

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

Giải 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 `find 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 hàng 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:** $d = x - y$.
- **Trung 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 ($00 \rightarrow 0, 01 \rightarrow 1, 10 \rightarrow 2, 11 \rightarrow 3$), 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:**

$$x' = m + [(d'+1) / 2] \text{ (làm tròn xuống)}$$

$$y' = m - [d' / 2] \text{ (làm tròn xuống)}$$

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`.

Tiếp 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 `find recovered_img.png` để kiểm chứng điều đó.

3.5 Kết thúc bài lab

Trên terminal, gõ:

```
stoptlab
```

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