đoạn mã này điều khiển một **Bộ tổ hợp tần số (Frequency Synthesizer)** để tạo ra các giá trị tần số nhảy tần y(t) cho sóng mang. Tần số y(t) thay đổi sau mỗi khoảng thời gian Tc​ giây theo giá trị m bit từ bộ tạo chuỗi PN.

***Bước 4: Trộn tín hiệu:***

* Tín hiệu x(t) (từ bộ điều chế FSK) và tín hiệu tần số nhảy y(t) (từ bộ tổ hợp tần số) được đưa vào một **Bộ trộn (Mixer)**.
* Bộ trộn kết hợp hai tín hiệu này, tạo ra các tần số tổng và hiệu. Một trong hai tần số này sẽ được lọc ra (thường bằng bộ lọc băng thông BPF, không thể hiện rõ trong sơ đồ nhưng là một phần của quá trình trộn chuẩn) trước khi đưa vào môi trường chứa (ví dụ: file âm thanh gốc).
* Về bản chất, thông tin mật sẽ được "trải" trên nhiều tần số khác nhau theo một trình tự giả ngẫu nhiên.

#### Quy trình Tách tin (Phía Máy thu):

***Bước 1: Đồng bộ hóa:***

* Đây là bước cực kỳ quan trọng. **Bộ đồng bộ định thời** đảm bảo rằng bộ tạo chuỗi PN ở phía máy thu hoạt động đồng bộ với phía máy phát.

***Bước 2: Tái tạo chuỗi nhảy tần:***

* **Bộ tạo chuỗi PN** ở máy thu (đã được đồng bộ và sử dụng cùng khóa/hạt giống như máy phát) tạo ra lại cùng một chuỗi giả ngẫu nhiên.
* Chuỗi này điều khiển **Bộ tổ hợp tần số** để tạo ra chính xác cùng một chuỗi tần số nhảy y(t) đã được sử dụng ở máy phát.

***Bước 3: Giải trải phổ (De-hopping):***

* Tín hiệu thu được từ môi trường (file âm thanh chứa tin) được đưa vào **Bộ trộn**.
* Tại đây, tín hiệu thu được sẽ được trộn với tín hiệu tần số nhảy y(t) (tái tạo ở bước 2). Nếu đồng bộ chính xác, quá trình này sẽ loại bỏ thành phần nhảy tần, khôi phục lại tín hiệu FSK x(t) ban đầu (sau khi qua bộ lọc BPF thích hợp).

***Bước 4: Giải điều chế FSK:***

* Tín hiệu x(t) được đưa vào **Bộ giải điều chế FSK**.
* Bộ giải điều chế FSK sẽ chuyển đổi tín hiệu tần số trở lại thành chuỗi bit dữ liệu (đã mã hóa nếu có).

***Bước 5: Giải mã (Tùy chọn):***

* Nếu thông tin đã được mã hóa ở phía máy phát, chuỗi bit dữ liệu thu được sẽ được đưa qua **Bộ giải mã** (sử dụng cùng khóa riêng) để khôi phục lại thông tin mật ban đầu.

#### Tấn công thêm nhiễu vào âm thanh giấu tin bằng FHSS

* ***Cơ chế tác động:***
  + FHSS chống nhiễu băng hẹp tốt, nhưng vẫn bị ảnh hưởng bởi nhiễu băng rộng: Nếu nhiễu chỉ tồn tại ở một vài tần số cố định, FHSS sẽ "nhảy" qua các tần số đó và ít bị ảnh hưởng. Tuy nhiên, nếu nhiễu là nhiễu băng rộng (trải khắp dải tần mà FHSS sử dụng), nó sẽ làm tăng mức nhiễu nền chung trên tất cả các kênh tần số mà tín hiệu nhảy tới.
  + Giảm tỷ số tín hiệu trên nhiễu (SNR) tại mỗi bước nhảy: Tại mỗi tần số nhảy, máy thu cần phải phát hiện và giải điều chế chính xác tín hiệu FSK (hoặc dạng điều chế khác) mang thông tin. Nhiễu băng rộng làm giảm SNR tại từng tần số cụ thể này, khiến việc phân biệt giữa các ký hiệu FSK (ví dụ: tần số cho bit '0' và tần số cho bit '1') trở nên khó khăn hơn, dẫn đến lỗi bit khi trích xuất.
  + Gây khó khăn cho đồng bộ hóa: Quá trình đồng bộ hóa của máy thu FHSS để bắt kịp chuỗi nhảy tần của máy phát có thể bị cản trở nghiêm trọng nếu mức nhiễu băng rộng quá cao, vì tín hiệu trở nên quá yếu so với nhiễu.
* **Yếu tố quyết định hiệu quả tấn công đối với FHSS:**
  + Loại nhiễu và công suất nhiễu: Nhiễu băng rộng (ví dụ: nhiễu trắng cộng Gauss - AWGN) bao trùm toàn bộ dải tần hoạt động của FHSS sẽ hiệu quả hơn nhiễu băng hẹp. Công suất nhiễu càng lớn so với công suất tín hiệu FHSS tại mỗi bước nhảy thì khả năng gây lỗi càng cao.
  + Độ lợi xử lý (Processing Gain) của hệ thống FHSS: Hệ thống FHSS có độ lợi xử lý càng cao (ví dụ, sử dụng dải tần tổng cộng càng rộng, số lượng kênh nhảy tần càng nhiều) thì khả năng chống nhiễu càng tốt. Kẻ tấn công cần nhiễu đủ mạnh để vượt qua độ lợi này.
  + Kỹ thuật điều chế và mã hóa sửa lỗi: Loại điều chế (ví dụ FSK) và sự có mặt của mã hóa sửa lỗi trong thông điệp gốc sẽ ảnh hưởng đến khả năng chịu đựng nhiễu của hệ thống.

#### 

#### 

#### Tấn công Lọc tần số vào âm thanh giấu tin bằng FHSS

* **Cơ chế tác động lên FHSS**
  + Ít hiệu quả với lọc băng hẹp: Do tín hiệu FHSS liên tục nhảy tần, việc lọc bỏ một vài tần số hoặc một dải tần hẹp chỉ làm mất đi một phần nhỏ thông tin được truyền khi tín hiệu nhảy vào đúng các tần số đó. Hệ thống vẫn có thể hoạt động được với các tần số còn lại
  + Hiệu quả hơn với lọc băng rộng hoặc lọc chắn dải (Notch filtering) mạnh mẽ:
    - Nếu bộ lọc loại bỏ một phần đáng kể của tổng băng thông mà FHSS sử dụng (ví dụ, một bộ lọc thông thấp hoặc thông cao rất hạn chế), nó sẽ làm giảm nghiêm trọng số lượng kênh tần số khả dụng cho việc nhảy tần. Điều này làm giảm độ lợi xử lý và khiến hệ thống dễ bị tổn thương hơn.
    - Nếu kẻ tấn công có thể xác định và lọc bỏ đồng thời nhiều dải tần số quan trọng trong chuỗi nhảy (khó thực hiện nếu chuỗi PN tốt), hiệu quả tấn công sẽ tăng.
  + Mất mát thông tin: Mỗi khi tín hiệu nhảy vào một tần số đã bị lọc bỏ, các bit thông tin được truyền tại thời điểm đó sẽ bị mất. Nếu số lượng tần số bị lọc lớn, tỷ lệ mất mát thông tin có thể vượt quá khả năng sửa lỗi của hệ thống.
* **Yếu tố quyết định hiệu quả tấn công đối với FHSS:**
  + Độ rộng của dải tần bị lọc so với tổng băng thông FHSS: Tỷ lệ này càng lớn, tấn công càng hiệu quả.
  + Số lượng kênh tần số khả dụng của FHSS: Càng nhiều kênh, hệ thống càng linh hoạt và ít bị ảnh hưởng bởi việc mất một số kênh.
  + Đặc điểm của chuỗi nhảy tần (PN sequence): Nếu chuỗi PN có đặc tính phân bố tần số không đồng đều hoặc có thể dự đoán, kẻ tấn công có thể nhắm vào các tần số được sử dụng thường xuyên hơn.
  + Mã hóa sửa lỗi: Khả năng phục hồi thông tin bị mất do lọc phụ thuộc vào hiệu quả của mã sửa lỗi.

#### Tấn công Nén MP3 vào âm thanh giấu tin bằng FHSS

* **Cơ chế tác động lên FHSS**
  + Lượng tử hóa các thành phần tần số: Nén MP3 dựa trên việc biến đổi tín hiệu sang miền tần số (ví dụ: MDCT) và sau đó lượng tử hóa các hệ số tần số. Quá trình lượng tử hóa này, đặc biệt ở các bitrate thấp, sẽ loại bỏ hoặc làm giảm độ chính xác của nhiều thành phần tần số.
  + Ảnh hưởng đến tín hiệu FSK tại mỗi bước nhảy: Tại mỗi thời điểm, tín hiệu FHSS mang thông tin (ví dụ: một tần số cụ thể của FSK) chiếm một băng tần hẹp. Khi tín hiệu âm thanh được chia thành các khung để xử lý bởi MP3, các thành phần tần số của tín hiệu FSK này có thể bị:
    - Loại bỏ hoặc làm yếu đi: Nếu năng lượng của chúng thấp hoặc rơi vào vùng mà mô hình tâm lý âm học của MP3 cho là ít quan trọng.
    - Biến dạng: Quá trình lượng tử hóa có thể làm thay đổi tần số hoặc biên độ của tín hiệu FSK đủ để gây lỗi giải điều chế.
  + Phá vỡ cấu trúc tín hiệu tức thời: Mặc dù FHSS trải rộng năng lượng theo thời gian và trên nhiều tần số, nhưng tại mỗi "điểm" trong thời gian và tần số, nó vẫn là một tín hiệu có cấu trúc cụ thể (ví dụ, một tone FSK). Nén MP3 có thể phá vỡ cấu trúc vi mô này.
* **Yếu tố quyết định hiệu quả tấn công đối với FHSS:**
  + Bitrate của MP3: Bitrate càng thấp, mức độ nén càng mạnh, lượng tử hóa càng thô, và do đó khả năng phá hủy tín hiệu FHSS càng cao.
  + Năng lượng của tín hiệu FHSS nhúng: Nếu tín hiệu FHSS được nhúng với biên độ rất nhỏ để đảm bảo tính vô hình (ví dụ, hệ số α nhỏ trong công thức x(n)=s(n)+αw(n) ), các thành phần tần số của nó càng dễ bị loại bỏ trong quá trình nén MP3.
  + Đặc điểm của tín hiệu âm thanh gốc: Mô hình tâm lý âm học của MP3 sẽ quyết định giữ lại những thành phần nào của âm thanh. Nếu tín hiệu FHSS nằm trong các vùng phổ mà MP3 có xu hướng loại bỏ mạnh, nó sẽ bị ảnh hưởng nhiều hơn.
  + Sự phức tạp của bộ mã hóa MP3: Các bộ mã hóa MP3 khác nhau có thể có những khác biệt nhỏ trong thuật toán, dẫn đến tác động khác nhau.

## Nội dung thực hành

Khởi động bài lab:

* Vào terminal, gõ lệnh:

rebuild attack\_fhss

### Nhiệm vụ 1: Tách tin trong âm thanh gốc và đọc thông tin được giấu

Thực hiện tiền xử lý file âm thanh có giấu tin, ta thu được frequency sample, dữ liệu biên độ chuyển đổi sang kiểu số thực của file âm thanh được có tin được giấu và file âm thanh gốc. Đây là bước trước khi ta tách tin để khi thực hiện tách tin, script tách tin sẽ có thể đọc từ file text dễ dàng hơn đọc từ file wav.

Ta có cú pháp sử dụng script python tienxuly: “python3 tienxuly.py <file wav chứa tin> <file txt chứa biên độ của file âm thanh giấu tin> <file txt chứa biên độ của file âm thanh gốc> ” Ta sử dụng lệnh:

***python3 tienxuly.py output.wav embedded\_origin.txt original\_origin.txt fs\_tach\_origin.txt***

Sau đó ta thực hiện tách tin bằng dòng lệnh: “python3 tachtin.py <file txt chứa biên độ của file âm thanh giấu tin> <file txt chứa biên độ của file âm thanh gốc> <file txt chứa frequency sample của âm thanh> <file txt chứa thông tin được giải mã>”

***python3 tachtin.py embedded\_origin.txt original\_origin.txt fs\_tach\_origin.txt recovered\_message\_origin.txt***

Tiếp theo, ta đọc thông điệp sau khi tách tin để chắc chắn rằng tin không bị lỗi và kiểm tra xem các phương thức tấn công đằng sau có hiệu quả không:

***cat recovered\_message\_origin.txt***

### Nhiệm vụ 2: Thực hiện tấn công bằng phương pháp thêm nhiễu và đọc thông tin được giấu sau khi tấn công

Để thực hiện tấn công bằng phương pháp thêm nhiễu, ta sử dụng lệnh

***python3 adding\_noise.py***

Sau đó ta thực hiện lại các bước tiền xử lý như ở trên đã nêu, thay <file wav chứa tin> → <file wav chứa tin sau khi tấn công thêm nhiễu>, <file txt chứa biên độ của file âm thanh giấu tin> → <file txt chứa biên độ của file âm thanh giấu tin sau tấn công thêm nhiễu>, <file txt chứa biên độ của file âm thanh gốc> → <file txt chứa biên độ của file âm thanh gốc sau khi bị tấn công thêm nhiễu>:

***python3 tienxuly.py attacked\_noise.wav embedded\_noise.txt original\_noise.txt fs\_tach\_noise.txt***

Và thực hiện tách tin:

***python3 tachtin.py embedded\_noise.txt original\_noise.txt fs\_tach\_noise.txt recovered\_message\_noise.txt***

Sau đó đọc file thông tin bị tấn công thêm nhiễu sau khi tách tin và thu được thông tin khác với thông tin gốc được tách:

***cat recovered\_message\_noise.txt***

### 

### 

### Nhiệm vụ 3: Thực hiện tấn công bằng phương pháp lọc tần số và đọc thông tin được giấu sau khi tấn công

Để thực hiện tấn công bằng phương pháp lọc tần số, ta sử dụng lệnh

***python3 filtering.py***

Sau đó ta thực hiện lại các bước tiền xử lý như ở trên đã nêu, thay <file wav chứa tin> → <file wav chứa tin sau khi tấn công lọc tần số>, <file txt chứa biên độ của file âm thanh giấu tin> → <file txt chứa biên độ của file âm thanh giấu tin sau tấn công lọc tần số>, <file txt chứa biên độ của file âm thanh gốc> → <file txt chứa biên độ của file âm thanh gốc sau khi bị tấn công lọc tần số>:

***python3 tienxuly.py attacked\_filtered.wav embedded\_filtered.txt original\_filtered.txt fs\_tach\_filtered.txt***

Và thực hiện tách tin:

***python3 tachtin.py embedded\_filtered.txt original\_filtered.txt fs\_tach\_filtered.txt recovered\_message\_filtered.txt***

Sau đó đọc file thông tin bị tấn công lọc tần số sau khi tách tin và thu được thông tin khác với thông tin gốc được tách:

***cat recovered\_message\_filtered.txt***

### Nhiệm vụ 4: Thực hiện tấn công bằng phương pháp nén file Mp3 và đọc thông tin được giấu sau khi tấn công

Để thực hiện tấn công bằng phương pháp lọc tần số, ta sử dụng lệnh

***python3 loss\_compression\_mp3.py***

Sau đó ta thực hiện lại các bước tiền xử lý như ở trên đã nêu, thay <file wav chứa tin> → <file wav chứa tin sau khi tấn công nén file mp3>, <file txt chứa biên độ của file âm thanh giấu tin> → <file txt chứa biên độ của file âm thanh giấu tin sau tấn công nén file mp3>, <file txt chứa biên độ của file âm thanh gốc> → <file txt chứa biên độ của file âm thanh gốc sau khi bị tấn công nén file mp3>:

***python3 tienxuly.py attacked\_mp3.wav embedded\_loss\_compression\_mp3.txt original\_loss\_compression\_mp3.txt fs\_tach\_loss\_compression\_mp3.txt***

Và thực hiện tách tin:

***python3 tachtin.py embedded\_loss\_compression\_mp3.txt original\_loss\_compression\_mp3.txt fs\_tach\_loss\_compression\_mp3.txt recovered\_message\_loss\_compression\_mp3.txt***

Sau đó đọc file thông tin bị tấn công lọc tần số sau khi tách tin và thu được thông tin khác với thông tin gốc được tách:

***cat recovered\_message\_loss\_compression\_mp3.txt***

## Kết quả cần đạt được

* Chạy được tất cả các bước như yêu cầu.

## Kết thúc bài lab:

Sử dụng lệnh: “checkwork” để kiểm tra số checkwork đã làm đúng.

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

***stoplab attack\_fhss***

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

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

***checkwork attack\_fhss***

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

***labtainer –r attack\_fhss***