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

1. Mục đích

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

Việc giấu tin trong ảnh là một nhánh quan trọng trong bảo mật thông tin. DFT là một phương pháp phổ biến vì khả năng giấu lượng lớn dữ liệu mà ít làm biến dạng ảnh.

2. Yêu cầu đối với sinh viên

Có kiến thức cơ bản về thuật toán DFT.

3. Nội dung thực hành

3.1 Khởi động bài lab

Cài đặt bài lab tại: https://github.com/dotrantrung2003/stego-image-code-dft

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

Trên terminal, gõ:

labtainer -r stego-image-code-dft

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

3.2 Tiền xử lý

Trên terminal sử dụng lệnh: *fim image.png* để xem ảnh được chuẩn bị.

Trên terminal, gõ lệnh:

python3 preprocessing.py image.png

Chương trình *preprocessing.py* sẽ chuyển đổi từ ảnh gốc ban đầu (*image.png*) thành ảnh xám (*gray_image.png*) và ma trận điểm ảnh của ảnh xám (*gray_matrix.txt*).

Ảnh xám này sẽ được sử dụng để giấu tin.

Quan sát ma trận điểm ảnh của ảnh xám, ta có 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 1280 x 720 pixels. Ta chia ma trận điểm ảnh (*gray_matrix.txt*) thành các khối 8x8 pixels => Ta có thể coi ma trận điểm ảnh (*gray_matrix.txt*) là 1 ma trận khối có kích thước 160 x 90 (khối 8x8).

Trên terminal, gõ lệnh:

sudo cp gray_matrix.txt new_gray_matrix.txt

để tạo 1 bản sao của ma trận điểm ảnh.

3.3 Giấu tin

Trên terminal, gõ lệnh:

python3 steganography_1.py gray_matrix.txt

Chương trình *steganography_1.py* sẽ lấy khối 8x8 tại vị trí hàng i cột j (i, j nhập từ bàn phím) ở ma trận khối mà ta đã đề cập ở bước trên.

Khối 8x8 trích xuất được lưu tại *gray_block.txt*.

Trên terminal, gõ lệnh:

python3 DFT.py gray_block.txt

Chương trình DFT.py sẽ thực hiện biến đổi DFT và lượng tử hóa lên khối 8x8 vừa trích xuất.

$$F(u,v) \; + \; \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,\, y)\, e^{-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

Công thức trên là phép biến đổi DFT.

f là ma trận cần biến đổi (miền không gian)

F là ma trận sau khi biến đổi (miền tần số)

M, N là kích thước của ma trận.

Ta có thể dễ dàng nhận thấy ma trận mới là một ma trận số phức. Do đó kết quả thu được sẽ được lưu thành 2 phần: phần thực (*dft_real.txt*) và phần ảo (*dft_imag.txt*).

Ta sẽ sử dụng phần thực để giấu tin.

Quan sát *dft_real.txt*, giá trị F(0, 0) gọi là hệ số DC biểu thị mức năng lượng tổng thể của ảnh. Không chứa thông tin về chi tiết hay cạnh ảnh, chỉ phản ánh tổng mức sáng.

Miền tần số thấp là các giá trị gần F(0, 0) thể hiện vùng màu mịn, chuyển sắc chậm.

Miền tần số cao là các giá trị xa F(0, 0) thể hiện chi tiết sắc nét, cạnh của ảnh.

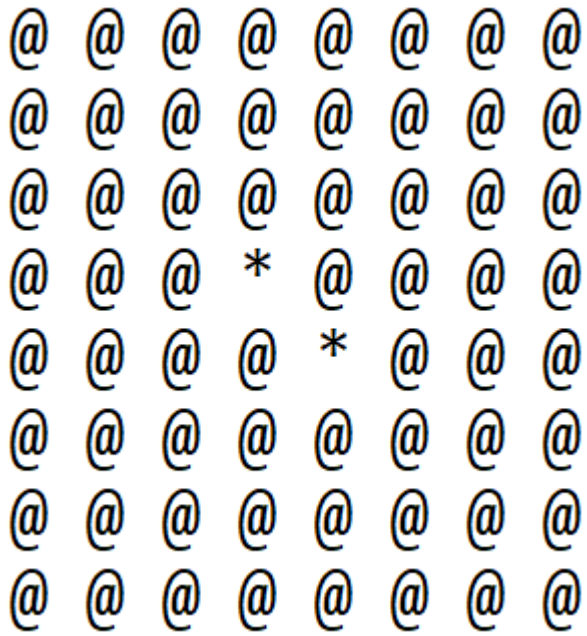
Giấu tin vào miền tần số thấp (gần F(0,0)):

- **Ưu điểm:** Ổn định, ít bị mất dữ liệu khi nén ảnh hoặc lọc nhiễu. Giữ được tin ẩn ngay cả khi ảnh bị chỉnh sửa nhẹ.
- **Nhược điểm:** Ảnh hưởng đến phần chính của ảnh, có thể gây thay đổi màu sắc hoặc làm mờ ảnh. Dễ bị phát hiện bằng các phương pháp kiểm tra nhiễu trong ảnh.

Giấu tin vào miền tần số cao (xa F(0,0)):

- **Ưu điểm:** Ít ảnh hưởng đến chất lượng ảnh gốc vì chỉ thay đổi chi tiết nhỏ. Khó bị phát hiện bằng mắt thường.
- **Nhược điểm:** Dễ mất thông tin nếu ảnh bị nén hoặc làm mịn. Không ổn định nếu ảnh bị chỉnh sửa nhiều.
- **Chọn giấu tin vào miền tần số trung bình**

Ta sẽ giấu tin vào 2 vị trí (3, 3) và (4, 4).



Với mỗi vị trí ta sẽ giấu 1 bit.

Giả sử thông điệp cần giấu có dạng nhị phân là 01:

- Vị trí (3, 3) = 11 = 0000 1011 (nhị phân). Thay thế 1 bit cuối cùng thành 1 bit cần giấu. 0000 1011 → 0000 101**0** = 10. Vị trí (3, 3) sau khi giấu sẽ là 10 (Giấu bit thứ nhất).

- Vị trí (4, 4) = 15 = 0000 1111 (nhị phân). Thay thế 1 bit cuối cùng thành 1 bit cần giấu. 0000 1111 → 0000 111**1** = 11. Vị trí (4, 4) sau khi giấu sẽ là 15. (Giấu bit thứ hai).

Tuy nhiên, các giá trị tại 2 vị trí (3, 3) và (4, 4) phần lớn đều bằng 0 nên chỉ cần đặt bit cần giấu có giá trị bằng 1 chính xác.

Sau khi giấu xong, trên terminal, gõ lệnh:

python3 IDFT.py dft_real.txt dft_imag.txt

Chương trình *IDFT.py* sẽ thực hiện biến đổi IDFT đưa 2 phần thực ảo (miền tần số) trở về ma trận 8x8 (miền không gian). Thu được *new_gray_block.txt* là khối 8x8 mới đã được giấu tin.

Trên terminal, gõ lệnh:

sudo python3 steganography_2.py new_gray_block.txt new_gray_matrix.txt

Chương trình *steganography_2.py* sẽ thay thế *new_gray_block.txt* vào vị trị hàng i cột j (i, j nhập từ bàn phím) của ma trận bản sao mà ta đã đề cập ở bước trên để tạo được ma trận điểm ảnh mới đã được giấu tin (*new_gray_matrix.txt*).

Các bước trên là hướng dẫn để giấu 2 bit vào 1 khối 8x8. Để hoàn thành bài thực hành này sinh viên cần giấu thông điệp: DFT.

Hướng dẫn:

- Chuyển thông điệp cần giấu về dạng nhị phân 8 bit của mã ASCII tương ứng. Ví dụ D → 68(ASCII) → 0100 0100. Chia thành các cặp bit để giấu 0100 0100 → 01 00 01 00 (4 cặp). Mỗi cặp sẽ được giấu vào 1 khối 8x8 như đã hướng dẫn ở trên.

- Vậy DFT sẽ có độ dài 24 bit → 12 cặp. Sinh viên cần giấu 12 cặp này vào 12 khối 8x8 (vị trí (0, 0), vị trí (0, 1) … vị trí (0, 12)).

- Các bước tổng quan: Sử dụng *steganography_1.py* để lấy khối 8x8 tại vị trí hàng i cột j của *gray_matrix.txt*→ DFT.py để biến đổi DFT → Giấu tin vào phần thực → IDFT.py để biến đôi IDFT → Thay thế khối 8x8 đã giấu tin vào vị trị hàng i cột j của *new_gray_matrix.txt.*

3.4 Khôi phục ảnh

Trên terminal, gõ lệnh:

python3 restore.py new_gray_matrix.txt

Chương trình restore.py sẽ chuyển đổi từ ma trận điểm ảnh trở thành hình ảnh (*new_gray_matrix.txt*).

Kết quả thu được ảnh xám mới đã được giấu tin.

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 stego-image-code-dft