Bài thực hành: Tách tin trong ảnh sử dụng thuật toán Different Expansion

1. Mục đích

Bài thực hành này hướng dẫn tách tin trong ảnh đã được giấu tin bằng thuật toán Different Expansion.

Tách tin là một khâu quan trong trong quá trình trao đổi thông điệp, dữ liệu được giấu. Thuật toán Different Expansion là một phương pháp giấu tin dựa trên miền không gian trong ảnh, thuộc nhóm Reversible Data Hiding (RDH) – nghĩa là sau khi giải giấu dữ liệu, ảnh gốc có thể được khôi phục hoàn toàn.

2. Yêu cầu đối với sinh viên

Có kiến thức cơ bản về thuật toán Different Expansion.

Đã làm bài thực hành stego_de_code.

Có kiến thức cơ bản về ngôn ngữ Python.

3. Nội dung thực hành

3.1 Khởi động bài lab

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

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

Trên terminal, gõ:

labtainer -r destego_image_code_de

Sau khi khởi động xong, một terminal ảo xuất hiện.

3.2 Tiền xử lý

Trong bài thực hành đã chuẩn bị sẵn một file ảnh xám *image.png* đã được thực hiện tiến hành giấu tin thông điệp bằng thuật toán Different Expansion. Ta có thể sử dụng lệnh *fim image.png* để tiến hành xem ảnh.

Sử dụng lệnh *ls* để kiểm tra, liệt kê các tập tin đang tồn tại.

Để có thể thực hiện quá trình tách tin từ ảnh đã được giấu thông điệp, ta cần thực hiện biến đổi lại ảnh image.png về ma trận điểm ảnh, tương tự như bước tiến hành giấu tin *(đề cập trong bài thực hành stego_code_de).*

Trên terminal, ta gõ lệnh:

python3 create_img_matrix.py image.png

(với *image.png* là tham số đầu vào)

Kết quả thành công với đầu ra thông báo *"Extracted image matrix: img_matrix.txt"*, ta thu được file img_matrix.txt chứa ma trận điểm ảnh có kích thước 512 x 512.

Sử dụng lệnh *nano img_matrix.txt* để quan sát ma trận điểm ảnh của ảnh đã được giấu tin, các giá trị đều nằm trong đoạn [0, 255].

3.3 Tách tin

Quy trình của quá trình thực hiện giấu tin đã được đề cập trong bài thực hành *stego_de_code* (sinh viên có thể xem lại).

Để có thể thực hiện tách tin, ta cần phải biết *thông điệp key* (khóa giải mã) mà người giấu tin đã sử dụng để tiến hành giấu tin trước đó.

Trong bài thực hành này, người giấu tin đã thực hiện sửa đổi thuộc tính metadata ("nhãn thông tin" - chứa thông tin về ảnh) của ảnh *image.png*. Người giấu tin đã mã hóa *thông điệp key* thành một chuỗi string để trong một thuộc tính nào đó của ảnh.

Sinh viên hãy sử dụng tool *Exiftool* để kiểm tra các thuộc tính metadata của ảnh *image.png*, nếu phát hiện được điều gì đó hãy thực hiện giải mã chuỗi string để lấy được *thông điệp key*. Để làm kiểm tra metadata, ta dùng lệnh:

exiftool image.png

Sau khi đã có được *thông điệp key*, lúc này ta cần sử dụng chúng để tiến hành trích xuất các cặp giá trị điểm ảnh mới mà người giấu tin đã thực hiện thay thế trong ma trận điểm ảnh. *Thông điệp key* sẽ cho biết vị trí của các cặp điểm ảnh đó, *tại hàng nào của ma trận?,* và *chỉ số các cặp điểm ảnh (key pairs) tương ứng tại hàng đó*.

Sử dụng chương trình *extract_key_pairs.py* để trích xuất. Trên terminal, ta gõ lệnh:

python3 extract_key_pairs.py img_matrix.txt <chỉ số hàng> <chỉ số đầu> <chỉ số cuối>

(Ví dụ ta có các cặp giá trị điểm ảnh trong một hàng của ma trận như sau: *4-5, 6-7, 8-9, 10-11,…, 18-19*. Lúc này *<chỉ số đầu>* là 4 và *<chỉ số cuối>* sẽ là 19) .

Kết quả ta thu được file *key_pairs.txt* chứa các cặp giá trị điểm ảnh mà người giấu tin đã thực hiện giấu thông điệp và thay thế vào trong ma trận điểm ảnh.

Hãy sử dụng lệnh *nano* để kiểm tra file *key_pairs.txt* chứa bao nhiêu cặp giá trị.

Tiếp đó, khi đã xác định số lượng và các cặp điểm ảnh trong file key_pairs.txt, ta tiến hành lấy các bit tin thông điệp đã được giấu. Quá trình này thực hiện giống với bước thực hiện giấu tin *(xem lại bài stego_code_de).*

Sinh viên tự mình trích xuất các bit thông điệp, cũng như khôi phục các cặp giá trị điểm ảnh theo các bước sau:

Với mỗi cặp (x', y'):

- *Chênh lệch: d' = x' – y'*
- ***Bit thông điệp: b = d' mod 4***

  Do bit thông điệp b là ở dạng thập phân của hai bit nhị phân một từ chuỗi bit nhị phân của message (đề cập trong bài stego_code_de), do đó từ giá trị b ở

dạng thập phân ta cần chuyển lại về hai bit dạng nhị phân (ví dụ: 0 → 00, 1 → 01, 2 → 10, 3 → 11).

- *Chênh lệch gốc: d = [d' / 4]*
- *Trung bình: m' = [(x' + y') / 2] (làm tròn xuống)*
- ***Khôi phục: x = m' + [(d + 1) / 2], y = m' – [d / 2] (làm tròn xuống)***

Sau quá trình thực hiện tách tin, ta thu được chuỗi bit thông điệp được ghép lại khi trích xuất từ các cặp giá trị điểm ảnh, sinh viên có thể sử dụng chương trình *binary_to_ascii.py* để chuyển chuỗi bit thành dạng ASCII văn bản thông thường:

python3 binary_to_ascii.py <chuỗi bit>

Lưu ý: <chuỗi bit> cần viết liền kề.

Quá trình tách tin thành công, sinh viên hãy thực hiện đọc file *results.txt* và thực hiện điền và trả lời các câu hỏi có trong file để hoàn tất bài thực hành:

- *Chuỗi bit thu được sau khi trích xuất?*
- *Thông điệp ASCII được chuyển đổi từ chuỗi bit?*
- *Các cặp giá trị điểm ảnh ban đầu đã được khôi phục là?*

3.4 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_image_code_de