Bài thực hành: Sử dụng thuật toán Different Expansion để giấu tin trong ảnh

1. Mục đích

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

Việc giấu tin trong ảnh là một nhánh quan trọng trong bảo mật thông tin. 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.

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/stego_de_code*

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

Trên terminal, gõ:

labtainer -r stego-de-code

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 *input_img.png* để ta tiến hành thực hiện giấu tin. Ta có thể sử dụng lệnh *fim input_img.png* để tiến hành xem ảnh.

Tại thư mục *preprocessor* chứa hai file mã python: *img_gray_512x512.py* và *create_img_matrix.py*

Trong đó, file *img_gray_512x512.py* sẽ thực hiện chuyển đổi từ ảnh gốc ban đầu (*input_img.png*) sang dạng ảnh xám/ảnh nhị phân (*img_gray_512x512.png*). Còn file *create_img_matrix.py* tiến hành tạo ma trận điểm ảnh từ ảnh xám (*img_gray_512x512.png*) được tạo ra trước đó.

Đầu tiên trên terminal, gõ lệnh:

python3 preprocessor/img_gray_512x512.py input_img.png

(với *input_img.png* là tham số đầu vào)

Ta sẽ sử dụng ảnh xám này sẽ được sử dụng để giấu tin.

Tiếp đó, ta gõ lệnh:

python3 preprocessor/create_img_matrix.py img_gray_512x512.png

Kết quả ta thu được file *img_matrix.txt* là một 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 xám, ta có nhận thấy các giá trị đều nằm trong đoạn [0, 255].

3.3 Giấu tin

Quy trình của quá trình thực hiện giấu tin như sau:

- Xác định các cặp điểm (pixel) trong ma trận điểm ảnh, dựa trên sự tương quan không gian.
- Tính toán một giá trị đại diện cho tương quan và một giá trị trung bình (như hiệu chênh lệch giữa hai điểm ảnh).
- Nhúng thông điệp bằng cách mở rộng chênh lệch để chứa bit tin mật, sau đó tính các cặp giá trị điểm ảnh mới thay thế lại vào trong ma trận.
- Đảm bảo các thay đổi nằm trong phạm vi hợp lệ (0–255).
- Từ ma trận đã được thay thế các cặp điểm ảnh mới, ta thực hiện khôi phục ảnh mới đã được giấu tin.

Trước tiên, ta chọn các cặp điểm ảnh từ ma trận *img_matrix.txt*, ví dụ (*x*,*y*), với giá trị *x* và *y* là hai giá trị điểm ảnh liền kề nhau.

Thực hiện, sử dụng file *take_8pairs.py* tại thư mục *processing_de* để lấy 8 cặp điểm ảnh (tương ứng 16 giá trị liền kề nhau) đầu tiên tại hang đầu tiên của ma trận *img_matrix.txt*

Trên terminal, ta gõ lệnh:

python3 processing_de/take_8pairs.py img_matrix.txt

8 cặp điểm ảnh trích xuất được lưu tại *8pairs.txt* trong thư mục *processing_de,* hãy dùng lệnh nano để kiểm tra.

Trên terminal, gõ lệnh:

python3 ascii_to_binary.py

Chương trình *ascii_to_binary.py* sẽ thực hiện chuyển đổi thông điệp được giấu là “DE” sang 16 bit ở dạng nhị phân để ta có thể tiến hành giấu các bit trong các cặp điểm ảnh.

Trong bài thực hành này yêu cầu sinh viên tự thực hiện việc giấu bit thông điệp và tính toán các giá trị điểm ảnh mới của ma trận theo công thức Different Expansion.

Thực hiện các bước sau:

1. Tính chênh lệch và trung bình (với x, y là từng cặp điểm ảnh ta đã trích xuất lưu trong 8pairs.txt):

- ***Chênh lệch: d = x – y.***
- ***Trụng bình (làm tròn xuống): m = [(x+y) / 2]***

2. Mở rộng chênh lệch và tính giá trị điểm ảnh mới:

- *Thực hiện nhúng bit thông điệp b, do đã trích xuất 8 cặp điểm ảnh, nên ta cần tính 8 giá trị thông điệp b bằng cách chuyển 2 bit một từ chuỗi 16 bit thu được từ kết quả chương trình ascii_to_binary.py sang dạng thập phân và gán cho từng giá trị b. Ví dụ: từ chuỗi 0101001100100100, ta chuyển hai bit một sang dạng thập phân (00 → 0, 01 → 1, 10 → 2, 11 → 3), ta thu được 1 1 0 3 0 2 1 0, tương ứng 8 giá trị b sẽ được giấu trong 8 cặp điểm ảnh.*

  Tính chênh lệch mới: d' = 2.d + b

- ***Tính giá trị điểm ảnh mới:***

  x' = m + [(d'+1) / 2] *(làm tròn xuống)*

  y' = m – [d' / 2] *(làm tròn xuống)*

Sau khi đã tính được 8 cặp giá trị điểm ảnh mới *(x', y')*, thực hiện nhập 8 cặp (x', y') lưu vào trong file *new_8pairs.txt* tại thư mục *processing_de*, mỗi cặp một dòng tương tự như trong file *8pairs.txt*.

Tiến hành, thay thế lại 8 cặp điểm ảnh mới vào lại ma trận *img_matrix.txt* bằng chương trình *create_new_img_matrix.py* tại thư mục *processing_de.*

Trên terminal, ta gõ lệnh:

python3 processing_de/create_new_img_matrix.py img_matrix.txt processing_de/new_8pairs.txt

Chương trình *create_new_img_matrix.py* sẽ thay thế các giá trị mới, và tạo ma trận mới được lưu trong file *new_img_matrix.txt.*

3.4 Khôi phục ảnh

Thực hiện bước cuối cùng, ta khôi phục ảnh mới đã được giấu tin từ ma trận điểm ảnh đã được thay thế 8 cặp điểm ảnh mới *new_img_matrix.txt* bằng chương trình *recovery_img.py*.

Trên terminal, ta gõ lệnh:

python3 recovery_img.py new_img_matrix.txt

Kết quả ta thu được một ảnh mới đã giấu tin *recovered_img.png* giống hoàn toàn so với ảnh gốc và không thể phân biệt bằng mắt. Sử dụng lệnh *fim recovered_img.png* để kiểm chứng điều đó.

3.5 So sánh thuật toán

Qua các nhiệm vụ ở trên, có thể thấy DE là một thuật toán giấu tin trong miền không gian (spatial domain), hoạt động bằng cách mở rộng chênh lệch giữa hai pixel lân cận (thường là một cặp pixel) để nhúng dữ liệu. DE chỉ thay đổi chênh lệch pixel, nên sự biến dạng hình ảnh thường không đáng kể, tức là DE cho phép khôi phục gần như tuyệt đối so với ảnh gốc sau khi tiến hành giấu tin trong ảnh.

Đối với các thuật toán giấu tin trong miền tần số như DCT, chúng biến đổi các pixel ảnh thành các hệ số tần số DCT, sau đó nhúng dữ liệu vào các hệ số này, do đó mà các hệ số được nhúng giấu thông điệp sẽ bị biến dạng đáng kể, và khi thực hiện khôi phục lại ảnh, các phần ma trận điểm ảnh được giấu tin của ảnh cho ra sẽ có chất lượng thay đổi so với ban đầu.

Để kiểm chứng các điều trên, trong bài thực hành tại thư mục *dct_items* chứa hai file là *img_dct.png* và *img_matrix_dct.txt* lần lượt là ảnh được khôi phục và ma trận điểm ảnh đã được tiến hành giấu tin bằng phương pháp DCT. Sinh viên cần thực hiện các cách để thực hiện so sánh giữa hai thuật toán.

Đầu tiên, ta có thể thực hiện tạo ảnh "chênh lệch" (difference image) để trực quan hóa. Ảnh “chênh lệch” sẽ hiển thị những vùng trong ảnh bị thay đổi so với ảnh còn lại, tức là ta sẽ thực hiện hiện thị vùng ảnh khác biệt giữa hai ảnh được khôi phục từ thuật toán DE và thuật toán DCT. Những vùng màu tối → không có sự thay đổi (pixel giống nhau). Những vùng màu sáng hoặc nhiễu lốm đốm → có sự thay đổi (pixel khác nhau). Trên terminal, ta gõ lệnh:

convert recoverd_img.png dct_items/img_dct.png -compose difference -composite <ảnh_chênh_lệch>.png

Đầu ra sẽ tạo một file ảnh “chênh lệnh”, sinh viên hãy kiểm tra kỹ lại ảnh để phát hiện ra điều đặc biệt, sau đó có thể kiểm chứng lại bằng cách xem lại hai *recoverd_img.png img_dct.png*.

Tiếp theo, ta thực hiện sử dụng phương pháp tính chênh lệch, độ biến dạng giữa các pixel (calculate distortion) để đo lường mức độ thay đổi giữa ma trận điểm ảnh gốc (img_matrix.txt) và ma trận đã giấu tin (new_img_matrix.txt và img_matrix_dct.txt) sau khi áp dụng các thuật toán giấu tin như DE và DCT. Trong đó, độ biến dạng là tổng giá trị tuyệt đối của sự khác biệt giữa các giá trị pixel của ma trận gốc và ma trận đã giấu tin theo công thức:

$$\text{Distortion} = \sum_{i,j} |\text{original}[i][j] - \text{stego}[i][j]|$$

Sinh viên thực hiện chạy chương trình *calculate_distortion.py* để tính độ biến dạng của ma trận điểm ảnh sau khi giấu bằng thuật toán DE và DCT với ma trận điểm ảnh ban đầu, trong đó đầu vào lần lượt là các file ma trận điểm ảnh: *img_matrix.txt, new_img_matrix.txt và img_matrix_dct.txt*.

Cuối cùng, ta thực hiện phương pháp tính histogram (biểu đồ tần suất), đây là cách quan trọng để thống kê, giúp đánh giá sự thay đổi phân bố giá trị pixel sau khi áp dụng các thuật toán giấu tin như DE và DCT. Histogram giúp phân tích sự thay đổi phân bố giá trị pixel giữa ma trận gốc và ma trận đã giấu tin, từ đó đánh giá khả năng kháng phân tích thống kê của thuật toán giấu tin. Tính độ chênh lệch Histogram theo công thức:

$$\text{Difference} = \sum_{i=0}^{255} |\text{hist\_original}[i] - \text{hist\_stego}[i]|$$

Trong đó, các *hist_original, hist_stego* là các tần suất xuất hiện của các giá trị điểm ảnh của ma trận gốc và ma trận đã giấu tin.

Sinh viên thực hiện chạy chương trình *calculate_histogram.py* để tính độ chênh lệch histogram giữa các ma trận điểm ảnh, đầu vào tương tự như trên. Đầu ra kết quả ngoài giá trị độ chênh lệch histogram còn một file ảnh biểu đồ *histogram_comparison.png,* sinh viên hay kiểm tra bằng lệnh *fim,* thông qua đó ta sẽ nhận thấy được rằng thuật toán DE ít thay đổi các giá trị pixel hơn so với thuật toán DCT khi nhúng thông điệp giấu tin, phản ánh việc DCT thay đổi nhiều pixel và tạo ra các giá trị mới hơn thuật toán DE.

Sau khi thực hiện các thao tác trên, sinh viên hãy điền kết quả thu được trong file *compare.txt* với các câu hỏi:

- *How many is the distortion of DE and DCT?*
- *How many is the histogram of DE and DCT?*

3.6 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 stego_de_code