Bài thực hành: Tách tin trong ảnh được giấu tin bằng thuật toán DCT (DISCRETE COSINE TRANSFORM)

1. Mục đích

Bài thực hành này hướng dẫn sử dụng thuật toán DCT để giấu tin trong ảnh.

Thuật toán DCT (Discrete Cosine Transform – Biến đổi Cosin Rời rạc) là một phép biến đổi toán học dùng phổ biến trong xử lý ảnh, DCT chuyển tín hiệu từ miền không gian (spatial domain) sang miền tần số (frequency domain).

2. Yêu cầu đối với sinh viên

Có kiến thức cơ bản về giấu tin trong ảnh bằng thuật toán DCT.

Có kiến thức cơ bản về ngôn ngữ Python.

Đã làm bài thực hành *stego_dct_code*.

3. Nội dung thực hành

3.1 Khởi động bài lab

Cài đặt bài lab tại: *https://github.com/NgocTaif/destego_dct_code*

Giản nén và chuyển vào thư mục */labtainer/trunk/labs*.

Trên terminal, gõ:

labtainer -r destego_dct_code

Sau khi khởi động xong, hai terminal ảo xuất hiện: *alice và bob*.

3.2 Tạo kết nối và giao tiếp

Đầu tiên, ta khởi động dịch vụ *ssh* để thực giao tiếp giữa hai terminal alice và bob:

Trên cả hai terminal, ta gõ lệnh:

sudo systemctl start ssh

Trên terminal *alice*, tiến hành gửi file sang terminal *bob* bằng lệnh (mật khẩu: ubuntu):

scp encrypted_file.bin recovered_img.png ubuntu@174.20.0.10:/home/ubuntu

Trong đó, *encrypted_file.bin* là file *key thông điệp* được mã hóa bằng RSA phục vụ cho quá trình tách tin, *recovered_img.png* là ảnh được giấu tin.

Trên terminal *bob*, sử dụng lệnh *ls* kiểm tra đã có hai file trên chưa.

Tiếp tục trên terminal *bob*, sử dụng chương trình *decrypt_message.py* để giải mã key thông điệp encrypted_file.bin, với key giải mã *private_key.pem:*

python3 decrypt_message.py encrypted_file.bin private_key.pem

Giải mã thành công, ta thu được file *decrypted_file.txt*, hãy kiểm tra để biết vị trí các ma trận điểm ảnh 8x8 nào đã được giấu bit, và số lượng bit được giấu trong mỗi ma trận đó: *nano decrypted_file.txt*

3.3 Tiền xử lý

Trên terminal *bob* sử dụng lệnh: *fim recovered_img.png* để xem ảnh đã được giấu tin.

Trên terminal *bob*, tiếp tục gõ lệnh:

python3 preprocessor/create_img_matrix.py recovered_img.png

Chương trình *create_img_matrix.py* trong thư mục *preprocessor* sẽ chuyển đổi từ ảnh gốc ban đầu (*recovered_img.png*) thành ma trận điểm ảnh của ảnh xám (*image_matrix.txt*).

Thực hiện sử dụng *nano* để kiểm tra file ma trận điểm ảnh *image_matrix.txt*, nhận thấy các giá trị đều nằm trong đoạn [0, 255], các giá trị này đại diện cho độ sáng của điểm ảnh tại vị trí đó.

Qua bước tiền xử lý ta biết được kích thước của ảnh là 512 x 512 pixels. Tại quá trình tiến hành tách tin tương tự ta chia ma trận điểm ảnh (*image_matrix.txt*) này thành các khối 8x8 pixels, có thể coi ma trận điểm ảnh (*image_matrix.txt*) là 1 ma trận điểm ảnh gồm 64 khối 8x8 pixels.

Ta sẽ cần tách các bit tin từ các khối ma trận 8x8 đã được giấu.

3.4 Tách tin

Trước tiên, ta cần thực hiện trích xuất các ma trận điểm ảnh có kích thước 8x8 từ ma trận điểm ảnh tổng trong *image_matrix.txt.*

Trên terminal bob ta sử dụng chương trình *extract_matrix.py* để trích xuất các ma trận khối 8x8 đó, ta gõ lệnh:

python3 extract_matrix.py image_matrix.txt <i> <j>

Trong đó *<i>*, *<j>* là các tham số đầu vào, chương trình *extract_matrix.py* sẽ lấy khối 8x8 tại vị trí hàng i cột j ở ma trận tổng khối 512 x 512 *image_matrix.txt* mà ta đã đề cập ở bước trên. Các ma trận khối 8x8 này khi trích xuất sẽ được lưu tại *matrix_8x8_vitri_<i>_<j>.txt* trong thư mục *extracted_matrix.*

Để biết được các khối nào đã được sử dụng để giấu tin, sinh viên hãy xem file *decrypted_file.txt* đã giải mã đã đề cập trước đó.

Tiếp tục trên terminal *bob*, ta gõ các lệnh:

python3 dct_transform/dct_discrete_cosine.py
extracted_matrix/matrix_8x8_vitri_<i>_<j>.txt

Chương trình *dct_discrete_cosine.py* trong thư mục *dct_transform* sẽ thực hiện biến đổi DCT lên từng ma trận khối 8x8 để chuyển đổi sang miền hệ số DCT. Biến đổi DCT sẽ tạo ra các hệ số DCT tương ứng với các tần số khác nhau trong ảnh.

$$F(u,v) = \frac{C(u)C(v)}{4}\sum_{j=0}^{7}\sum_{k=0}^{7} f(j,k)\cos\frac{(2j+1)u\pi}{16}\cos\frac{(2k+1)v\pi}{16}$$

Công thức trên là phép biến đổi DCT được sử dụng cho chương trình *dct_discrete_cosine.py*.

f(j ,k) là các mẫu của ảnh gốc trong khối 8x8 điểm ảnh.

F(u, v) là các hệ số của khối DCT 8x8.

Trên terminal *bob*, gõ các lệnh:

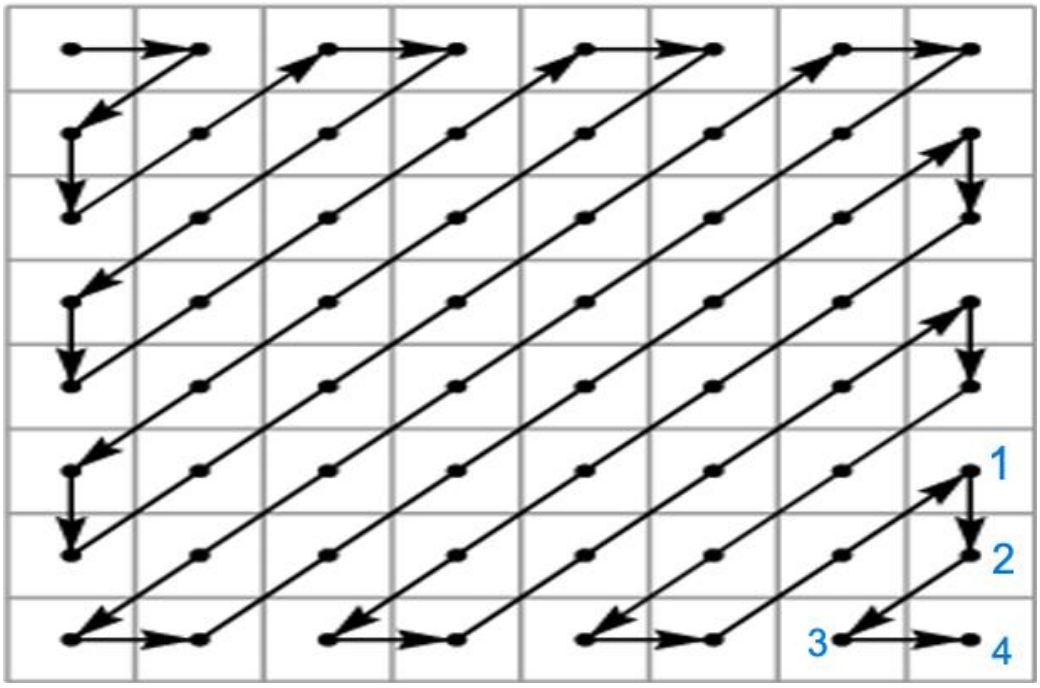
python3 dct_transform/dct_quantized.py
extracted_matrix/matrix_8x8_vitri_<i>_<j>.txt

Chương trình *dct_quantized.py* trong thư mục *dct_transform* để thực hiện lượng tử hóa tương tự như trong quá trình giấu tin các ma trận khối 8x8 sau khi đã được biến đổi DCT.

Sau khi đã biến đổi DCT và lượng tử hoá thành công các khối ma trận 8x8 trên, hãy thực hiện đọc các khối ma trận này và xem các bit đã được giấu trong đó, trong đó số lượng *x* bit nhị phân được giấu trong mỗi ma trận được đề cập trong file key *decrypted_file.txt.*

Trong bài thực hành này, mỗi ma trận sẽ được thay thế *x* bit cuối theo chiều ma trận zigzag bằng *x* bit nhị phân của thông điệp giấu.

Ví dụ nếu theo chiều ma trận zigzag như bên dưới và ta lấy 4 bit cuối của ma trận trích xuất, thì sẽ thu được chuỗi bit theo chiều sau: *1234*.



Kết quả ghép *x* bit thu được từ các khối ma trận theo trình tự ta sẽ thu được chuỗi thông điệp bit nhị phân hoàn chỉnh.

Trên terminal *bob*, để giải mã chuỗi bit trên ta sử dụng chương trình *take_data.py* để biến đổi thành dạng văn bản ASCII, sử dụng lệnh:

python3 take_data.py <chuỗi bit nhị phân>

Lưu ý: <chuỗi bit nhị phân> là tham số đầu vào và cần nhập liền kề.

Kết quả tách tin thành công, thu được thông điệp được giấu trong ảnh.

3.5 Kết thúc bài lab

Trên terminal, gõ:

stoplab

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

Sinh viên cần nộp file *.lab* để chấm điểm.

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

checkwork

Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, sử dụng lệnh:

labtainer –r destego_dct_code