Bài Thực Hành: Giấu Tin Trong Video Bằng Kỹ Thuật DCT Sử Dụng Phương Pháp Phát Hiện Thay Đổi Khung Cảnh

1. **Mục đích**

- Hiểu về kỹ thuật giấu tin trong video sử dụng phát hiện thay đổi khung cảnh.
- Áp dụng kỹ thuật DCT để nhúng và trích xuất thông điệp bí mật.
- Thực hành giấu tin và truyền video giữa hai container (Alice và Bob).

2. **Yêu cầu thực hành**

- Có kiến thức cơ bản về ngôn ngữ lập trình Python.
- Hiểu khái niệm Steganography và kỹ thuật DCT.
- Sử dụng terminal với các lệnh cơ bản và công cụ: ffmpeg, scp, Python, …
- Hiểu và nhận biết các điểm chuyển cảnh và phương pháp phát hiện

3. **Nội dung thực hành**

3.1. Khởi động bài lab:

- Chạy lệnh trên terminal:

  labtainer stego_code_scenechange_dct

  Tùy chọn -r đảm bảo môi trường được làm mới (reset) nếu đã chạy trước đó, cung cấp một môi trường sạch để thực hành.

  (Chú ý: Sinh viên sử dụng của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm.)

  Môi trường lab được khởi động. Để minh họa nguyên tắc giấu tin trong video HEVC, lab sử dụng một video mẫu và các công cụ như FFmpeg và Python.

3.2. Xem video và đếm các điểm chuyển cảnh

- Trên terminal của Alice, sử dụng ls để liệt kê các file có trong thư mục
- Xem video gốc và tập trung đếm các điểm chuyển cảnh, bạn có thể nhận biết được bao nhiêu điểm chuyển cảnh? Hãy ghi lại đáp án trong file answer.txt.

  ffplay <video>

- Có một đoạn code python để xác định các điểm chuyển cảnh được lưu trên terminal của Alice, chạy code để xem bạn có thực sự xác định đúng các điểm chuyển cảnh không.

  Python3 DetectSceneChange.py

  Một file scenes.txt được tạo ra liệt kê các frame chuyển cảnh. Bạn có thể xem và ghi nhớ để phục vụ cho các bước sau

3.3. Tách video và nhúng thông điệp vào frame chỉ định

Mục đích của việc xác định các frame chuyển cảnh là để giúp chúng ta có thể lựa chọn các frame này cho mục đích giấu tin.

Thông thường các frame này sẽ có sự thay đổi lớn về nội dung nên làm cho thông điệp khó bị phát hiện đồng thời các frame này ít bị ảnh hưởng với nén video, bảo toàn thông điệp.

- Tạo 1 thư mục để lưu các frame ảnh:

  Mkdir frames

- Để đảm bảo quá trình tách và nén video được đồng bộ, hãy xem các thông số của video gốc

ffprobe -v error -select_streams v:0 -show_entries stream=r_frame_rate <video>

ffprobe là 1 công cụ phân tích tệp media, các tùy chọn

-v error: chỉ hiển thị lỗi, giảm đầu ra dư thừa.

-select_streams v:0: Chọn luồng video đầu tiên.
-show_entries stream=r_frame_rate: Lấy tốc độ khung hình (fps) của video

- Tách video với fps giống với thông số vừa xem

ffmpeg -i <video> -vf "fps=xx" -q:v 2 frames/frame_%04d.png

-i <video>: Chỉ định tệp video đầu vào.
-vf "fps=xx": Bộ lọc video đặt tốc độ khung hình là xx (thay bằng fps từ ffprobe). Đảm bảo tách đúng tốc độ.
-q:v 2: Đặt chất lượng ảnh đầu ra (1 = cao nhất, 31 = thấp nhất; 2 đảm bảo PNG chất lượng cao).
frames/frame_%04d.png: Mẫu tên frame đầu ra (ví dụ: frame_0001.png, frame_0002.png

- Kiểm tra xem thư mục frames đã chứa các frame ảnh chưa. Nếu đã có, bây giờ chúng ta sẽ nhúng thông điệp vào frame ảnh chỉ định bằng kĩ thuật DCT.

Python3 enc_dct.py secret.txt frames/frame__xxxx.png

Hãy lựa chọn 1 frame ảnh tùy ý trong thư mục, đảm bảo đó là 1 trong những frame chứa chuyển cảnh, nếu không thì mục đích của bài lab sẽ không còn ý nghĩa.

3.4. Tái tạo lại video chứa thông điệp

- Ghép các frame lại thành video

Vì frame chứa thông điệp nằm ở 1 trong các frame giữa. Việc bạn cần làm là nén các frame thành 3 phần, từ frame 1=> trước frame bạn chọn, frame chứa thông điệp cần chuyển sang mp4 không mất mát, cuối cùng là các frame còn lại.

ffmpeg -framerate 30 -start_number 1 -i frames/frame_%04d.png -c:v libx264 -crf 23 -preset fast -r 30 -frames:v xxxx -movflags +faststart part1_lossy.mp4

- -framerate 30: Đặt tốc độ khung hình đầu vào là 30 fps (điều chỉnh theo video gốc).
- -start_number 1: Bắt đầu đánh số frame từ 1.
- -i frames/frame_%04d.png: Nhập các frame (ví dụ: frame_0001.png đến frame_xxxx.png).
- -c:v libx264: Sử dụng codec H.264 để nén.
- -crf 23: Hệ số tốc độ không đổi (0 = không mất mát, 51 = tệ nhất; 23 cân bằng chất lượng và kích thước).
- -preset fast: Tốc độ mã hóa nhanh (mã hóa nhanh hơn, tệp hơi lớn).
- -r 30: Đặt tốc độ khung hình đầu ra là 30 fps.
- -frames:v xxxx: Giới hạn đầu ra đến xxxx frame (thay bằng số frame trước frame đã chọn).
- -movflags +faststart: Tối ưu MP4 để phát trực tuyến nhanh bằng cách đặt metadata ở đầu.
- Mục đích: Tạo video nén (part1_lossy.mp4) từ frame 1 đến xxxx-1.

ffmpeg -loop 1 -framerate 30 -i frames/frame_xxxx.png -c:v libx264 -qp 0 -r 30 -frames:v 1 framexxxx_lossless.mp4

- -loop 1: Xem ảnh đơn như video một khung.
- -framerate 30: Đặt tốc độ khung hình đầu vào là 30 fps.
- -i frames/frame_xxxx.png: Nhập frame đã chọn (ví dụ: frame_0100.png) chứa thông điệp ẩn.
- -c:v libx264: Sử dụng codec H.264.
- -qp 0: Tham số lượng tử hóa (0 = mã hóa không mất mát).
- -r 30: Đặt tốc độ khung hình đầu ra là 30 fps.

- -frames:v 1: Chỉ xuất một frame.
- Mục đích: Chuyển frame chứa thông điệp thành video một khung không mất mát (framexxxx_lossless.mp4) để bảo toàn thông điệp.

ffmpeg -framerate 30 -start_number xxxx -i frames/frame_%04d.png -c:v libx264 -crf 23 -preset fast -r 30 -movflags +faststart part2_lossy.mp4

- -framerate 30: Đặt tốc độ khung hình đầu vào là 30 fps.
- -start_number xxxx: Bắt đầu đánh số frame từ xxxx (frame sau frame đã chọn).
- -i frames/frame_%04d.png: Nhập các frame còn lại (ví dụ: frame_xxxx.png đến cuối).
- -c:v libx264: Sử dụng codec H.264.
- -crf 23: Cân bằng chất lượng và kích thước tệp.
- -preset fast: Ưu tiên tốc độ mã hóa.
- -r 30: Đặt tốc độ khung hình đầu ra là 30 fps.
- -movflags +faststart: Tối ưu cho phát trực tuyến.
- Mục đích: Tạo video nén (part2_lossy.mp4) từ các frame sau frame đã chọn đến cuối.

Hãy thay các thông số xxxx bằng frame bạn đã chọn và sửa lại fps nếu như video đầu vào bạn đã xem và tách không trùng với fps mặc định của câu lệnh là 30.

- Chuyển 3 phần video sang dạng trung gian để dễ dàng ghép video

ffmpeg -i part1_lossy.mp4 -c copy -f mpegts part1.ts
ffmpeg -i part2_lossy.mp4 -c copy -f mpegts part2.ts
ffmpeg -i frame74_lossless.mp4 -c copy -f mpegts framexxxx.ts

- -i part1_lossy.mp4: Nhập đoạn video đầu tiên.
- -c copy: Sao chép luồng video và âm thanh mà không mã hóa lại.
- -f mpegts: Xuất dưới định dạng MPEG Transport Stream (TS), phù hợp để nối.
- Mục đích: Chuyển 3 phần sang định dạng TS trung gian để ghép.

- Ghép video cuối cùng

ffmpeg -i "concat:part1.ts|frame74.ts|part2.ts" -c copy -r 30 -movflags +faststart video_hidden.mp4

- Kiểm tra dung lượng video gốc và video từ tạo để xem dung lượng của 2 file. Phải đảm bảo không chênh lệnh quá nếu. Nếu như video chứa thông điệp chỉ hơn video gốc xấp xỉ dưới 1MB thì coi như bạn đã thành công.

ls -lh <video gốc>
ls -lh video_hidden.mp4

3.5. Gửi video cho bob và giải mã

- Sử dụng ssh để gửi video cho bob thông qua câu lệnh:

Scp video_hidden.mp4 ubuntu@<ip máy bob>:/home/ubuntu
Mật khẩu của mỗi container chính là tên của container đó.

- Sau khi bob nhận được video, hãy tiến hành xem video để xem có khác biệt nào không, thực hiện các bước tương tự để Bob có thể tách frames vào thư mục chỉ định
- Bob tìm kiếm thông điệp qua frame chuyển cảnh:

Python3 dec_dct.py frames/framexxxx.png

Một file secrect.txt được sinh ra, hãy đọc file, nếu như thông điệp trùng khớp với secret.txt gốc thì bạn đã thành công

4. Kết quả cần đạt

- Chạy được tất cả các bước như yêu cầu.

- Cần nộp 1 file: trong thư mục: /home/student/labtainer_xfer/TÊN_BÀI_LAB (tên tài khoản.TÊN_BÀI_LAB.lab)
- Kết thúc bài lab: Trên terminal đầu tiên sử dụng câu lênh sau để kết thúc bài lab: stoplab stego_code_scenechange_dct
- Khi bài lab kết thúc, một tệp lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.
- 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
- Khởi động lại bài lab: Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh: labtainer –r stego_code_scenechange_dct