**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: lsb\_audio\_encryptalgo**

Sinh viên thực hiện: Hồ Phan Đức Anh

Mã sinh viên: B21DCAT024

Nhóm: 04

Giảng viên hướng dẫn: Đỗ Xuân Chợ

**HÀ NỘI 2025**

**Bài lab Các kỹ thuật giấu tin: lsb\_audio\_encryptalgo**

1. **Mục đích**

Giúp sinh viên hiểu được thuật toán LSB và các thuật toán mã hóa RSA, AES cùng cách kết hợp chúng để phục vụ giấu tin trong âm thanh.

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**
   1. **Thuật toán LSB**

Phương pháp LSB (Least Significant Bit) là một kỹ thuật giấu tin trong âm thanh bằng cách thay thế các bit có trọng số thấp nhất trong mẫu âm thanh số bằng dữ liệu cần giấu. Do các bit này ít ảnh hưởng đến chất lượng âm thanh tổng thể, sự thay đổi của chúng thường không được thính giác con người nhận ra.



*Giá trị 8 bit của tín hiệu A = 218*



*Giá trị 8 bit của tín hiệu B = 219 sau khi giấu bit 1 vào LSB của A*

Khi giấu 1 bit mật vào mẫu dữ liệu thì độ lệch giá trị giữa mẫu dữ liệu gốc và dữ liệu mang tin tối đa là 1. Để tăng độ bền vững của kỹ thuật giấu này, ta có thể giấu nhiều lần chuỗi thông điệp mật hoặc điều chỉnh vị trí các bit cao hơn trong mỗi mẫu để giấu tin. Các kỹ thuật giấu dùng phương pháp điều chỉnh LSB thường chọn các bit từ vị trí 1 đến 3, tính từ bên phải sang trái

**Ưu điểm:**

* Đơn giản và hiệu quả: Dễ triển khai, cho phép giấu một lượng lớn dữ liệu mà không làm tăng kích thước tệp âm thanh.
* Khó phát hiện: Sự thay đổi ở các bit LSB thường không thể nhận biết được bằng thính giác con người.

**Nhược điểm:**

* Dễ bị tấn công: Các kỹ thuật nén mất dữ liệu hoặc chỉnh sửa âm thanh có thể làm mất hoặc hỏng thông tin được giấu.
* Thiếu tính bền vững: Do sự nhạy cảm với các biến đổi, phương pháp LSB không đảm bảo tính toàn vẹn của thông tin giấu khi tệp âm thanh bị chỉnh sửa hoặc nén.

Do đó, việc kết hợp giấu tin giữa phương pháp LSB với các thuật toán mã hóa như RSA và AES là hoàn toàn hợp lý và cần thiết.

**3.2 Thuật toán mã hóa RSA**

RSA là một thuật toán mã hóa khóa công khai, được sử dụng rộng rãi trong nhiều ứng dụng bảo mật, như chữ ký số, xác thực, mã hóa đường truyền. Cơ chế hoạt động của RSA dựa trên việc sử dụng hai khóa khác nhau: một khóa công khai và một khóa bí mật. Khóa công khai là một số nguyên lớn, được công khai cho tất cả mọi người, và được sử dụng để mã hóa dữ liệu. Khóa bí mật là một số nguyên lớn, được giữ bí mật bởi người sở hữu, và được sử dụng để giải mã dữ liệu. Khóa công khai và khóa bí mật có mối quan hệ toán học với nhau, nhưng rất khó để tìm ra khóa bí mật từ khóa công khai.

Về thuật toán, cụ thể cách tạo khóa trong RSA như sau:

- Chọn hai số nguyên tố lớn p và q, sao cho p khác q.

- Tính n = p \* q. Số n này sẽ là phần của khóa công khai và khóa bí mật.

- Tính phi(n) = (p - 1) \* (q - 1). Số phi(n) này là hàm số Euler, biểu thị số lượng số nguyên dương nhỏ hơn n và nguyên tố cùng nhau với n.

- Chọn một số e sao cho 1 < e < phi(n) và e nguyên tố cùng nhau với phi(n). Số e này sẽ là phần còn lại của khóa công khai.

- Tìm một số d sao cho d \* e ≡ 1 (mod phi(n)). Số d này sẽ là phần còn lại của khóa bí mật.

Sau khi có được hai khóa, ta có thể sử dụng chúng để mã hóa và giải mã dữ liệu theo công thức sau:

- Để mã hóa một thông điệp M, ta tính C = M^e (mod n), với C là thông điệp đã mã hóa.

- Để giải mã một thông điệp C, ta tính M = C^d (mod n), với M là thông điệp gốc.

Như vậy, ta đã thấy cách hoạt động của RSA. Tuy nhiên, trong thực tế, các số p, q, e, d, n được chọn rất lớn, có thể lên đến hàng trăm chữ số, để đảm bảo tính an toàn của thuật toán. Việc phá vỡ RSA đòi hỏi phải phân tích n thành hai thừa số nguyên tố p và q, nhưng đây là một bài toán rất khó, và chưa có thuật toán nào có thể giải quyết nhanh chóng được. Vì vậy, để dễ tiếp cận, bài lab này chỉ mô phỏng các tham số với giá trị khá nhỏ.

**3.3 Thuật toán AES**

* AES là gì?
  + AES là một thuật toán mã hóa khối đối xứng (symmetric block cipher).
  + Được chọn bởi Viện Tiêu chuẩn và Công nghệ Hoa Kỳ (NIST) vào năm 2001, thay thế thuật toán DES cũ.
  + Mã hóa và giải mã dùng cùng một khóa bí mật.
* Nguyên lý hoạt động:
  + Dữ liệu được chia thành các khối 128-bit (16 byte).
  + AES hỗ trợ 3 độ dài khóa:
    - 128-bit (AES-128) → 10 vòng (rounds)
    - 192-bit (AES-192) → 12 vòng
    - 256-bit (AES-256) → 14 vòng
* Các bước chính trong mỗi vòng:
* SubBytes: Thay thế từng byte bằng giá trị từ bảng S-box (phi tuyến).
* ShiftRows: Dịch các hàng của ma trận dữ liệu.
* MixColumns: Trộn dữ liệu theo cột để khuếch tán bit.
* AddRoundKey: XOR dữ liệu với khóa con (tạo từ khóa chính ban đầu).
* Vòng cuối không có bước MixColumns.
* Ưu điểm:
  + Nhanh, dễ cài đặt bằng cả phần mềm và phần cứng.
  + Rất an toàn: hiện chưa bị phá vỡ (miễn là dùng đúng).
* Dùng rộng rãi trong các hệ thống: HTTPS, VPN, mã hóa file...
* Ứng dụng thực tế:
* Mã hóa tin nhắn, file, ổ đĩa, giao tiếp mạng…
* Nhiều thư viện như pycryptodome hỗ trợ AES dễ dàng trong Python.

**3.3 Ứng dụng trong bài lab**

Mục đích của bài lab là mô phỏng lại phương pháp LSB kết hợp với thuật toán RSA và LSB kết hợp với thuật toán AES. Thông điệp được giấu trong file âm thanh gốc sẽ không ở dạng plain text mà phải đi qua 1 lớp mã hóa, và quá trình giải mã thông điệp từ phía nhận cũng phải cung cấp key để thu được dạng bản rõ của thông điệp. Lab này sẽ xây dựng 2 giao diện GUI: LSB kết hợp với RSA và LSB kết hợp với AES.

1. **Nội dung thực hành**

Khởi động bài lab:

Vào terminal, gõ:

*rebuild lsb\_audio\_encryptalgo*

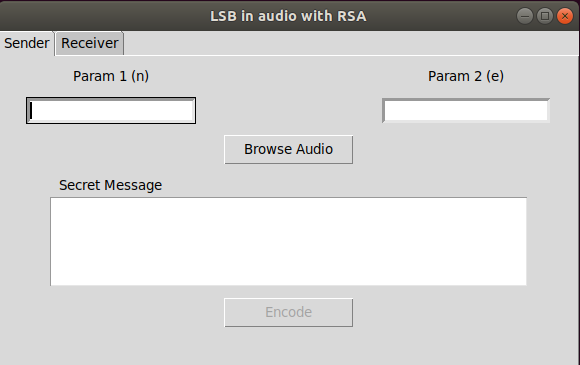
(Chú ý: Sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)

**Nhiệm vụ 1: LSB với RSA**

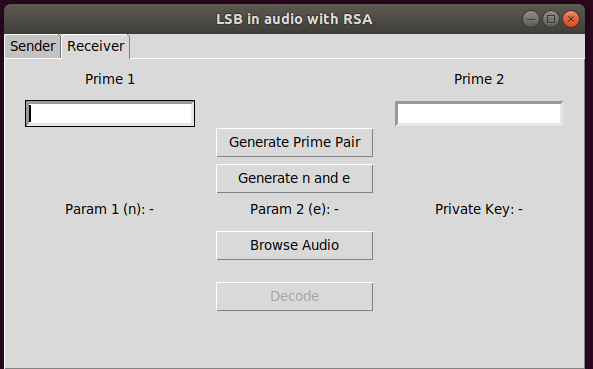
Sinh viên cd vào thư mục RSA, tại đây sinh viên sẽ thấy 1 file mainRSA.py, chạy lệnh sau:

***python3 mainRSA.py***

Giao diện mô phỏng quá trình trình gửi và nhận thông điệp hiển thị

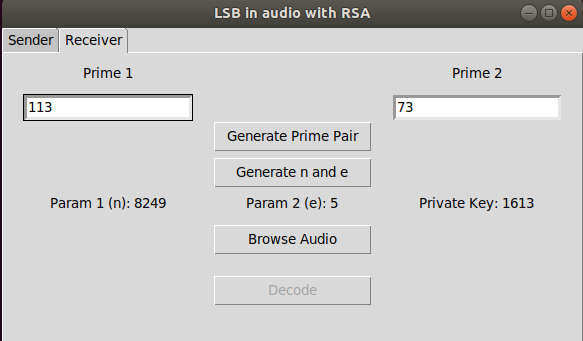


*Phía gửi*

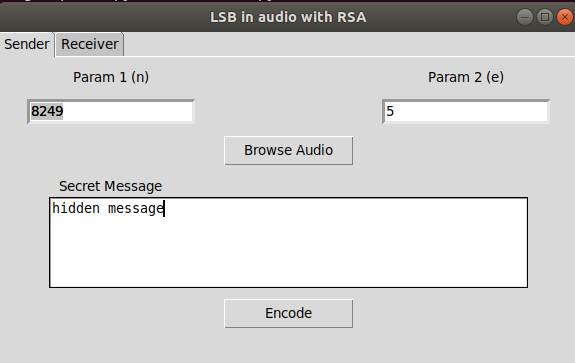


*Phía nhận*

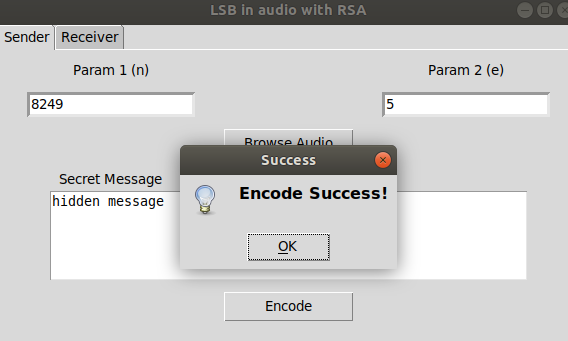
Như đã trình bày trong phần lý thuyết, đối với thuật toán RSA, ta cần 2 số nguyên số p và q, nhấn “Generate Prime Pair” để sinh ra p, q. Sau đó nhấn tiếp “Generate n and e” để sinh ra n và e (thành phần của khóa công khai) và private key. Kết quả thu được tương tự như sau



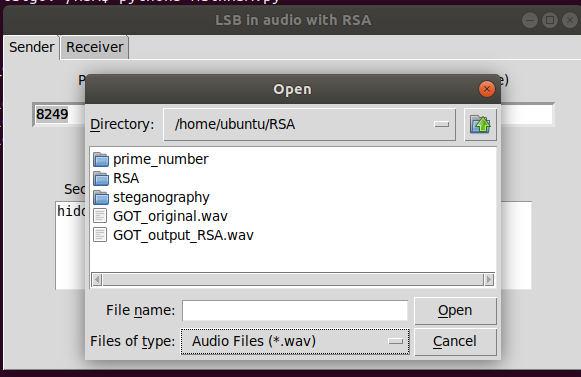
Bây giờ chuyển sang tab Sender, nhập n và e vừa sinh ra rồi Browse Audio để chọn file âm thanh gốc (trong lab này đã có sẵn file GOT\_original.wav), sau đó nhập Secret Message mong muốn (nên dùng tiếng Anh).



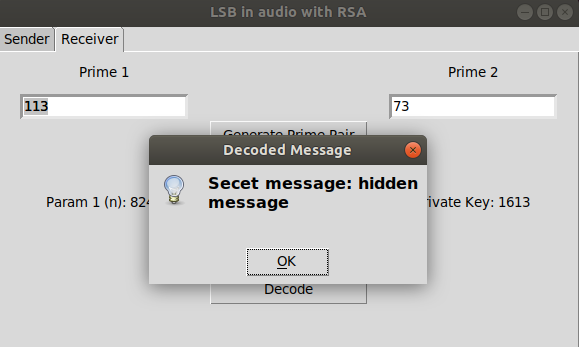
Cuối cùng nhấn Encode, thông báo thành công sẽ hiện ra



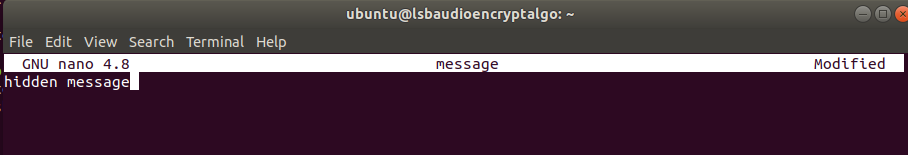
Vậy là chúng ta đã giấu thành công message vào file audio gốc, message này được mã hóa bằng thuật toán RSA và xuất ra file mới là GOT\_output\_RSA.wav, sinh viên có thể kiểm tra lại file đó bằng cách nhấn Browse Audio 1 lần nữa



Để giải mã thì ở phía nhận cần có private key. Do private key đã được sinh ra ở bước trước đó nên chúng ta chỉ cần sử dụng để giải mã thông điệp. Tại tab Receiver, nhấn Browse Audio và chọn file GOT\_output\_RSA.wav, sau đó nhấn Decode



Thông điệp đã được giải mã thành công, tạo 1 file tên *message* và lưu lại thông điệp này.

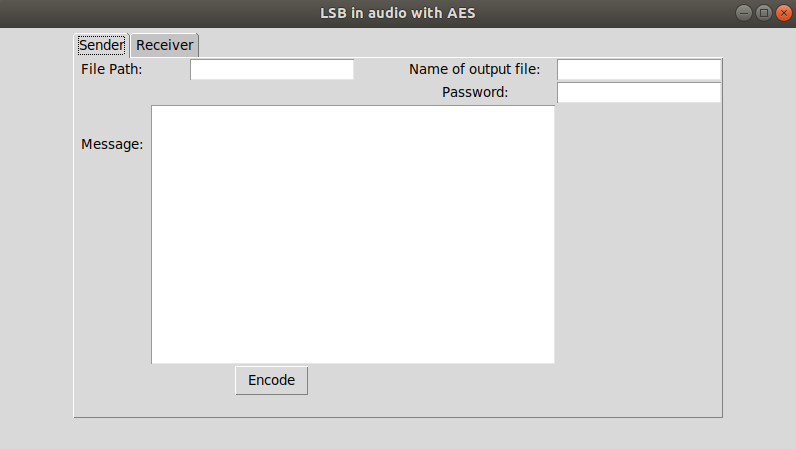


**Nhiệm vụ 2: LSB với AES**

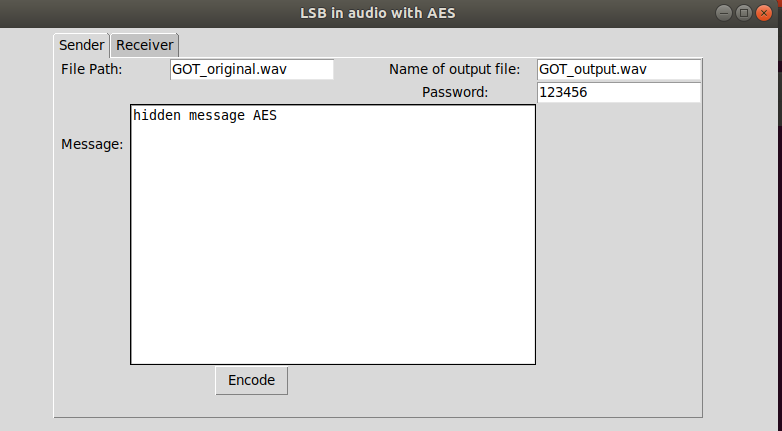
Tương tự nhiệm vụ trước, lần này chúng ta sẽ sử dụng thuật toán mã hóa AES. Do AES là mã hóa đối xứng nên nó sử dụng cùng 1 key cho cả quá trình mã hóa và giải mã. Trước tiên, chúng ta cd vào thư mục AES và chạy file mainAES.py

***python3 mainAES.py***

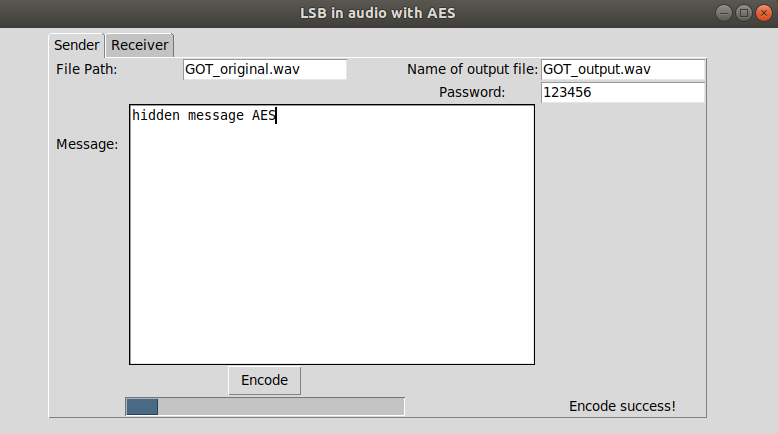
Giao diện mô phỏng quá trình gửi và nhận thông điệp hiện ra



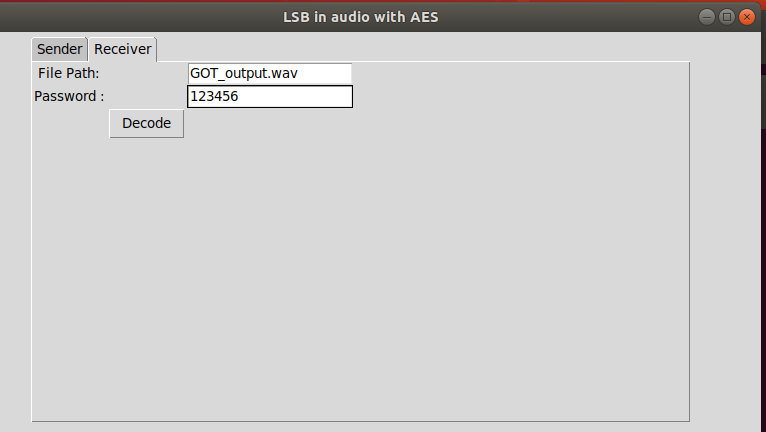
Tại tab Sender, chúng ta cần nhập đường dẫn file audio gốc, tên file audio sau khi giấu và mã hóa thông điệp, key để giải mã (Password) và thông điệp cần giấu (Message). Sinh viên có thể nhập tương tự như sau (File path là GOT\_original.wav - file âm thanh gốc đã có sẵn trong lab này)



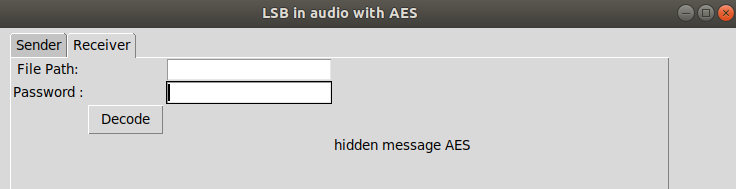
Sau đó nhấn Encode và đợi khoảng 1 phút, nếu thành công thì màn hình sẽ hiển thị tương tự như ảnh dưới



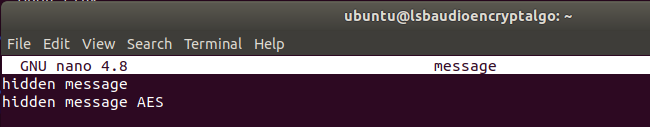
Tại tab Receiver, tiến hành giải mã và trích xuất thông điệp từ file GOT\_output.wav, sinh viên nhập tên file và key



Sau đó nhấn Decode và đợi 1 lúc, nếu thành công thì thông điệp sẽ hiển thị dưới dạng bản rõ.

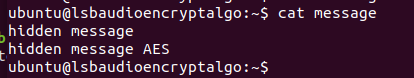


Lưu lại thông điệp đó vào file message



**Nhiệm vụ 3: Kiểm tra**

Sinh viên đã thực hiện xong 2 nhiệm vụ trên, bây giờ chỉ cần cat file message để xác nhận đã hoàn thành



**Kết thúc bài lab:**

Kiểm tra checkwork:

*checkwork*

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

*stoplab*