Hướng dẫn thực hành bài lab: Mã hóa thông điệp bằng RSA và giấu tin trong ảnh sử dụng DWT

1. Mục đích

● Hiểu cơ chế mã hóa khóa công khai RSA và cách áp dụng để bảo mật thông điệp.
● Làm quen với kỹ thuật giấu tin trong ảnh sử dụng DWT – một phương pháp dựa trên miền tần số.
● Thực hành trao đổi khóa, mã hóa và truyền thông điệp bí mật giữa hai người dùng qua mạng.
● Kết hợp bảo mật nội dung (mã hóa RSA) và bảo mật hình thức (giấu tin bằng DWT).
● Làm quen với thư viện Python hỗ trợ xử lý ảnh và giấu tin trên Ubuntu.

2. Lý thuyết thuật toán

2.1. RSA – Thuật toán mã hóa khóa công khai

● RSA là thuật toán mã hóa phi đối xứng sử dụng một cặp khóa:
○ **Khóa công khai** dùng để mã hóa dữ liệu.
○ **Khóa riêng** dùng để giải mã dữ liệu đã mã hóa.

● Ưu điểm của RSA:
○ Đảm bảo tính bí mật: chỉ người có khóa riêng mới giải mã được dữ liệu.
○ Được ứng dụng rộng rãi trong truyền thông an toàn (email, HTTPS, VPN,...).

2.2. DWT – Discrete Wavelet Transform

● DWT là kỹ thuật chuyển đổi tín hiệu từ miền không gian sang miền tần số bằng cách phân tích đa cấp (multi-level decomposition).
● Trong xử lý ảnh, DWT chia ảnh thành 4 vùng tần số: **LL (thấp-thấp)**, **LH (thấp-cao)**, **HL**, và **HH**.

● Ứng dụng DWT trong giấu tin (steganography):
○ Nhúng thông điệp vào các hệ số tần số phụ (ví dụ: LH hoặc HL) để hạn chế làm biến dạng ảnh.
○ DWT cung cấp sự cân bằng tốt giữa độ bền, tính khó phát hiện và chất lượng ảnh.

So sánh DWT và LSB/DCT:

Thuật toán	Độ bền	Chất lượng ảnh	Khả năng che giấu
LSB	Thấp	Cao (nhanh)	Dễ bị phát hiện
DCT	Trung bình	Trung bình	Tốt hơn LSB
DWT	Cao	Cao	Rất khó phát hiện

3. Nội dung thực hành

3.1. Chuẩn bị môi trường

Khởi động bài lab:
Truy cập đường dẫn: https://github.com/vuongnguyen168/dwt_rsa_coding tải bài lab về và lưu trong thư mục labtainer/trunk/labs
Cài đặt lab: rebuild -b dwt_rsa_tool
Khởi động bài lab: labtainer -r dwt_rsa_tool

Khởi động lại dịch vụ SSH:

sudo systemctl restart ssh

3.2. Tạo và trao đổi khóa RSA

Tại user1 – Tạo cặp khóa RSA (2048 bit):

openssl genpkey -algorithm RSA -out private.pem -pkeyopt rsa_keygen_bits:2048
openssl rsa -in private.pem -pubout -out public.pem

Trao đổi khóa công khai:
Tại user2, sao chép khóa từ user1 về bằng scp:

scp ubuntu@<IP_user1>:/home/ubuntu/public.pem /home/ubuntu

3.3. Mã hóa và giấu thông điệp

Tại user2 – Tạo tệp thông điệp cần mã hóa:

echo "This is the secret message" > message.txt

Mã hóa bằng khóa công khai của user1:

openssl rsautl -encrypt -inkey public.pem -pubin -in message.txt -out encrypted.bin

3.4. Giấu dữ liệu mã hóa vào ảnh bằng DWT

Cài đặt các thư viện Python cần thiết:

pip install PyWavelets Pillow numpy

Thực hiện giấu dữ liệu:
Giả sử file `encode.py` sử dụng thuật toán DWT để nhúng `encrypted.bin` vào ảnh mẫu `cover.png`.

python3 encode.py

Ảnh đầu ra sẽ là `stego_output.png`, chứa dữ liệu đã được giấu.

3.5. Truyền ảnh chứa thông điệp về user1

scp ubuntu@<IP_user2>:/home/ubuntu/stego_output.png /home/ubuntu

3.6. Giải mã thông điệp từ ảnh

Tại user1 – Giải nén dữ liệu nhúng từ ảnh:

pip install PyWavelets Pillow numpy
python3 decode.py

Dữ liệu nhúng sẽ được lưu lại thành `recovered_secret.bin`.

Giải mã dữ liệu bằng khóa riêng:

openssl rsautl -decrypt -inkey private.pem -in recovered_secret.bin -out decrypted.txt
cat decrypted.txt

Kết quả: `This is the secret message`

4. Kết quả

● Hoàn thành toàn bộ chuỗi thao tác: tạo khóa → mã hóa → giấu tin bằng DWT → giải mã.
● Thông điệp được khôi phục chính xác và đầy đủ.
● Ảnh đầu ra gần như không có sai lệch đáng kể so với ảnh gốc.

5. Nhận xét

● **Tính bảo mật**:
○ Kết hợp RSA (bảo mật nội dung) và DWT (bảo mật hình thức) giúp truyền tin an toàn và khó bị phát hiện.

● **Chất lượng ảnh**:
○ DWT giữ lại chất lượng ảnh tốt hơn LSB và khó bị phân tích hơn DCT.

● **Tính thực tiễn**:
○ Mô hình này có thể mở rộng áp dụng trong bảo mật thông tin cá nhân, xác thực nội dung số, hoặc các hệ thống giấu tin bảo mật cao.

● **Hạn chế**:
○ Độ phức tạp cao hơn so với các kỹ thuật đơn giản như LSB.
○ Yêu cầu tính toán nhiều và hiểu biết về xử lý ảnh số.