

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH



MẬT MÃ VÀ AN NINH MẠNG (mở rộng)

**Báo cáo đề tài**  
**Steganography**

---

Giảng viên hướng dẫn: TS Nguyễn Đức Thái

Sinh viên: Huỳnh Đình Quang Khải - 1812612

Bùi Hoàng Lân - 1812783



# Mục lục

<b>1</b>	<b>Giới thiệu</b>	<b>2</b>
1.1	Steganography . . . . .	2
1.2	Lịch sử . . . . .	2
1.3	Một số kỹ thuật giấu dữ liệu phổ biến . . . . .	3
1.3.1	Steganography vật lý . . . . .	3
1.3.2	Steganography văn bản . . . . .	4
1.3.3	Steganography hình ảnh . . . . .	5
1.4	Ứng dụng thực tế . . . . .	5
<b>2</b>	<b>Ý tưởng thực hiện che giấu thông điệp trong một hình ảnh</b>	<b>6</b>
<b>3</b>	<b>Hiện thực</b>	<b>8</b>
3.1	Lựa chọn ngôn ngữ lập trình . . . . .	8
3.2	OpenCV . . . . .	8
3.3	Xây dựng ứng dụng steganography đơn giản . . . . .	10
3.3.1	Hiện thực kỹ thuật steganography . . . . .	10
3.3.2	Xây dựng ứng dụng web . . . . .	12
3.3.3	Mô phỏng quá trình sử dụng ứng dụng . . . . .	14
<b>4</b>	<b>Đánh giá cá nhân</b>	<b>18</b>

# 1 Giới thiệu

## 1.1 Steganography

Kỹ thuật giấu tin (Steganography) là phương pháp che giấu một thông điệp bí mật bên trong một thứ gì đó không bí mật. Những thứ được che giấu có thể là bất cứ thứ gì ta muốn. Ngày nay, Steganography được áp dụng chủ yếu trong việc che giấu một đoạn văn bản trong một hình ảnh, hay ẩn một đoạn văn bản hoặc một tập lệnh trong một văn bản Word hoặc Excel. Mục đích của steganography là để che giấu và đánh lừa. Đây là một hình thức giao tiếp bí mật và có thể liên quan đến việc sử dụng bất kỳ phương tiện nào để ẩn tin nhắn. Đây không phải là một dạng mật mã vì nó không liên quan gì đến việc xáo trộn dữ liệu hay sử dụng khóa. Thay vào đó, nó là một dạng ẩn dữ liệu và có thể được thực thi theo những cách thông minh. Trong khi mật mã là một ngành khoa học phần lớn cho phép quyền riêng tư, thì steganography là một phương pháp thực hành cho phép giữ bí mật - và lừa dối.

Bản chất steganography chỉ là một tên gọi chung cho các phương pháp giấu tin. Còn cụ thể hơn về các ứng dụng của kỹ thuật thuộc họ này thì là nhiều vô kể. Từ những năm trước công nguyên cho đến nay, steganography luôn là một chủ đề sôi nổi trong bảo mật dữ liệu. Từ steganography xuất phát từ tiếng Hy Lạp steganographia, kết hợp các từ steganós, có nghĩa là "được che đậy hoặc che giấu", và -graphia có nghĩa là "viết".

## 1.2 Lịch sử

Kỹ thuật steganography đầu tiên được phát triển ở nền văn minh Hy Lạp cổ đại vào khoảng 440 năm trước công nguyên. Người đứng vương quốc Hi Lạp, Histaeus, đã thiết kế một phiên bản của steganography liên quan đến các cách thức như cạo đầu nô lệ, xăm thông điệp lên da đầu nô lệ, đến khi tóc mọc lại sẽ hiển thị được thông điệp bí mật, người nô lệ cũng sẽ có nhiệm vụ truyền đi thông điệp bằng đường bộ. Người nhận sẽ cạo đầu nô lệ một lần nữa để tiếp nhận thông điệp và sẽ hồi âm theo cung một cách thức.

Trong cùng thời đại đó, một hình thức khác của steganography cũng được phát triển, liên quan đến một người tên Demeristus, ông đã truyền một thông điệp đến cho người Spartans cảnh báo về một cuộc xâm lăng sắp diễn ra của đội quân Xerxes. Một tấm phảng gần như trống trơn đã được gửi đi chứa bên trong nó là thông điệp bí mật.

Steganography tiếp tục được cải tiến trong những năm của thế kỉ 17, Sir Francis Bacon đã sử dụng biểu cảm khuôn mặt để mang đi những mảnh nhỏ của thông điệp. Từ những giai đoạn 1930, steganography đã được sử dụng trong chiến tranh thế giới lần thứ 2. Thậm chí người ta còn gọi cuộc chiến tranh này với một tên gọi khác là cuộc thi đua của kỹ thuật giấu tin (Steganography Competition). Về cách mà kỹ thuật này được áp dụng trong cuộc chiến khốc liệt trên toàn cầu này, để trao đổi thông tin tránh bị chiếm đoạt và phát hiện bởi quân Đức, vương quốc Anh đã cho ra một phương pháp để giấu tin trong một lá thư mà không tạo ra bất cứ nghi ngờ gì. Có thể kể đến điển hình như mực vô hình được sử dụng để viết thông tin trên các mảnh giấy để giấy hiển thị với người bình thường chỉ là những mảnh giấy trắng. Các chất lỏng như nước tiểu, sữa, giấm và nước trái cây đã được sử dụng, bởi vì khi mỗi một trong số các chất này bị nung nóng, chúng tối đi và mắt người có thể nhìn thấy được.

Steganography liên tục được phát triển theo suốt chiều dài lịch sử nhân loại. Trong thời kì chiến tranh, giữa cuộc chiến của quân cách mạng Mỹ, cả quân Anh và Mỹ đã sử dụng các loại mực vô hình được làm từ nhiều loại nguyên liệu bao gồm sữa giấm, nước ép trái cây, ... để che giấu thông điệp. Để giải mã cần có tác động của ánh sáng hoặc nhiệt. Bên cạnh đó là các cách thức sử dụng microdots, null cipher, ...

## 1.3 Một số kỹ thuật giấu dữ liệu phổ biến

### 1.3.1 Steganography vật lý

Những ý tưởng khởi đầu của steganography có thể được xếp vào mục này, khi những thông điệp được che giấu thông qua các tác nhân vật lý. Từ những thông điệp được giấu bằng cách cạo đầu, xăm trên người nô lệ, tới việc sử dụng một số loại 'mực bí mật', khiến cho người bình thường không thể nhận ra được.

Các kỹ thuật steganography sử dụng tác nhân vật lý khác có thể kể đến như:

- Tin nhắn ẩn được phân phối, theo một quy tắc hoặc khóa nhất định, dưới dạng các phần nhỏ hơn (ví dụ: từ hoặc chữ cái) trong số các từ khác của văn bản trang bìa ít đáng ngờ hơn. Dạng mật mã đặc biệt này được gọi là mật mã rỗng.

- Các thông điệp được viết bằng mã Morse trên sợi và sau đó được dệt kim vào một bộ quần áo do người chuyển phát nhanh mặc.
- Thông điệp được viết trên phong bì trong khu vực được bao phủ bởi tem bưu chính. Trong những ngày đầu của báo in, người ta thường trộn các kiểu chữ khác nhau trên một trang in vì máy in không có đủ bản sao của một số chữ cái trong một kiểu chữ. Do đó, một tin nhắn có thể bị ẩn bằng cách sử dụng hai hoặc nhiều kiểu chữ khác nhau, chẳng hạn như bình thường hoặc in nghiêng.

### 1.3.2 Steganography văn bản

Steganography dựa trên văn bản có thể được phân loại theo ba loại cơ bản - dựa trên định dạng, tạo ngẫu nhiên và thống kê và phương pháp ngôn ngữ.

Các phương pháp dựa trên định dạng đã sử dụng định dạng văn bản vật lý của văn bản như một nơi để ẩn thông tin. Nói chung, phương pháp này sửa đổi văn bản hiện có để ẩn thông điệp cần che giấu. Chèn khoảng trắng, các lỗi chính tả có chủ ý phân bố trên khắp văn bản, thay đổi kích thước phông chữ là một số trong nhiều phương pháp dựa trên định dạng đang được sử dụng trong nội dung ẩn văn bản. Các phương pháp dựa trên định dạng này có thể đánh lừa hầu hết mắt người nhưng nó không thể đánh lừa một khi hệ thống máy tính đã được sử dụng.

Cách tiếp cận thứ hai để tạo ký tự là sử dụng các thuộc tính thống kê của độ dài từ và tần suất chữ cái để tạo ra các “từ” (không có giá trị từ vựng) trông có vẻ có các thuộc tính thống kê giống như các từ thực tế trong một ngôn ngữ nhất định. Việc ẩn thông tin trong các chuỗi từ, các mục từ điển thực tế có thể được sử dụng để mã hóa một hoặc nhiều bit thông tin trên mỗi từ bằng cách sử dụng số mã ánh xạ giữa các mục từ vựng và chuỗi bit, hoặc bản thân các từ có thể mã hóa thông tin ẩn.

Loại cuối cùng là phương pháp ngôn ngữ đặc biệt xem xét các thuộc tính ngôn ngữ của văn bản được tạo ra và sửa đổi, thường sử dụng cấu trúc ngôn ngữ như một nơi cho các thông điệp ẩn. Trên thực tế, dữ liệu mật mã có thể được ẩn trong chính cấu trúc cú pháp.

### 1.3.3 Steganography hình ảnh

Kỹ thuật được sử dụng rộng rãi nhất hiện nay là ẩn các thông điệp bí mật thành một hình ảnh kỹ thuật số. Steganography này kỹ thuật khai thác điểm yếu của hệ thống thị giác con người. Mắt người không thể phát hiện sự thay đổi về độ chói của màu vectơ tại tập các điểm ảnh màu. Các điểm ảnh riêng lẻ có thể được biểu thị bằng phía tần số quang học cao hơn của quang phổ hình ảnh. Một bức tranh có thể được biểu thị bằng các đặc điểm như 'độ sáng', 'sắc độ', v.v. Mỗi đặc điểm này có thể được biểu thị bằng kỹ thuật số dưới dạng 1 và 0.

Một số phương pháp giấu thông điệp trong hình ảnh phổ biến:

- Ẩn tin nhắn trong các bit thấp nhất của tệp hình ảnh hoặc âm thanh nhiều. Có thể tìm thấy một cuộc khảo sát và đánh giá các tài liệu / kỹ thuật có liên quan về chủ đề kỹ thuật in ẩn hình ảnh kỹ thuật số tại đây.
- Ẩn dữ liệu trong dữ liệu được mã hóa hoặc trong dữ liệu ngẫu nhiên. Thông điệp cần che giấu được mã hóa, sau đó được sử dụng để ghi đè lên một phần của khối dữ liệu được mã hóa lớn hơn nhiều hoặc khối dữ liệu ngẫu nhiên.

## 1.4 Ứng dụng thực tế

Steganography với ý đồ che giấu thông điệp được sử dụng trong một số lĩnh vực thực tế, có thể kể đến một số ví dụ như :

- Một số máy in máy tính hiện đại sử dụng kỹ thuật in ẩn, bao gồm máy in laser màu nhãn hiệu Hewlett-Packard và Xerox. Các máy in thêm các chấm nhỏ màu vàng vào mỗi trang. Các dấu chấm khó nhìn thấy chứa số sê-ri máy in được mã hóa và tem ngày giờ.
- Trong một số lĩnh vực nhân tạo. Năm 2010, Cục Điều tra Liên bang cáo buộc rằng cơ quan tình báo nước ngoài của Nga sử dụng phần mềm steganography tùy chỉnh để nhúng các tin nhắn văn bản được mã hóa vào bên trong tệp hình ảnh để liên lạc nhất định với "điệp viên bất hợp pháp" (đặc vụ không có vỏ bọc ngoại giao) đóng quân ở nước ngoài.

## 2 Ý tưởng thực hiện che giấu thông điệp trong một hình ảnh

Việc che giấu thông điệp trong một hình ảnh có rất nhiều cách thực hiện. Trong bài báo cáo này, tôi sẽ sử dụng một phương pháp khá đơn giản, được dựa trên bài viết <https://www.geeksforgeeks.org/image-based-steganography-using-python/>. Sẽ có một chút khác biệt so với bản gốc để việc hiện thực bằng ngôn ngữ lập trình được đơn giản hơn.

Ý tưởng của steganography dựa trên hình ảnh này khá đơn giản, hình ảnh được cấu thành từ nhiều dữ liệu số, ở đây là pixel hay điểm ảnh, thứ được dùng để mô tả những gì chứa trong một bức ảnh, thường là các màu tạo nên một điểm ảnh. Biết được rằng mỗi một hình ảnh đều được hình thành từ nhiều điểm ảnh và mỗi điểm ảnh chứa đựng 3 loại giá trị biểu thị cho 3 loại màu cơ bản: đỏ, lục, lam.

### Mã hóa dữ liệu:

Mỗi kí tự dữ liệu sẽ được chuyển thành 8 bit nhị phân sử dụng các giá trị ASCII. Khi này, những điểm ảnh sẽ lần lượt được đọc từ đầu tới cuối theo từng nhóm 3, vậy mỗi nhóm 3 điểm ảnh sẽ chứa tổng cộng 9 giá trị. 8 giá trị đầu tiên của mỗi 9 giá trị đó sẽ được sử dụng để lưu trữ dữ liệu binary. Mỗi giá trị sẽ được chuyển thành chẵn hoặc lẻ dựa theo quy luật sau:

- Nếu bit nhị phân là 1, giá trị trong điểm ảnh sẽ được chuyển thành lẻ nếu ban đầu nó là chẵn bằng cách trừ đi 1.
- Nếu bit nhị phân là 0, giá trị trong điểm ảnh sẽ được chuyển thành chẵn nếu ban đầu nó là lẻ bằng cách trừ đi 1.

Để thuận tiện cho việc giải mã, ta thêm vào cuối chuỗi một kí tự ASCII ít được sử dụng để đánh dấu kết thúc thông điệp, sau đó tiến hành mã hóa dữ liệu vào trong hình ảnh theo giải thuật được mô tả ở trên. Trong bài nghiên cứu này, ta chọn kí tự '|' (vì kí tự này vốn ít được xài trong các văn bản thường ngày).

Ví dụ:

Thông điệp được gửi đi là "ai"

Ảnh là một ma trận 3 chiều được chuyển về 2 chiều với chiều thứ nhất là

tổng số lượng điểm ảnh, chiều thứ mang giá trị 3 với ý nghĩa là 3 màu cơ bản đỏ, lục, lam để tiện cho việc hiện thực giải thuật:

[(27, 64, 164), (248, 244, 194), (174, 246, 250), (149, 95, 232), (188, 156, 169), (71, 167, 127), (132, 173, 97), (113, 69, 206), (255, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175)]

Thông điệp được gửi đi sẽ được thêm vào cuối kí tự '|', trở thành "ai|".

Giá trị ASCII của 'a' là 97, tương đương với chuỗi 8 bit nhị phân 01100001.

Nhóm 3 điểm ảnh đầu tiên (27, 64, 164), (248, 244, 194), (174, 246, 250) sẽ được sử dụng để mã hóa kí tự 'a' đầu tiên trong thông điệp. Giá trị 27 với bit đầu tiên là 0 tương ứng với trường hợp bit chẵn, giá trị lẻ vậy giá trị sẽ được trừ 1 để mang tính chẵn, vậy  $27 - 1 = 26$ , tương tự cho các giá trị còn lại trong 3 điểm ảnh, trừ giá trị cuối cùng (250), ta được chuỗi điểm ảnh như sau (26, 63, 163), (248, 244, 294), (174, 245, 250). Tương tự, kí tự 'i' cũng sẽ được mã hóa theo cách thức như trên. Sau khi cả 3 kí tự đều đã được mã hóa ta được chuỗi các điểm ảnh chứa các giá trị màu tương ứng như sau:

[(26, 63, 163), (248, 244, 294), (174, 245, 250), (148, 95, 231), (188, 155, 168), (70, 167, 126), (132, 173, 97), (112, 69, 206), (254, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175)]

Sau đó ta thực hiện chuyển ma trận các điểm ảnh trên về đúng với định dạng ban đầu (3 chiều) và gửi đi, vậy ta thành công gửi đi thông điệp đã được mã hóa trong hình ảnh mà không làm bức ảnh thay đổi nhiều về mặt thị giác, bởi vì các giá trị màu trên từng điểm ảnh chỉ sai biệt so với giá trị ban đầu của chúng lớn nhất là 1 đơn vị (trên 255 giá trị có thể có của mỗi màu).

### **Giải mã dữ liệu:**

Để giải mã dữ liệu, mỗi 3 điểm ảnh được đọc tại một thời điểm, cho đến trước giá trị cuối cùng (giá trị thứ 9 của mỗi 3 điểm ảnh), để có thể giải mã ra 8 bit tương ứng với cách thức mã hóa, tiếp tục đến 3 điểm ảnh tiếp theo cho đến khi gặp được kí tự kết thúc (kí tự '|'). Như vậy mỗi 8 bit sẽ được chuyển thành một kí tự ASCII, từ đó ta có thể thành công lấy lại thông điệp được mã hóa trong bức ảnh.

Ví dụ:

Giải mã chuỗi với ảnh được cho, ta chuyển ảnh từ định dạng ban đầu là ma trận 3 chiều, sang ma trận 2 chiều với cùng phương thức mã hóa, cho rằng ta được ma trận các giá trị sau:



[(26, 63, 163), (248, 244, 294), (174, 245, 250), (148, 95, 231), (188, 155, 168), (70, 167, 126), (132, 173, 97), (112, 69, 206), (254, 29, 213), (53, 153, 220), (246, 225, 229), (142, 82, 175)]

Tại đây, mỗi 3 điểm ảnh sẽ được chuyển thành một kí tự ASCII, với (26, 63, 163), (248, 244, 294), (174, 245, 250), ta bắt đầu với 26, đây là giá trị chẵn vậy bit trái cùng (most significant bit) là 0, kể đến giá trị 63 cho ta bit kể bit trái cùng là 1, tương tự cho đến hết 8 giá trị của 3 điểm ảnh, ta được 01100001, tương ứng với kí tự 'a' trong bảng mã ASCII.

Tiếp tục cho đến khi gặp kí tự '|', ta dừng việc giải mã, và nhận về chuỗi các kí tự "ai".

Vậy ta thành công giải mã và lấy về được thông điệp gốc.

## 3 Hiện thực

### 3.1 Lựa chọn ngôn ngữ lập trình

Dựa trên những gì đã phân tích về ý tưởng thực hiện steganography. Tôi quyết định lựa chọn `python` là ngôn ngữ lập trình để xây dựng ứng dụng. Giải thích về lựa chọn trên, `python` là một ngôn ngữ đang phổ biến hiện nay cho công tác xử lí ảnh, với nhiều thư viện phổ biến đang được hỗ trợ cực nhiều, cộng đồng sử dụng cũng đang tăng lên từng ngày. Ngoài ra, việc xây dựng ứng dụng để tạo ra một không gian sử dụng kỹ thuật này thông qua tương tác máy tính đối với `python` cũng được hỗ trợ bằng framework khá nhiều.

Với cách code đơn giản, cộng với sự hỗ trợ bởi nhiều thư viện, việc sử dụng `python` là ngôn ngữ hiện thực kỹ thuật steganography thông qua ứng dụng có lẽ là một lựa chọn hợp lí.

### 3.2 OpenCV

OpenCV (Open Source Computer Vision Library) được bắt đầu từ Intel năm 1999 bởi Gary Bradsky, là một thư viện mã nguồn mở cung cấp nhiều tiện ích cho các lĩnh vực như xử lí ảnh và học máy. Nó có trên các giao diện C++, C, `python` và `java` và hỗ trợ Windows, Linux, Mac OS, iOS và Android.

OpenCV kết hợp với `python` mang lại những tiện lợi về mặt cú pháp, cũng

như xử lý xử liệu ảnh một cách linh hoạt bằng cách kết hợp với thư viện numpy có thể giúp ta thao tác với những hình ảnh dễ dàng hơn bao giờ hết.

Để hiện thực kỹ thuật steganography được dễ dàng hơn, tôi sẽ nhờ vào sự hỗ trợ của OpenCV trong việc đọc và ghi hình ảnh. Ngoài ra, OpenCV cũng sẽ giúp tôi tương tác trực tiếp với giá trị từng điểm ảnh trong hình. Có rất nhiều định dạng hình ảnh khác nhau, nghĩa là cấu trúc file hình ảnh đó cũng khác nhau, việc lấy ra được giá trị điểm ảnh với từng định dạng có cũng sẽ khác nhau. Nhờ vào sức mạnh của OpenCV, ta sẽ không cần phải quan tâm tới vấn đề này mà đi thẳng luôn vào việc che giấu thông điệp qua giá trị điểm ảnh.

```
"""
read_image:
    color: True => reading as color image (default) in RGB format
    color: False => reading as gray
"""
def read_image(filename, color=True):
    # print(filename)
    image = cv2.imread(filename, 1 if color else 0)
    return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

"""
if image_data is in color format then RGB must be used
"""
def write_image(filename, image_data):
    image_data = cv2.cvtColor(image_data, cv2.COLOR_RGB2BGR)
    cv2.imwrite(filename, image_data)
```

Hình 1: Sử dụng OpenCV hỗ trợ đọc ghi file hình ảnh

Hình 1 cho thấy hai hàm sử dụng OpenCV để tiến hành đọc hay ghi file hình ảnh. Hàm `read_image(filename, color=True)` tiến hành đọc hình ảnh theo tên file và trả về một mảng 3 chiều theo định dạng (*row, col, dim*), với ý nghĩa lần lượt là số điểm ảnh theo chiều dọc, số điểm ảnh theo chiều ngang, và số chiều của màu (là 3 nếu là ảnh màu, và là 1 nếu là ảnh xám - ảnh trắng đen), giá trị trả về của hàm này rất thích hợp để ta trực tiếp hiện thực kỹ thuật steganography. Hàm `write_image(filename, image_data)` nhận tham số là tên file muốn lưu và mảng điểm ảnh 3 chiều có định dạng vừa nêu để lưu lại hình ảnh.

## 3.3 Xây dựng ứng dụng steganography đơn giản

### 3.3.1 Hiện thực kỹ thuật steganography

#### Che giấu thông điệp:

Để che giấu thông điệp vào hình ảnh. Đầu tiên ta tiến hành thêm kí tự '|' vào cuối thông điệp gốc. Thực hiện đọc hình theo chế độ RGB. Chuyển ma trận 3 chiều thành 2 chiều với một chiều là tổng số điểm ảnh, chiều còn lại gồm 3 giá trị đỏ, lục, lam.

```
text += TERMINATE_CHARACTER

image = read_image(PATH + img_name, True)
#image = (row, col, dim)
print(image.shape)
print(image.shape[0] * image.shape[1])
# print(image[0])

original_width = image.shape[0]
original_height = image.shape[1]
image = image.reshape((image.shape[0] * image.shape[1],3))
```

Hình 2: Đọc ảnh và chuyển đổi chiều ma trận điểm ảnh

Sử dụng một số hàm phụ trợ dưới đây, sẽ giúp ta chuyển đổi thông điệp từ dạng chuỗi sang một mảng các số nhị phân và ngược lại. Điều này giúp ta dễ dàng hơn trong việc so sánh từng bit của kí tự với mỗi giá trị điểm ảnh của hình.

```
def convert_decimal_to_bin(num):
    if num <= 0:
        return ''
    if num % 2 == 0:
        return convert_decimal_to_bin(num // 2) + '0'
    return convert_decimal_to_bin(num // 2) + '1'

def convert_bin_to_decimal(bin):
    res = 0
    for i in range(len(bin)):
        res += int(bin[len(bin) - i - 1]) * 2**i
    return chr(res)
```

Hình 3: Một số hàm chuyển đổi phụ trợ

Với những gì đã chuẩn bị, ta tiến hành che giấu thông điệp vào một hình

ảnh đã được đọc. Vòng lặp chạy qua từng chuỗi nhị phân tương ứng với mỗi kí tự trong tệp. Từng bit trong chuỗi nhị phân đó sẽ được so sánh với giá trị điểm ảnh của hình. Kết thúc vòng lặp, ta chuyển đổi lại chiều của ma trận điểm ảnh từ 2 chiều thành 3 chiều ban đầu và lưu nó với tên mới, trả về kết quả là tên file ảnh mới. Chi tiết về xử lí trên điểm ảnh được thể hiện trong hình sau.

```
#convert a text to a binary ascii array
ascii_num_array = [ord(x) for x in text]
bin_num_array = [convert_decimal_to_bin(x) for x in ascii_num_array]
print(bin_num_array)

for j in range(len(bin_num_array)):
    num_leading_0s = 8 - len(bin_num_array[j])
    # Leading 0s -> even
    for k in range(num_leading_0s):
        if image[j * 3 + k // 3][k % 3] % 2 != 0: image[j * 3 + k // 3][k % 3] -= 1
    for k in range(len(bin_num_array[j])):
        idx = j * 3 + (k + num_leading_0s) // 3
        idx1 = (k + num_leading_0s) % 3
        if (image[idx][idx1] % 2 == 0 and bin_num_array[j][k] == '1') or \
            (image[idx][idx1] % 2 != 0 and bin_num_array[j][k] == '0'):
            image[idx][idx1] -= 1

image = image.reshape(original_width, original_height, 3)
```

Hình 4: Che giấu thông điệp

### Tìm thông điệp:

Các công việc khởi trị ban đầu cho việc tìm thông điệp trong một hình cũng tương tự như che giấu thông điệp. Hình ảnh cần tìm thông điệp cũng sẽ được đọc để lấy ma trận điểm ảnh 3 chiều. Chuyển ma trận 3 chiều thành 2 chiều với một chiều là tổng số điểm ảnh, chiều còn lại gồm 3 giá trị đỏ, lục, lam.

Việc tìm thông điệp trong một hình duyệt trên từng bộ 3 điểm ảnh, tìm ra kí tự mà bộ 3 đó thể hiện cho tới khi tìm ra kí tự '|' (là kí tự đánh dấu kết thúc chuỗi mà ta quy định), hoặc cho tới khi chạy hết ma trận điểm ảnh. Trả về kết quả là bản rõ đã được rút trích từ hình được đọc vào ban đầu. Chi tiết về việc tìm thông điệp trong hình được thể hiện trong hình sau.

```
plain_text = ""
bin_char = ""
for i in range(image.shape[0]):
    done = False
    for j in range(3):
        if (image[i, j] % 2 == 0):
            bin_char += '0'
        else:
            bin_char += '1'

        if len(bin_char) == 8:
            char = convert_bin_to_decimal(bin_char)
            # print(char, "-", bin_char)
            bin_char = ""
            if char != TERMINATE_CHARACTER:
                plain_text += char
            else:
                done = True
                break
    if done:
        break
return plain_text
```

Hình 5: Tìm thông điệp trong một hình ảnh

### 3.3.2 Xây dựng ứng dụng web

Kỹ thuật steganography đã được hiện thực xong, ta sẽ tiếp tục tiến hành tạo ra một ứng dụng web đơn giản, nơi mà người dùng chỉ đơn giản gửi hình và thông điệp để có được kết quả thông điệp. Với python là ngôn ngữ lập trình được sử dụng, tôi sẽ tiếp tục nhờ vào sự hỗ trợ để xây dựng ứng dụng đơn giản hơn. Framework Django thích hợp trong trường hợp này. Django là một framework Web Python cấp cao (high-level), tạo điều kiện phát triển ứng dụng nhanh chóng với thiết kế sạch sẽ, thực dụng. Được xây dựng bởi các nhà phát triển có kinh nghiệm, nó xử lý phần lớn sự phức tạp của quá trình phát triển Web, vì vậy bạn có thể tập trung vào việc viết ứng dụng của mình mà không cần phải phát minh lại bánh xe. Django là nguồn mở và miễn phí.

Ứng dụng web với Django sẽ tập trung vào việc tương tác với người dùng để tiến hành thực hiện kỹ thuật steganography. Ý đồ chính là sẽ nhận file và thông điệp từ người dùng để chạy giải thuật che giấu thông điệp, sau đó trả về đường dẫn để người dùng tải về hình ảnh đã được che giấu thông điệp. Ngược lại, người dùng cũng có thể đưa một hình ảnh lên để xem thông điệp được che giấu bên trong nó.

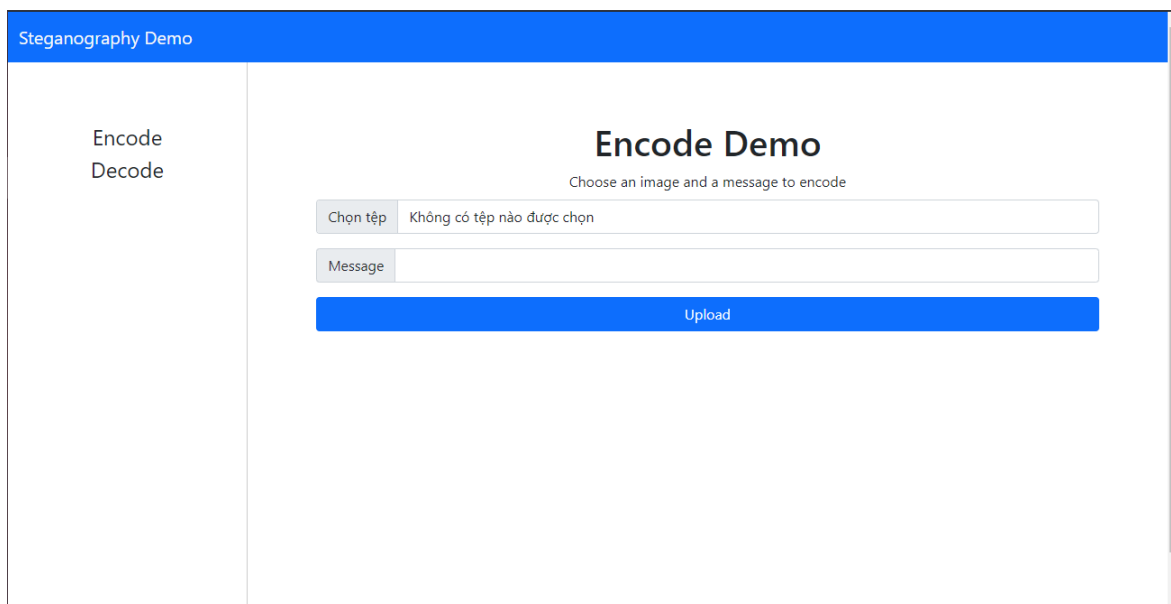
```
def index(request):
    context = {}
    if request.method == "POST":
        if request.POST['demoMode'] == 'encode':
            uploaded_file = request.FILES['image']
            fs = FileSystemStorage()
            file = fs.save(uploaded_file.name, uploaded_file)
            text = request.POST['demoText']
            print(file)
            file_res = steg.steganography(file, text)
            context['url'] = fs.url(file_res)

        elif request.POST['demoMode'] == 'decode':
            uploaded_file = request.FILES['image']
            fs = FileSystemStorage()
            file = fs.save(uploaded_file.name, uploaded_file)
            stegano_res = steg.steganography_decode(file)
            context['decode'] = stegano_res
    return render(request, 'pages/base.html', context)
```

Hình 6: Hàm xử lý chính trong Django

Hình trên cho thấy cách xử lý trong ứng dụng xây dựng bằng Django, mỗi lần người dùng cần ứng dụng thực hiện kỹ thuật steganography sẽ phải gửi hình ảnh tương ứng, hoặc gửi thêm thông điệp nếu thực hiện che giấu thay gì giải mã. Ứng dụng sẽ xử lý POST request trên, lấy ra dữ liệu tương ứng và thực hiện gọi hàm thích hợp che giấu hay giải mã thông điệp từ một hình ảnh.

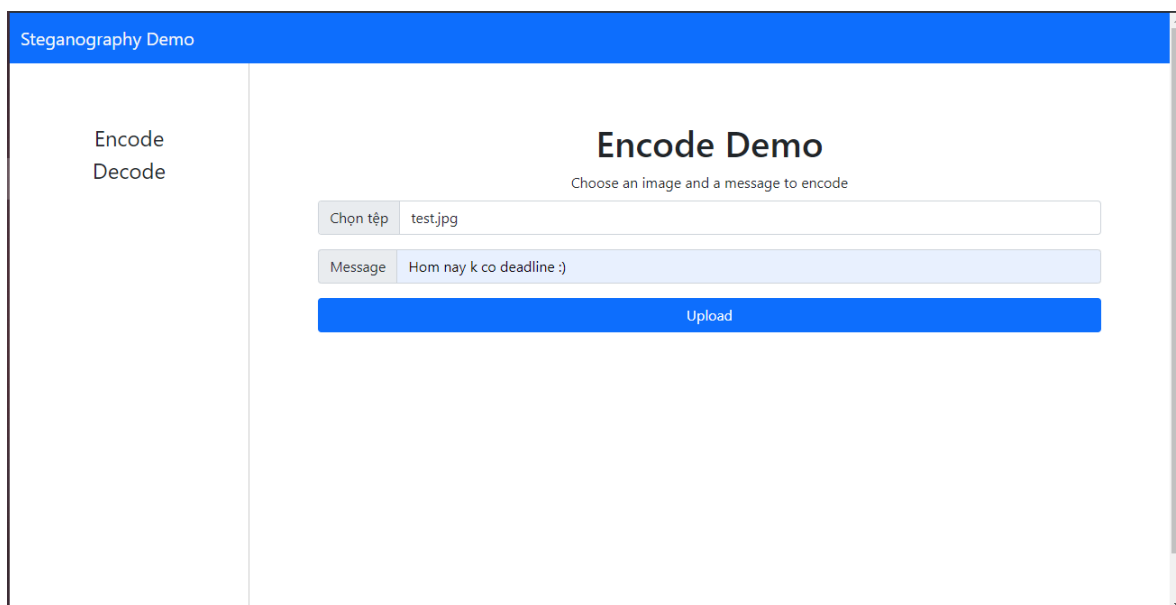
### 3.3.3 Mô phỏng quá trình sử dụng ứng dụng



The screenshot shows a web application titled "Steganography Demo". On the left, there is a sidebar with two buttons: "Encode" and "Decode". The main area is titled "Encode Demo" and contains the instruction "Choose an image and a message to encode". Below this, there are two input fields: "Chọn tệp" (Choose file) with the placeholder text "Không có tệp nào được chọn" (No file selected), and "Message" with an empty text box. At the bottom of the main area is a blue "Upload" button.

Hình 7: Giao diện trang chủ

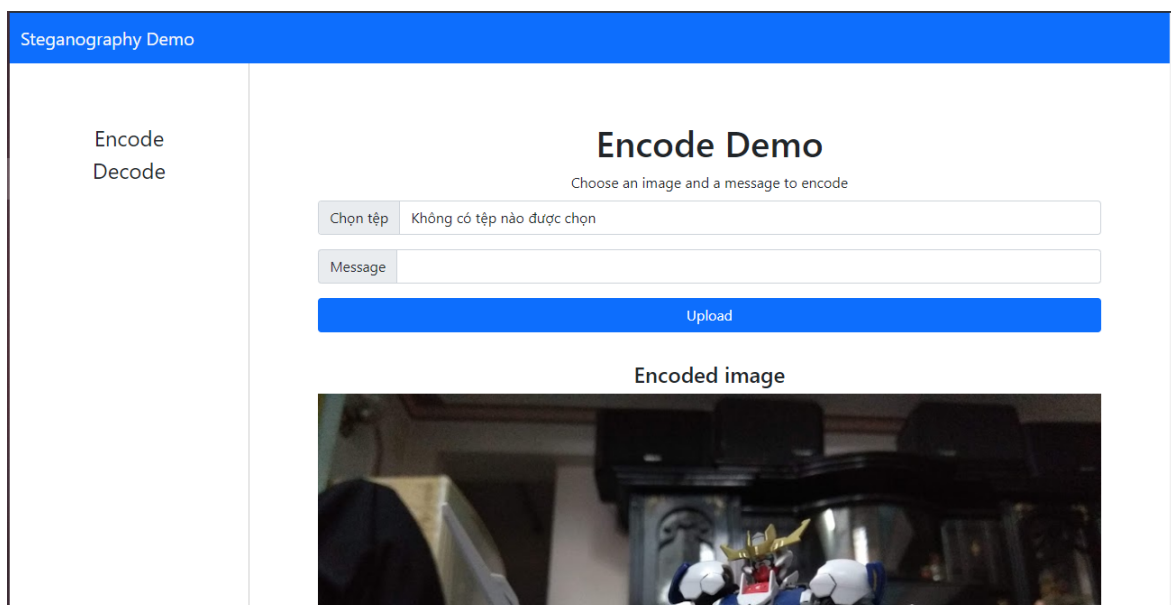
Ứng dụng bắt đầu với giao diện web được thiết kế bằng `bootstrap`, cột bên trái hiển thị những lựa chọn mà ta có thể chọn với Encode là che giấu thông điệp trong một hình ảnh hay Decode là tìm kiếm thông điệp được che giấu trong một hình ảnh đầu vào.



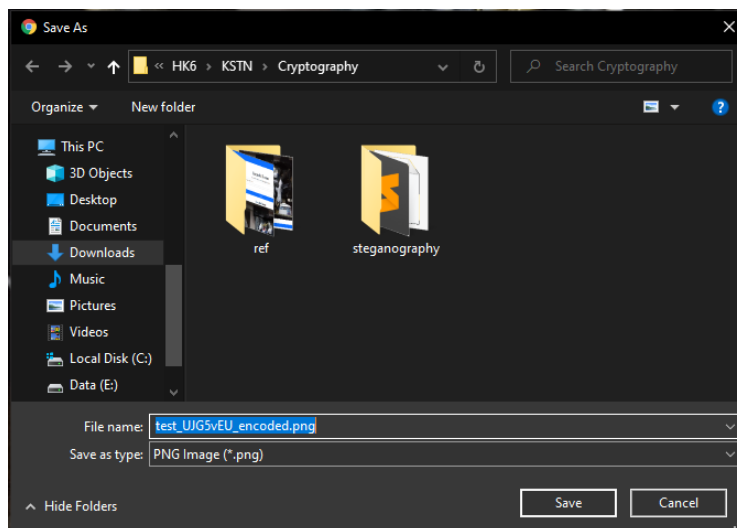
Hình 8: Chọn file và nhập thông điệp cần che giấu

Ta thực hiện che giấu thông điệp vào một hình ảnh, ở đây file ảnh được dùng để chứa thông điệp được che giấu có tên 'test.jpg'. Nhập thông điệp và chọn 'Upload' để ứng dụng che giấu thông điệp vào hình, ta được kết quả như hình sau. Hình ảnh có chứa thông điệp được hiển thị sau khi che giấu thành công, ngay dưới hình là một nút bấm để ta có thể tải ảnh về máy.





Hình 9: Kết quả trả về



Hình 10: Tải ảnh vừa được thay đổi để chứa thông điệp

Tên file hình ảnh chứa thông điệp được thay đổi so với ban đầu. So sánh hình này với hình gốc, ta hoàn toàn không thể phân biệt được sự khác biệt vì sự thay đổi giá trị điểm ảnh là rất nhỏ, mắt người không thể nhận ra.



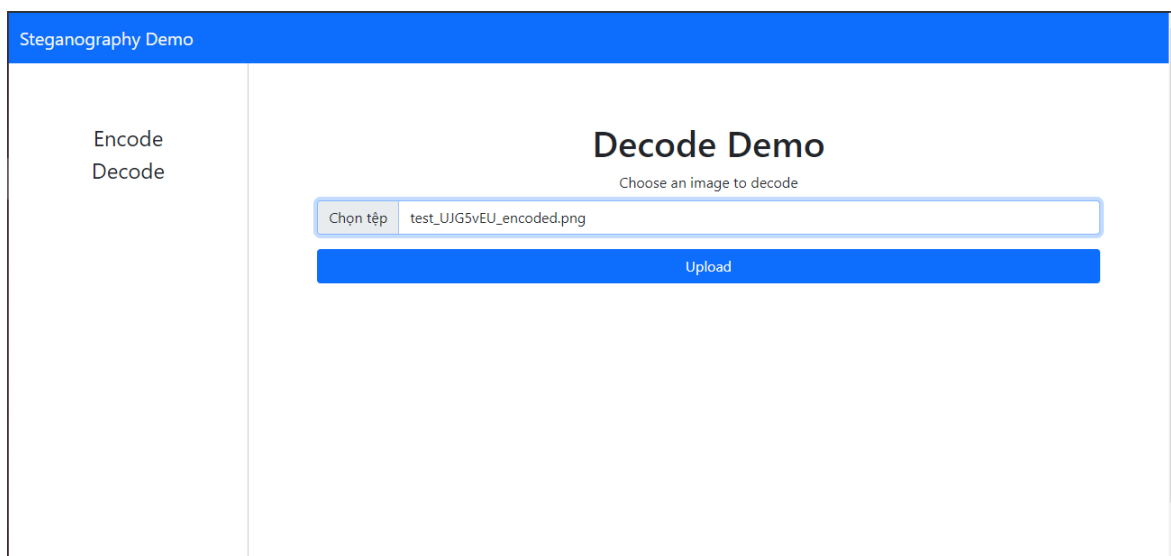
(a) Hình ảnh có chứa thông điệp



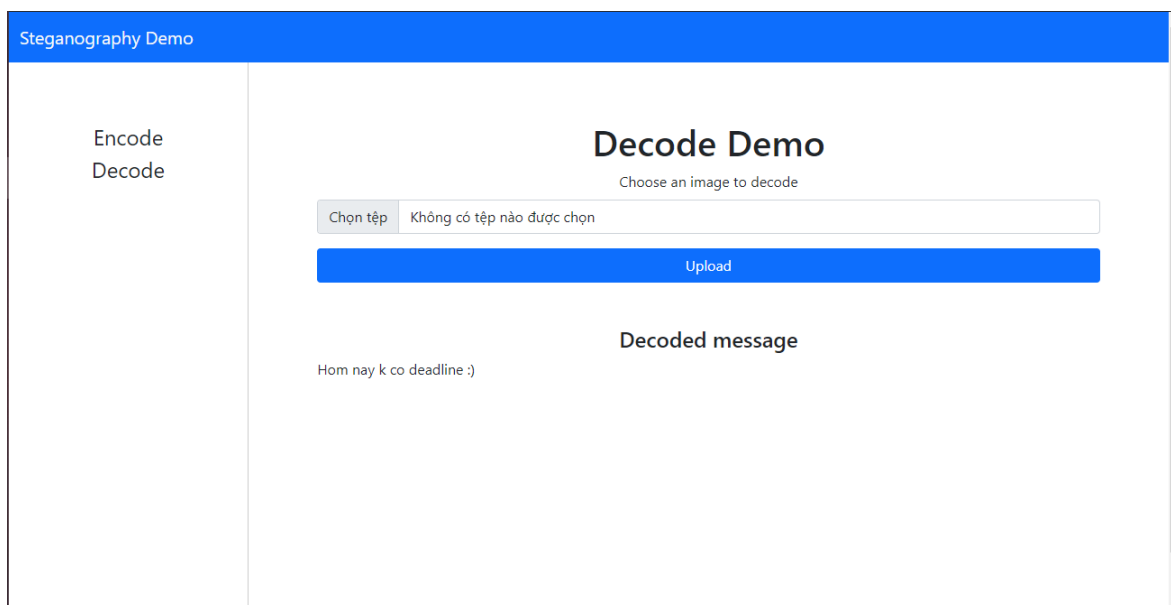
(b) Ảnh gốc

Hình 11: So sánh hai hình ảnh sau khi thực hiện steganography

Ngoài ra, ứng dụng cũng sẽ giúp ta tìm thông điệp được chứa trong một hình ảnh, nếu cách mà thông điệp đó che giấu trùng với cách mà ứng dụng dùng. Ta có thể chọn vào mục Decode ở cột bên trái, tải ảnh cần tìm ra thông điệp và nhấn 'Upload', ứng dụng sẽ trả về thông điệp mà ảnh đó đang chứa.



Hình 12



Hình 13

## 4 Đánh giá cá nhân

Cách hiện thực kỹ thuật steganography để che giấu thông điệp vào một hình ảnh trong bài báo cáo này hy vọng đã cho thấy một cách tiếp cận đơn giản, dễ hiện thực. Trong thực tế, việc che giấu này thường gắn thêm với một giá trị khóa nào đó nhằm tăng tính bí mật của thông điệp. Ngoài ra giá trị khóa này cũng là một cách để ta kiểm tra nhanh rằng hình ảnh đó có chứa thông điệp mà ta cần xem hay không.

Trong quá trình hiện thực steganography nhờ sự giúp đỡ của OpenCV, tôi nhận thấy hàm `imwrite()` của thư viện này ghi lại hình với giá trị điểm ảnh sai lệch so với ban đầu đối với hình ảnh có đuôi '.jpg'. Điều này không có ảnh hưởng với mắt người xem nhưng sẽ làm cho việc che giấu thông điệp bị sai, dẫn tới không thể thực hiện được. Cách xử lý cho vấn đề này là lưu dưới định dạng một file hình ảnh khác (cách này được sử dụng trong báo cáo), định dạng nên dùng ở đây là '.png'.

Đánh giá cá nhân về việc hiện thực steganography trong bài báo cáo. Giải thuật cho kết quả trả về nhanh, chính xác. Tuy nhiên, việc sử dụng '|' để kết thúc thông điệp cũng như đánh dấu để kết thúc việc tìm thông điệp trong hình không phải là một phương pháp quá hay. Với những hình ảnh không chứa thông điệp được che giấu theo kiểu này, kết quả cho ra vô cùng kì lạ với

những kí tự lạ, hiệu suất cho ra cũng không tốt khi phải duyệt hết điểm ảnh trong một hình (hình càng nét thì điểm ảnh càng nhiều, càng mất thời gian duyệt). Ứng dụng có thể phát triển theo hướng thay đổi một số điểm trong việc che giấu và tìm thông điệp để cho ra kết quả và hiệu suất tốt hơn.

Về steganography với mã hóa, trong khi hai quy trình này thường được thực hiện riêng biệt, chúng cũng có thể được kết hợp với nhau để đạt được những lợi thế từ cả hai lĩnh vực. Nếu bạn muốn che giấu sự thật rằng giao tiếp đang diễn ra, nhưng cũng bảo vệ tin nhắn trong trường hợp nó bị phát hiện, trước tiên bạn có thể mã hóa nó và sau đó che giấu nó bằng kỹ thuật steganography.

## Tài liệu tham khảo

- [1] James Stanger, July 6th, 2020 - *The Ancient Practice of Steganography: What Is It, How Is It Used and Why Do Cybersecurity Pros Need to Understand It* - url: <https://www.comptia.org/blog/what-is-steganography>
- [2] Ashwin Goel , August 20th, 2020 - *Image based Steganography using Python* - url: <https://www.geeksforgeeks.org/image-based-steganography-using-python/>
- [3] Góc của Chung , January 25th, 2021 - *Nghệ thuật bảo mật: STEGANOGRAPHY- KỸ THUẬT GIẤU TIN VÀ NHỮNG ỨNG DỤNG* - url: <https://www.goccuachung.com/nghe-thuat-bao-mat-steganography-ky-thuat-giau-tin-va-nhung-ung-dung/>
- [4] Nguyễn Hồng Sơn , October 29th, 2018 - *Giới thiệu kỹ thuật giấu tin trong ảnh Steganography* - url: <https://securitydaily.net/gioi-thieu-ky-thuat-giau-tin-trong-anh-steganography/>
- [5] Phan Văn Tấn , January 15th, 2020 - *Tổng quan về Django* - url: <https://viblo.asia/p/tim-hieu-ve-django-framework-ho-tro-pythhon-trong-lap-trinh-web-QpmlxbkZrd>