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: fhss-encode

Sinh viên thực hiện:	Nguyễn Trường Sơn
Mã sinh viên:	B21DCAT164
Nhóm:	04
Giảng viên hướng dẫn:	Đỗ Xuân Chợ

HÀ NỘI 2025

Bài lab Các kỹ thuật giấu tin: fhss-encode

1. Mục đích

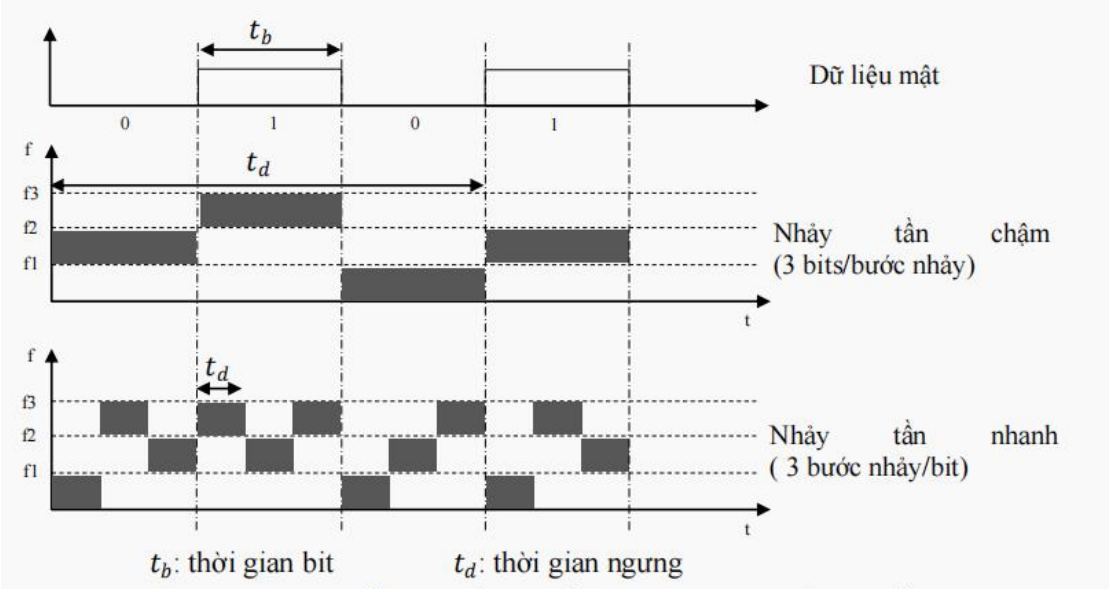
Giúp sinh viên hiểu được thuật toán giấu tin trong âm thanh sử dụng phương pháp trải phổ nhảy tần FHSS, cụ thể là quy trình giấu tin của phương pháp này.

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

Trải phổ nhảy tần (FHSS) là công nghệ thay đổi tần số truyền đột ngột trong một dãy băng tần. Băng thông được chia thành nhiều khe tần không lấn nhau, và tín hiệu truyền đi có thể chiếm một hoặc nhiều khe tần, việc chọn khe tần được thực hiện theo chuỗi giả ngẫu nhiên. Dựa vào tốc độ nhảy tần, có hai loại: nhảy tần nhanh (tốc độ nhảy nhanh hơn tốc độ dữ liệu) và nhảy tần chậm (tốc độ nhảy chậm hơn tốc độ dữ liệu). Bài thực hành này thực hiện theo phương pháp nhảy tần chậm.



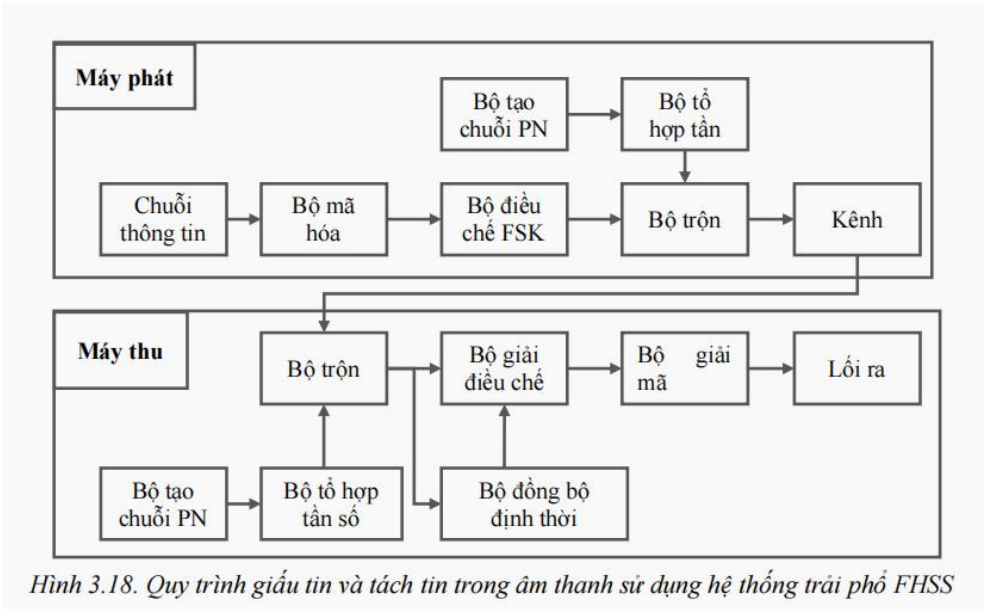
Hình 3.17. Minh hoạ về trải phổ nhảy tần nhanh và trải phổ nhảy tần chậm

Quy trình giấu tin:

- Quy trình giấu tin của hệ thống trải phổ FHSS nhận dữ liệu vào gồm 2 thành phần chính là thông tin cần giấu và bộ chuỗi giả ngẫu nhiên. Trong đó, thông tin cần giấu được đưa vào Bộ mã hóa. Tại đây, tín hiệu được mã hóa bằng khóa riêng trước khi được đưa vào Bộ điều chế. Đây là bước tùy chọn, nghĩa là tùy người gửi tin cài đặt cho máy phát lựa chọn có mã hóa hay không, nếu có thì chọn kỹ thuật mã hóa nào. Tín hiệu sau khi được mã hóa sẽ được đưa vào bộ điều chế FSK (điều chế số theo tần số tín hiệu) để

điều chế thành tín hiệu nhị phân x(t) có 1 trong 2 tần số tương ứng với bit 0 và 1. Bộ điều chế FSK sẽ chọn một trong hai tần số tương ứng với việc truyền đi bit dữ liệu 0 hay 1.

- Bộ trộn tín hiệu nhận cả thông tin cần giấu và chuỗi giả ngẫu nhiên. Chuỗi giả ngẫu nhiên (PN) giúp chọn tần số để nhảy trong quá trình truyền bằng cách điều khiển bộ tổ hợp tần. Quá trình này thực hiện bằng cách gửi 1 đoạn mã m bit đến bộ tổ hợp tần tại các thời điểm nhảy tần. Khi danh sách tần số đã nhảy hết, máy phát sẽ lặp lại từ đầu.
- Tại bộ tổ hợp tần:
  - Nhận tín hiệu điều khiển từ PN, tạo ra các giá trị tần số nhảy tần cho sóng mang.
  - Nhảy sang hoạt động ở 1 tần số tương ứng với đoạn mã m bit nhận được từ PN, tần số này là y(t) và sẽ thay đổi mỗi T giây, với T là thời gian nhảy tần.
  - Tổng cộng có 2^m tần số khác nhau được tạo ra ứng với m bit được gửi đến.
- Bộ trộn tín hiệu sẽ kết hợp hai tín hiệu:
  - x(t): Tín hiệu mang dữ liệu.
  - y(t): Tín hiệu điều khiển từ bộ tổ hợp tần số.
- Kết quả là hai thành phần tần số tổng và hiệu, sau đó:
  - Bộ lọc BPF giữ lại một tần số thích hợp để truyền đi.
  - Tín hiệu được đưa lên kênh truyền, có thể bị ảnh hưởng bởi nhiễu, suy hao.



Hình 3.18. Quy trình giấu tin và tách tin trong âm thanh sử dụng hệ thống trải phổ FHSS

Trong bài lab này, thuật toán được mô tả như sau:

- **Đầu vào:**
  - Thông điệp bí mật.
  - File âm thanh dùng để chứa thông điệp.
- **Các bước thực hiện giấu thông điệp**
  - Bước 1: Đọc nội dung thông điệp và file âm thanh gốc, kiểm tra nếu file âm thanh gốc là stereo thì chuyển về mono, mục đích là để việc giấu tin trở nên đơn giản hơn và dễ kiểm soát. Sau đó chuyển message thành dạng bit, thêm chuỗi đồng bộ vào đầu message. Đây gọi là bước tiền xử lý.
  - Bước 2: Đọc dữ liệu tiền xử lý thu được, tạo chuỗi nhảy tần PN và đưa vào bộ tổ hợp tần.
  - Bước 3: Điều chế FSK kết hợp với nhảy tần để tạo tín hiệu FHSS.
  - Bước 4: Trộn tín hiệu FHSS vào âm thanh gốc và ghi vào file audio mới.
- **Đầu ra:** File audio chứa thông điệp.

4. Nội dung thực hành

Khởi động bài lab:

git clone https://github.com/SonNTB21DCAT164/fhss-encode

Sau đó giải nén file *fhss-encode.tar.xz* rồi chuyển thư mục đã giải nén vào trunk/labs

Vào terminal, gõ:

rebuild fhss-encode

Nhiệm vụ 1: Tạo thông điệp cần giấu

Chạy lệnh *"echo <thông điệp> > message.txt"* để tạo thông điệp cần giấu. Sinh viên có thể giấu một thông điệp bất kỳ nhưng nên để thông điệp là tiếng Anh để quá trình trở nên chính xác và thông điệp là *'hidden message'* để checkwork thành công.

echo 'hidden message' > message.txt

Sau đó thử đọc file message.txt để xác nhận đúng thông điệp.

Nhiệm vụ 2: Tiền xử lý

Bước này có nhiệm vụ đọc thông điệp từ message.txt và dữ liệu từ file âm thanh gốc. Sau đó thực hiện 2 điều kiện:

- Nếu file âm thanh là stereo, cần chuyển về mono. Mục đích là tiện cho việc xử lý giấu tin.

- Chuyển message thành dạng bit và thêm vào chuỗi đồng bộ ở đầu message. Bước thêm chuỗi đồng bộ có thể bỏ qua nếu chỉ thực hiện riêng quy trình giấu tin, nhưng trong quy trình giấu và tách tin chuẩn thì phải có bước này để phục vụ cho việc nhận diện thông điệp để tách tin.

Cuối cùng là ghi ra các file, kết thúc tiền xử lý. Toàn bộ quá trình này được thực hiện trong file *tien_xu_ly.py*, sinh viên thực hiện câu lệnh sau:

python3 tien_xu_ly.py

Sử dụng lệnh *ls* để liệt kê các file mới xuất hiện, đây là các file dùng làm đầu vào/xử lý cho quá trình giấu tin chính.

Nhiệm vụ 3: Giấu tin

Sau khi tiền xử lý, quy trình giấu tin đã sẵn sàng. Tại đây chúng ta sẽ:

- Đọc dữ liệu tiền xử lý thu được, tạo chuỗi nhảy tần PN và đưa vào bộ tổ hợp tần.

- Điều chế FSK kết hợp với nhảy tần để tạo tín hiệu FHSS.

- Trộn tín hiệu FHSS vào âm thanh gốc và ghi vào file audio mới. Lưu ý rằng nếu file audio gốc là stereo thì trước bước ghi file phải trả về định dạng ban đầu cho nó, bởi vì mục đích của việc chuyển sang mono chỉ là để tiện cho quá trình xử lý.

Toàn bộ quá trình này được thực hiện trong file *giau_tin.py,* sinh viên thực hiện câu lệnh sau:

python3 giau_tin.py

Nhiệm vụ 4: So sánh 2 file âm thanh trước và sau giấu tin

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

soxi sample_29s.wav

soxi output.wav

Kết quả cho thấy 2 file có cùng 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. Sử dụng lệnh sau

diff <(sox sample_29s.wav -t .raw -) <(sox output.wav -t .raw -)

Giải thích:

- sox: Đây là lệnh gọi công cụ sox để xử lý tệp âm thanh, ở đây công cụ đang nhận dữ liệu đầu vào là các file wav.
  - -t .raw: Tùy chọn này chỉ định loại tệp đầu ra, là tệp nhị phân thô (RAW), không có header.
  - -: Dấu gạch ngang ở cuối chỉ ra rằng tệp đầu ra sẽ được gửi đến đầu ra chuẩn (stdout), thay vì lưu vào tệp.
- diff <(...) <(...):
  - Lệnh diff được dùng để so sánh nội dung của hai tệp.
  - diff so sánh kết quả đầu ra của hai lệnh con (sox chuyển đổi tệp WAV thành tệp thô trong trường hợp này).
  - Dấu <(...) là cú pháp để thực thi một lệnh trong bash và so sánh kết quả đầu ra của chúng mà không cần lưu vào tệp.

Nếu kết quả của lệnh trên có dạng "Binary files /dev/fd/63 and /dev/fd/62 differ", điều đó cho thấy rằng 2 tệp wav có raw data khác nhau (dạng nhị phân).

Tiếp theo chúng ta thử xác định xem raw data của 2 file wav này khác nhau bắt đầu từ byte nào. Sử dụng lệnh

cmp <(sox sample_29s.wav -t .raw -) <(sox output.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