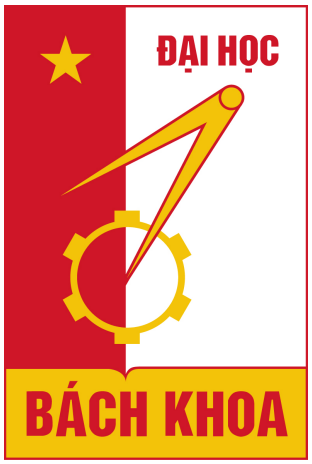
ĐẠI HỌC BÁCH KHOA HÀ NỘI

KHOA TOÁN - TIN



ĐỒ ÁN I

PHÂN TÍCH CHUNG VỀ GIAO TIẾP BÍ MẬT THÔNG QUA CHE GIẤU THÔNG TIN TRONG HÌNH ẢNH

Giảng viên hướng dẫn: TS. Vũ Thành Nam

Sinh viên thực hiện: Hoàng Anh Tuấn

MSSV: 20216899

Lớp: Toán tin - 01 - K66

Hà Nội, tháng 06 năm 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục tiêu và nội dung của đồ án

(a) Mục tiêu:

(b) Nội dung:

2. Kết quả đạt được

(a)

(b)

(c)

3. Ý thức làm việc của sinh viên

(a)

(b)

(c)

Hà Nội, ngày ... tháng ... năm 2024

Giảng viên hướng dẫn

Vũ Thành Nam

Lời cảm ơn

Lời đầu tiên, em xin gửi lời cảm ơn chân thành và sâu sắc tới thầy – TS. Vũ Thành Nam vì đã tạn tình hỗ trợ, giúp đỡ em trong quá trình thực hiện đồ án. Qua thời gian làm việc với thầy trong đồ án lần này, em đã học thêm được nhiều kiến thức hữu ích về Steganography hay những kiến thức về giấu tin trong ảnh. Những kiến thức và trải nghiệm này là một bài học quý báu để em có thể phát triển bản thân hơn cho các báo cáo lần sau. Em rất mong sẽ có nhiều cơ hội làm việc với thầy hơn trong tương lai.

Bên cạnh đó, em cũng xin gửi lời cảm ơn đến các thầy cô trong Khóa Toán-Tin, Đại học Bách khoa Hà Nội, đặc biệt là thầy – TS. Trần Ngọc Khuê vì đã cung cấp cho em những kiến thức nền tảng thống kê, phân phối đủ vững chắc để hoàn thiện đồ án này. Đồ án là kết quả của nhiều nền tảng kiến thức từ nhiều học phần khác nhau, nhờ sự hướng dẫn và giảng dạy tận tình của các thầy cô, em đã có cơ hội áp dụng những kiến thức này vào đồ án một cách hiệu quả.

Trong quá trình hoàn thiện đồ án, dù đã chuẩn bị khá kỹ lưỡng nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Em rất mong có thể nhận được nhiều góp ý từ các thầy cô để đồ án có thể hoàn thiện hơn, qua đó em cũng rút ra được kinh nghiệm và phát triển bản thân hơn.

Em xin chân thành cảm ơn!

Tóm tắt nội dung đồ án

Steganography là một kỹ thuật được sử dụng để che giấu sự hiện diện của dữ liệu trong môi trường kỹ thuật số. Bài viết này cung cấp những phân tích chuyên sâu về các phương pháp che giấu thông tin hiện đại, đánh giá tính hiệu quả của chúng thông qua các phép đo thực nghiệm và các tiêu chí cụ thể. Tôi khám phá sự mạnh mẽ và bảo mật của các kỹ thuật giấu tin khác nhau, bao gồm hình mờ kỹ thuật số và phương pháp mật mã, để xác định các hệ thống phù hợp nhất nhằm nâng cao khả năng giấu tin. Tôi đi sâu vào các thực tiễn lịch sử và hiện đại, so sánh các kỹ thuật miền không gian và biến đổi, đồng thời kiểm tra tiềm năng của các phương pháp End-of-File (EOF) và Spread Spectrum. Ngoài ra, tôi còn đánh giá và cải thiện tính hiệu quả của phân tích Chi-Square trong việc phát hiện các sửa đổi bit ít quan trọng nhất (LSB). Nghiên cứu này nhấn mạnh tầm quan trọng của kỹ thuật giấu tin trong việc bảo vệ thông tin số, nêu bật các ứng dụng của nó trong truyền thông an toàn, xác thực phương tiện và điều tra pháp y. Nghiên cứu kết thúc bằng cách đánh giá các kỹ thuật phân tích mật mã và tác động của mã hóa đối xứng và bất đối xứng đến tính hiệu quả của kỹ thuật giấu tin.

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

MỤC LỤC

GIỚI THIỆU CHUNG

Trong thế giới hiện đại, bảo mật thông tin là một phần quan trọng của an ninh không gian mạng, đảm bảo dữ liệu bí mật không bị đánh cắp, thay đổi hoặc hư hỏng trái phép. Đó không chỉ là vấn đề quốc gia mà còn là vấn đề liên quan đến lợi ích của người dân.

Một trong những phương pháp đảm bảo an toàn cho dữ liệu số là sử dụng kỹ thuật giấu tin. Điều này đặc biệt có thể áp dụng cho các dữ liệu đa phương tiện khác nhau, chẳng hạn như hình ảnh, âm thanh, video, v.v. Việc sử dụng kỹ thuật giấu tin để che giấu thông tin là một lĩnh vực khoa học ngày càng trưởng thành và đang phát triển.

Steganography, phương pháp che giấu thông tin trong phương tiện kỹ thuật số, đã thu hút được sự chú ý đáng kể trong vài thập kỷ qua. Các khái niệm cơ bản của kỹ thuật giấu tin liên quan đến việc nhúng dữ liệu bí mật vào trong phương tiện truyền tải theo cách mà sự tồn tại của dữ liệu được che giấu khỏi người quan sát. Các kỹ thuật ẩn dữ liệu đã phát triển, từ sửa đổi bit có ý nghĩa nhỏ nhất (LSB) đơn giản [1] đến các thuật toán phức tạp hơn như mã hội chứng-trellis cho hình ảnh JPEG [2]. Hơn nữa, các ứng dụng của steganography trải rộng trên nhiều lĩnh vực khác nhau, bao gồm thông tin liên lạc an toàn, hình mờ kỹ thuật số và xác thực phương tiện [3]. Nghiên cứu trong lĩnh vực này tiếp tục phát triển nhằm cải thiện tính mạnh mẽ và khả năng không bị phát hiện của các phương pháp steganographic [4].

Cái nhìn tổng quan

"Những gì bạn nhìn thấy chưa chắc đã là toàn bộ sự thật." Quả thực, với công nghệ hiện tại, việc che giấu thông tin trong hình ảnh, âm thanh hoặc thậm chí các tập tin có thể gây ngạc nhiên.

So với thị giác hay thính giác của con người, hình ảnh và âm thanh luôn chứa đựng nhiều hơn những gì chúng ta cảm nhận được. Vì lý do này, trong nhiều thập kỷ, con người luôn cố gắng nghĩ ra các phương pháp để che giấu thông tin, dẫn đến lịch sử của Steganography [5].

Các giai đoạn lịch sử

Thời cổ đại

Lịch sử vào những năm 400 trước Công nguyên về truyền thống viết bí mật. Trong các bài viết của mình, ông đã thảo luận về những xung đột giữa Hy Lạp và Ba Tư. Một vị vua tên là Histiaeus đã khuyến khích Aristagoras xứ Miletus nổi dậy chống lại vua Ba Tư. Anh ta thường cắt tóc cho những người hầu thân tín nhất của mình và xăm những thông điệp bí mật lên da đầu của họ, chờ tóc mọc lại. Sau đó, những người hầu có thể tự do đi lại qua biên giới mà không mang theo bất cứ thứ gì đáng ngờ.

Một ví dụ khác về kỹ thuật steganography liên quan đến việc sử dụng Cardano Grille, được đặt theo tên của nhà phát minh Girolamo Cardano, Lưới tản nhiệt Cardan được làm từ một tờ giấy hoặc giấy da khá cứng hoặc từ kim loại mỏng. Tờ giấy được coi là thể hiện các dòng chữ viết tay và các khu vực hình chữ nhật được cắt ra ở những khoảng cách tùy ý giữa các dòng này. Khi lưới được đặt trên văn bản in, thông điệp cố ý ẩn có thể được tiết lộ. Trong phạm vi các kỹ thuật liên quan đến Cardano Grille, các phương pháp giấu tin cổ điển bao gồm ghim vào văn bản (ví dụ: báo) và viết lên văn bản in bằng bút chì. Có bằng chứng cho thấy rằng trước Nội chiến, đã tồn tại một phương pháp bí mật cung cấp các thông điệp bí mật cho nô lệ để hỗ trợ họ trốn thoát. Bằng cách sử dụng nhiều họa tiết khác nhau trên chăn, thường được treo trên bậu cửa sổ để phơi khô, những thông điệp này đã được truyền tải đến những người bị bắt làm nô lệ, hướng dẫn họ hướng tới tự do. Một ví dụ về kiểu mẫu như vậy là biểu tượng Bear Paw được sử dụng bởi nô lệ người Mỹ gốc Phi để trốn sang các bang tự do và Canada trong thời gian từ đầu đến giữa thế kỷ 19 [6].

Thời thế chiến thứ II

Từ những năm 1930, kỹ thuật giấu tin đã được sử dụng trong Thế chiến thứ hai. Trên thực tế, cuộc chiến này thậm chí còn được gọi là Cuộc thi Steganography.

Về cách áp dụng kỹ thuật này trong cuộc xung đột toàn cầu này, để trao đổi thông tin và tránh bị quân Đức bắt và phát hiện, Vương quốc Anh đã phát triển một phương pháp giấu tin nhắn trong một bức thư mà không gây ra bất kỳ nghi ngờ nào.

Một phát minh khác của thời kỳ này, được người Đức phát triển từ những năm 1920 đến 1930 là công nghệ microdot, một phương pháp thu nhỏ tài liệu xuống kích thước của một dấu chấm thông thường, cho phép họ in hình ảnh rõ ràng, chất lượng cao. Bản chất nhỏ gọn của các chấm nhỏ làm nổi bật các thông điệp ẩn, đặc biệt khi các chấm này được gắn vào một phần khó thấy của văn bản hoặc hình ảnh[7].

Thời hiện đại

Ngày nay, kỹ thuật steganography được nghiên cứu vì cả lý do hợp pháp và bất hợp pháp. Trong số các giải pháp đầu tiên là viễn thông thời chiến, sử dụng máy phát vô tuyến trải phổ hoặc sóng sao băng để che giấu cả thông điệp và nguồn của nó.

Trong thị trường công nghiệp, với sự ra đời của phương tiện truyền thông kỹ thuật số và công nghệ lưu trữ, một trong những vấn đề quan trọng nhất là thực thi bản quyền, vì vậy các kỹ thuật đóng dấu kỹ thuật số đang được phát triển để hạn chế việc sử dụng dữ liệu có bản quyền.

Một cách sử dụng quan trọng khác là nhúng dữ liệu bệnh nhân vào hình ảnh y tế để tránh mọi vấn đề về hồ sơ và hình ảnh trùng khớp.

Trong số các hoạt động bất hợp pháp có việc che giấu dữ liệu được mã hóa mạnh để phục vụ các mục đích trái pháp luật hoặc thực hiện các cuộc tấn công chống lại các tổ chức trên toàn thế giới.

Tính cấp thiết

Những năm gần đây, việc nghiên cứu về giấu tin càng ngày càng quan trọng, nó phục vụ cho nhu cầu của sự phát triển mạnh mẽ về khoa học. Đặc biệt là khi số lượng truy cập mạng Internet bùng nổ, dẫn đến việc các loại hình tấn công khác nhau cực kì tinh vi ra đời, trong đó có giấu mã đọc vào bên trong các tệp, hình ảnh hoặc âm thanh dẫn đến những tổn hại vô cùng nghiêm trọng.

Không chỉ vậy đối với ứng dụng trong quân sự, nhằm bảo vệ các thông tin bí mật được gửi qua lại giữa các quân khu hay thậm chí là giữa các quốc gia với nhau cũng đòi hỏi rất lớn về kỹ thuật giấu tin để phục vụ quốc gia.

Ngoài ra, yêu cầu về xác thực và bảo vệ bản quyền số ứng dụng trong môi trường an ninh quốc gia cũng đòi hỏi cấp thiết nghiên cứu về kỹ thuật giấu tin.

Chính vì vậy bài viết này không chỉ đi sâu về giấu tin mà còn nói lên độ an toàn của nó cũng như ứng dụng thực tiễn.

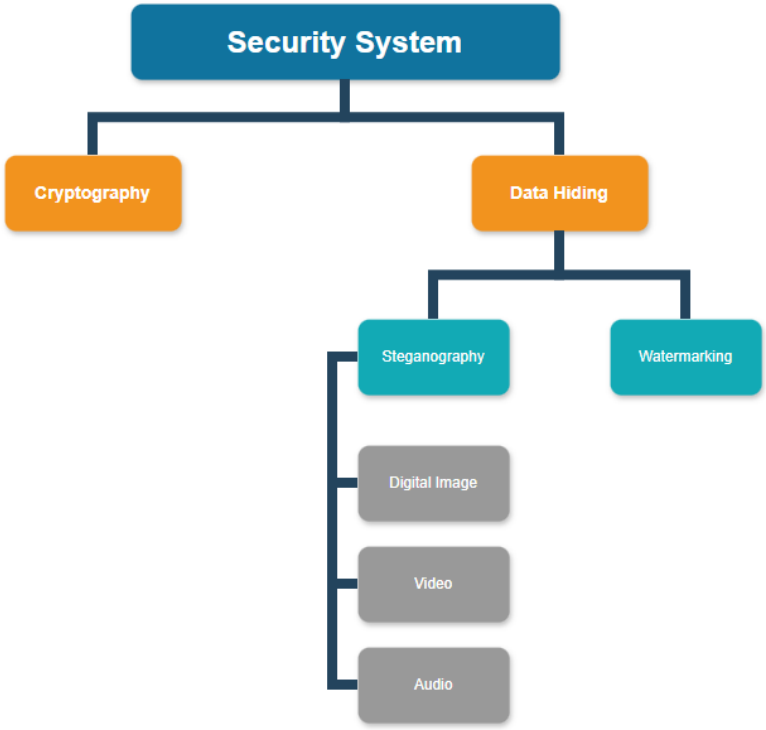
CHƯƠNG 1. TỔNG QUAN VỀ GIẤU TIN (STEGANOGRAPHY)

1.1 KHÁI NIỆM ĐẦU TIÊN

Trong khoa học máy tính, ẩn thông tin là nguyên tắc tách biệt các quyết định thiết kế dễ thay đổi nhất trong một chương trình máy tính, từ đó bảo vệ các phần khác của chương trình khỏi bị sửa đổi sâu rộng nếu các quyết định thiết kế đó bị thay đổi. Việc bảo vệ này bao gồm việc cung cấp một giao diện ổn định để bảo vệ phần còn lại của chương trình khỏi quá trình triển khai (các chi tiết có thể thay đổi). Nói cách khác, ẩn thông tin là khả năng ngăn chặn khách hàng truy cập vào các khía cạnh nhất định của một lớp hoặc thành phần phần mềm, bằng cách sử dụng các tính năng ngôn ngữ lập trình (như biến riêng tư) hoặc chính sách xuất rõ ràng.

Các kỹ thuật ẩn thông tin ban đầu tập trung vào việc nhúng dữ liệu vào hình ảnh mà không làm suy giảm đáng kể. Phương pháp ẩn thông tin nhanh được đề xuất bởi Hartung và đồng nghiệp [8].

Ẩn thông tin là một thuật ngữ rộng và chúng tôi dựa vào hệ thống bảo mật để phân loại thông tin như minh họa bên dưới:



Hình 1. Phân loại Security Systems

1.1.1 Digital Watermarking (Thủy vân số)

Hình mờ kỹ thuật số là một vấn đề quan trọng mà mọi người thường nghĩ việc che giấu thông tin là che giấu sự tồn tại của dữ liệu, nhưng đó chỉ là một phần trong đó. Một nhánh cơ bản không kém phần quan trọng là hình mờ kỹ thuật số.

Hình mờ kỹ thuật số là quá trình nhúng thông tin hoặc một loại điểm đánh dấu một cách tinh tế (chẳng hạn như hình ảnh, chuỗi số, v.v.) vào phương tiện kỹ thuật số như âm thanh, video hoặc hình ảnh. Điều này cho phép xác định quyền sở hữu bản quyền hoặc nguồn của nội dung.

Hình mờ kỹ thuật số, liên quan chặt chẽ đến kỹ thuật giấu tin, liên quan đến việc nhúng chữ ký vào phương tiện kỹ thuật số để khẳng định quyền sở hữu hoặc tính xác thực. Artz cung cấp cái nhìn tổng quan về kỹ thuật tạo hình chìm mờ kỹ thuật số và các ứng dụng của chúng[9].

Kỹ thuật này được sử dụng để theo dõi các hoạt động vi phạm bản quyền trên mạng xã hội và xác minh tính xác thực của các ghi chú trong hệ thống ngân hàng.

1.1.2 Steganography (Giấu tin)

Steganography là một trong những phương pháp được sử dụng để trao đổi thông tin ẩn và có thể được định nghĩa là nghiên cứu về giao tiếp vô hình, thường đề cập đến các cách che giấu sự tồn tại của thông điệp được truyền tải.

Để đạt được mục tiêu này, nội dung của ảnh bìa (mang dữ liệu ẩn) phải được sửa đổi.

Mặc dù cả mật mã và steganography đều có chung mục tiêu là liên lạc an toàn nhưng cách tiếp cận của chúng lại khác nhau. Ngược lại với mật mã, biến đổi dữ liệu gốc thành văn bản được mã hóa, kỹ thuật steganography bảo tồn dữ liệu gốc bằng cách ẩn nó trong các phương tiện khác. Hạn chế của mật mã là sự tồn tại của dữ liệu gốc, dù được mã hóa hay không. Do đó, steganography cung cấp thêm một lớp bảo mật cho tin nhắn khi truyền dữ liệu. Sự mạnh mẽ của các kỹ thuật ẩn thông tin và mật mã được nhìn nhận khác nhau. Khi kẻ tấn công có thể truy cập dữ liệu thực tế, hệ thống mật mã không còn an toàn nữa. Ngược lại, nếu kẻ tấn công giành được quyền đối với dữ liệu bí mật trong hệ thống steganographic thì hệ thống được coi là bị xâm phạm [10].

Đặc điểm	**Steganography**	**Watermarking**	**Cryptography**
Mục đích	Bảo vệ dữ liệu bí mật khỏi bị phát hiện	Bảo vệ tính hợp pháp của phương tiện	Mã hóa nội dung của dữ liệu
Thách thức	Khó nhận biết, bảo mật và dung lượng	Độ bền	Độ bền
Khóa	Không nhất thiết	Không nhất thiết	Cần thiết

Bảng 1. So sánh Steganography, Watermarking và Cryptography

1.2 STEGANOGRAPHY HÌNH ẢNH

Như đã nêu trước đó, hình ảnh là đối tượng che phủ phổ biến nhất được sử dụng cho kỹ thuật giấu tin. Trong lĩnh vực hình ảnh kỹ thuật số, tồn tại nhiều định dạng tệp hình ảnh khác nhau, hầu hết chúng dành cho các ứng dụng cụ thể. Đối với các định dạng tệp hình ảnh khác nhau này, tồn tại các thuật toán steganographic khác nhau.

1.2.2 Khái niệm

Giấu thông tin trong ảnh được hiểu là kỹ thuật nhúng đối tượng dữ liệu vào ảnh để nhận file ảnh chứa dữ liệu ẩn, đảm bảo chỉ có người nhận và người gửi mới biết sự tồn tại của nó.

1.2.3 Thuộc tính của Steganography hình ảnh

Các phương pháp Steganography hình ảnh phải đảm bảo đáp ứng đủ 3 thuộc tính dưới đây:

- **Bảo mật**: phương pháp steganography an toàn khi dữ liệu mà nó ẩn không thể bị phát hiện bằng kỹ thuật thống kê bởi bất kỳ bên thứ ba nào. Bảo mật là yêu cầu chính để ngăn chặn sự truy cập của các cá nhân hoặc máy tính bất hợp pháp trong khi liên lạc bằng kênh không bảo mật, do đó đảm bảo dữ liệu được an toàn.
- **Không thể nhận ra**: Tính không thể nhận thấy là ưu tiên hàng đầu trong kỹ thuật giấu tin và nhằm mục đích che giấu dữ liệu bí mật bằng các phương tiện khác. Mắt người không thể hiểu được nó ngay cả khi so sánh với hình ảnh gốc
- **Mạnh mẽ**: Tính mạnh mẽ biểu thị khả năng của phương pháp nhúng và trích xuất có thể chống lại mọi hành vi sai trái của thực thể thứ ba thông qua bất kỳ phương pháp xử lý nào. Chẳng hạn như, các cuộc tấn công như nén, chuyển đổi định dạng tệp và chuyển đổi giữa định dạng kỹ thuật số và tương tự có thể xảy ra trong quá trình giao tiếp.

1.3 PHÂN LOẠI ĐỊNH DẠNG ẢNH

1.3.1 Định dạng Bitmap (BMP)

- Đây là một định dạng phổ biến, không nén và được phát triển bởi Microsoft và IBM[11].
- Bitmap là thuật ngữ chung cho một hình ảnh được biểu diễn dưới dạng ma trận pixel, với mỗi pixel có một giá trị màu cụ thể. Nó không giới hạn ở một định dạng hình ảnh cụ thể mà có thể tham khảo nhiều định dạng khác nhau.
- Số bit trên mỗi pixel càng cao thì chất lượng hình ảnh và độ sắc nét càng cao.

Hình 2. Định dạng Bitmap

1.3.2 Định dạng GIF (Graphics Interchange Format)

- Định dạng này được CompuServe giới thiệu vào năm 1987 và từ đó được sử dụng rộng rãi trên World Wide Web [12].
- GIF sử dụng bảng màu hỗ trợ tối đa 256 màu (độ sâu màu 8 bit) nên thường được sử dụng trong biểu đồ, biểu tượng và ảnh động nhưng không dùng cho hình ảnh có nhiều màu vì giá trị màu bị suy giảm đáng kể.

1.3.3 Định dạng PNG (Portable Network Graphics)

- Là định dạng ảnh sử dụng phương pháp nén dữ liệu mới, giữ nguyên dữ liệu gốc.
- Định dạng này đã được World Wide Web Consortium (W3C) chuẩn hóa và được thiết kế để thay thế định dạng GIF độc quyền hiện có.
- Nó có chung đặc điểm với GIF nhưng hỗ trợ nhiều màu khác nhau dựa trên bảng màu 24-32 bit. Tuy nhiên, nó không thể tạo hình ảnh động [13].

1.3.4 Định dạng JPEG (Joint Photographic Experts Group)

- Đây là định dạng hình ảnh phổ biến nhất và được giới thiệu vào năm 1992.
- Là định dạng ảnh sử dụng phương pháp nén dữ liệu lossy, thường được sử dụng trong nhiếp ảnh kỹ thuật số. Mắt người có thể chấp nhận mức độ mất dữ liệu này và không làm giảm đáng kể chất lượng hình ảnh.
- Hơn nữa, đây là định dạng được sử dụng rộng rãi nhất trên web do tính chất nhẹ của nó.
- Hạn chế duy nhất của ảnh JPEG là việc nén dẫn đến mất dữ liệu và gần như không thể phục hồi được.

Hình 3. Ảnh về một con mèo hoang Châu Âu với tốc độ nén và tổn thất liên quan giảm dần từ trái sang phải

1.4 ỨNG DỤNG CỦA STEGANOGRAPHY HÌNH ẢNH

- **Ứng dụng quân sự**: Quân nhân có thể sử dụng kỹ thuật giấu tin để trao đổi thông tin nhạy cảm hoặc mệnh lệnh một cách an toàn trong môi trường thù địch. Điều này có thể rất quan trọng đối với các hoạt động bí mật hoặc duy trì liên lạc trong các cuộc xung đột.[22]

- **Xác thực và Xác minh**: Kỹ thuật Steganographic có thể được sử dụng để nhúng mã định danh hoặc mã xác thực duy nhất vào trong sản phẩm hoặc tài liệu nhằm chống hàng giả và đảm bảo tính xác thực. Điều này đặc biệt hữu ích trong các ngành công nghiệp như dược phẩm và hàng xa xỉ.[23]

- **Bảo mật dữ liệu**: Dữ liệu nhạy cảm, chẳng hạn như hồ sơ y tế[24] hoặc thông tin tài chính, có thể được bảo vệ bằng cách nhúng dữ liệu đó vào các vật chứa vô hại như hình ảnh hoặc tệp âm thanh. Điều này bổ sung thêm một lớp bảo mật cho việc truyền và lưu trữ dữ liệu.

- **Điều tra pháp y**: Các cơ quan thực thi pháp luật có thể sử dụng kỹ thuật giấu tin để phát hiện các thông điệp hoặc bằng chứng ẩn trong các tệp kỹ thuật số bị thu giữ từ tội phạm hoặc nghi phạm. Điều này có thể rất quan trọng trong các cuộc điều tra liên quan đến tội phạm mạng, gián điệp hoặc các hoạt động bất hợp pháp khác.[25]

- **Giao tiếp bí mật**: Trong các tình huống mà giao tiếp trực tiếp có nhiều rủi ro hoặc không thể thực hiện được, kỹ thuật giấu tin có thể được sử dụng để liên lạc bí mật. Gián điệp hoặc nhà hoạt động có thể nhúng thông tin nhạy cảm vào các tệp hình ảnh, âm thanh hoặc video để trao đổi tin nhắn mà không bị phát hiện.

1.5 NGUYÊN TẮC HOẠT ĐỘNG

Phần này cung cấp cái nhìn tổng quan về các kỹ thuật nhúng trong kỹ thuật giấu tin. Khái niệm cơ bản của bất kỳ phương pháp hoặc kỹ thuật nào trong kỹ thuật giấu tin kỹ thuật số là giấu dữ liệu riêng tư hoặc bí mật một cách kín đáo trong một phương tiện (cụ thể là hình ảnh kỹ thuật số) để truyền dữ liệu đó đến người nhận một cách an toàn.

1.5.1 Nguyên tắc nhúng dữ liệu

Có nhiều cách tiếp cận khác nhau để phân loại các kỹ thuật steganographic. Những cách tiếp cận này có thể được phân loại dựa trên phương tiện che phủ (các định dạng hình ảnh phổ biến như JPEG, GIF, PNG, v.v.) được sử dụng để liên lạc bí mật. Một cách tiếp cận khác phân loại các kỹ thuật dựa trên cách chúng sửa đổi ảnh bìa trong quá trình nhúng dữ liệu.

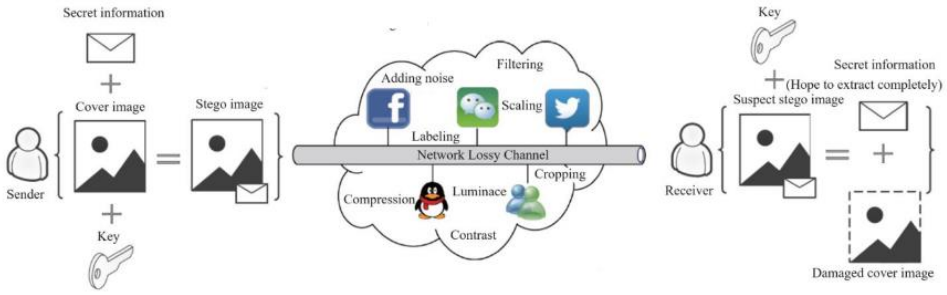
Tuy nhiên, quy trình nhúng dữ liệu phổ biến nhất được mô tả như sau:

- C: Đại diện cho ảnh bìa (nơi dữ liệu sẽ được nhúng).
- C': Biểu thị ảnh stego (ảnh bìa có thông tin ẩn).
- K: Một khóa tùy chọn (được sử dụng để mã hóa tin nhắn hoặc chuyển tin nhắn thành thứ gì khác).
- M: Thông điệp cần truyền đi.
- Nhúng (Em): Chúng ta sử dụng ảnh bìa C và phím tùy chọn K để nhúng tin nhắn M. Kết quả là hình ảnh stego C'.
- Trích xuất (Ex): Trong quá trình trích xuất, chúng ta sử dụng ảnh stego C' và cùng khóa K để lấy lại tin nhắn ẩn M.

$$Em = C \oplus \mathrm{K} \oplus \mathrm{M} \rightarrow \mathrm{C}' \quad (1)$$

$$v\grave{a}\ Ex\big(Em(c,k,m)\big) \approx m\ \forall c \in C, k \in K, m \in M \quad (2)$$

Điều này có thể được hiểu rõ hơn qua sơ đồ sau:



Hình 4.Quá trình nhúng dữ liệu và truyền đến người nhận [14]

1.5.2 Phân loại các kỹ thuật giấu tin

Qua nhiều năm, nhiều kỹ thuật khác nhau đã được nghiên cứu. Để phân biệt rộng rãi giữa các kỹ thuật steganographic khác nhau, người ta phải xem xét cả phương pháp sửa đổi hình ảnh và phương pháp sửa đổi định dạng tệp.

Kỹ thuật được chia thành hai phần chính:

- **Chèn (Insertion-based)**: Đúng như tên gọi, nó chèn thông báo cần ẩn vào tệp để tăng kích thước tệp mà không ảnh hưởng đến việc hiển thị ảnh bìa.
- **Thay thế (Substitution-based)**: Nó liên quan đến việc truy cập dữ liệu có sẵn, thông tin của ảnh bìa và thay thế cẩn thận nó bằng phân đoạn dữ liệu của thông báo ẩn để tránh hiện tượng giả hình ảnh.

Ngoài ra, có một phương pháp phân loại gần đây được gọi là **Thế hệ (Generation - Based)**.

Tuy nhiên, những kỹ thuật này đều có điểm mạnh và điểm yếu riêng. Mặc dù một kỹ thuật có thể che giấu nhiều thông tin hơn nhưng nó có thể có độ tin cậy kém.

Các kỹ thuật sẽ được thảo luận trong các phần sau của bài viết bao gồm:

- Spatial domain
- Transform domain
- Frequency domain
- Spread spectrum
- Exploiting the image format (EOF)

CHƯƠNG 2. CÁC KỸ THUẬT IMAGE STEGANOGRAPHY

2.1 SPATIAL DOMAIN (MIỀN KHÔNG GIAN)

2.1.1 Spatial Domain là gì ?

Kỹ thuật Spatial Domain liên quan đến việc nhúng dữ liệu để truyền bằng cách thay thế các bit được chọn cẩn thận từ các pixel ảnh bìa bằng các bit thông điệp bí mật.

Quá trình lựa chọn và thay thế này được xem xét cẩn thận và hầu như không có sự thay đổi rõ rệt nào về hiển thị so với ảnh gốc.

Mặc dù ưu điểm lớn nhất của kỹ thuật Miền không gian là tải trọng cao (khả năng ẩn nhiều dữ liệu) mà không ảnh hưởng đến chất lượng hình ảnh, nhưng chúng cũng có những hạn chế nghiêm trọng như tính dễ vỡ (dễ bị cắt xén, nén hình ảnh) và dễ bị phát hiện bởi các cuộc tấn công.

Các phương pháp thường được sử dụng bao gồm Least Significant Bit (LSB), Pixel value differencing (PVD), v.v.

2.1.2 Least Significant Bit (LSB)

Kỹ thuật dựa trên LSB là thuật toán giấu tin được biết đến rộng rãi nhất. Chúng hoạt động bằng cách thay thế các bit pixel hình ảnh ít quan trọng nhất.

Ví dụ: trong ảnh màu 24 bit sử dụng không gian màu RGB, mỗi pixel bao gồm ba màu: đỏ, lục và lam. Do đó, một lưới pixel có thể được xem là 3 pixel của ảnh màu 24 bit, chiếm 9 byte bộ nhớ:

(39 233 200) → (00100111 11101001 11001000)

(39 232 233) → (00100111 11001000 11101001)

(200 39 233) → (11001000 00100111 11101001)

Sử dụng ký tự 'A' có giá trị nhị phân là "10000001" để thay thế, chúng ta nhận được kết quả sau:

(00100111 11101000 11001000) → (39 232 200)

(00100110 11001000 11101000) → (38 200 232)

(11001000 00100111 11101001) → (20 039 232)

Trong ví dụ này, chúng tôi chọn bit ít quan trọng nhất làm bit để thay thế và chúng tôi đã sử dụng 3 bit cho mỗi pixel để ẩn ký tự 'A' trong ảnh. Sự thay đổi về giá trị RGB của từng màu là không đáng kể đối với mắt người, do đó ẩn dữ liệu một cách hiệu quả so với ảnh gốc.

2.1.3 Đánh giá và kết luận

Có thể thấy, việc tiếp tục nhấn mạnh và phát triển các kỹ thuật Spatial Domain nói chung và kỹ thuật LSB nói riêng chính là việc tối đa hóa sức mạnh tải trọng của chúng. Không chỉ vậy, nó còn cần phải giải quyết được nhược điểm lớn nhất của kỹ thuật này, đó là những lo ngại về bảo mật và độ bền.

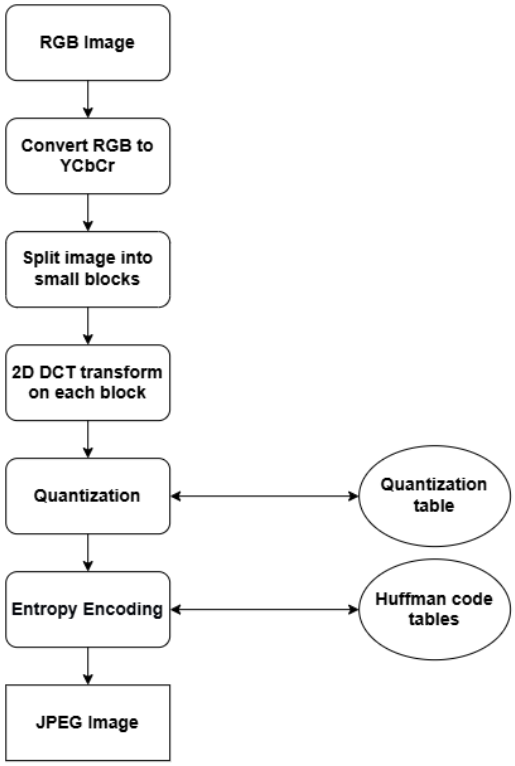
2.2 TRANSFORM DOMAIN (MIỀN CHUYỂN ĐỔI)

Như chúng ta đã biết, hình ảnh JPEG là một trong những định dạng hình ảnh phổ biến nhất trên Internet do kích thước tệp nhỏ của chúng. Tuy nhiên, để đạt được điều này, ảnh JPEG phải trải qua nhiều quá trình nén ảnh, điều này có thể dẫn đến việc sửa đổi pixel để giảm kích thước của chúng. Do đó, kỹ thuật giấu tin truyền thống sử dụng hình ảnh JPEG có thể dẫn đến hỏng dữ liệu trong quá trình nén.

Để giải quyết vấn đề này, nảy sinh ý tưởng tìm và sử dụng các pixel ít bị ảnh hưởng nhất trong quá trình cắt xén, nén và các kỹ thuật xử lý hình ảnh khác.

2.2.1 Quy trình nén ảnh JPEG

Mắt người khá thành thạo trong việc phát hiện những khác biệt nhỏ về độ sáng trên các khu vực tương đối lớn, nhưng không đủ chính xác để phân biệt các mức cường độ khác nhau ở tần số không gian cao [15]. Điều này có nghĩa là cường độ tần số cao hơn có thể giảm mà không làm thay đổi đáng kể hình thức trực quan của hình ảnh. Do đó, sự phát triển của nén ảnh JPEG có các bước như sau :



Hình 5. Quá trình nén ảnh JPEG

Bước 1 : Chuyển Đổi Không Gian Màu

JPEG thường làm việc trên không gian màu YCbCr thay vì RGB:

- Y (luminance): Thành phần độ sáng.
- Cb và Cr (chrominance): Thành phần màu xanh và đỏ.

Ảnh RGB được chuyển đổi sang không gian YCbCr vì mắt người nhạy cảm hơn với sự thay đổi độ sáng so với màu sắc.

Bước 2 : Chia Ảnh Thành Các Khối

Ảnh YCbCr được chia thành các khối nhỏ kích thước 8×8 pixel. Điều này giúp giảm độ phức tạp của tính toán và tận dụng sự tương quan cục bộ trong ảnh.

Bước 3: Áp dụng phép biến đổi Cosine rời rạc (DCT)

- DCT là một phép toán giống với "biến đổi Fourier rời rạc" nó sẽ được áp dụng và hoạt động độc lập trên mỗi khối 8x8.
- Nó chuyển đổi dữ liệu miền không gian thành dữ liệu miền tần số.
- DCT được biết đến với đặc tính "nén năng lượng", nghĩa là nó thể hiện tín hiệu một cách hiệu quả bằng cách sử dụng một số thành phần tần số thấp.
- Các hệ số DCT được tính toán cho mỗi khối, thể hiện sự đóng góp của sóng cosin vào ảnh.

Công thức DCT hai chiều cho khối 8x8 như sau:

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\ cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

Trong đó, C(u) và C(v) là các hệ số chuẩn hóa, f(x,y) là các giá trị pixel tại tọa độ (x,y), và F(u,v) là hệ số DCT tại tọa độ (u,v). Với C(x) = 1/√2 khi x = 0 và C(x) = 1 trong trường hợp còn lại.

Ngoài ra còn có thể sử dụng biến đổi Fourier rời rạc (DFT) hoặc biến đổi Wavelet rời rạc (DWT)

Bước 4 : Lượng tử hóa

Các hệ số DCT sau đó được lượng tử hóa bằng cách chia từng hệ số cho một giá trị trong ma trận lượng tử hóa và làm tròn kết quả.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Bảng 2. Ma trận lượng tử hóa

Bước 5 : Sắp Xếp Theo Thứ Tự Zigzag

Các hệ số DCT đã lượng tử hóa được sắp xếp theo thứ tự zigzag để nhóm các hệ số không (zero) lại với nhau, giúp tăng hiệu quả của bước mã hóa tiếp theo.

Bước 6 : Mã Hóa Entropy

Các hệ số được mã hóa sử dụng mã Huffman hoặc mã RLE (Run-Length Encoding) để giảm thêm dung lượng:

Mã Huffman: Sử dụng mã biến độ dài cho các giá trị phổ biến.

Mã RLE: Nén các chuỗi dài của các giá trị giống nhau.

Bước 7 : Tạo Tệp JPEG Cuối Cùng

Các khối đã mã hóa được ghép lại và lưu trữ trong một tệp JPEG với các thông tin tiêu đề cần thiết.

Ví dụ:

Ta có khối pixel ảnh 8x8 ban đầu khi chuyển đổi sang không gian màu YcbCr như sau:

52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	68	77	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

Bảng 3. Ảnh có không gian màu Ycbcr

Sau quá trình biến đổi Cosine rời rạc và lượng tử hóa đồng thời sắp xếp theo ziczac, ta sẽ được:

-26	-3	-3	0	0	0	0	0
-6	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

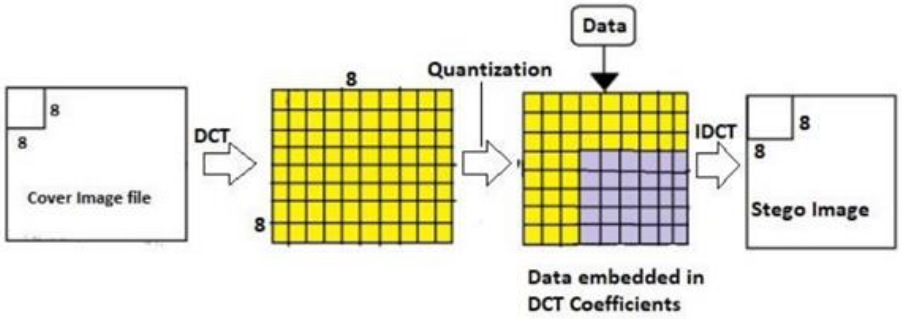
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Bảng 4. Ảnh sau khi biến đổi DCT

Từ ví dụ, chúng ta có thể thấy phần tử ở vị trí (0,0) là lớn nhất và cũng đại diện cho pixel có tần số cao nhất hoặc pixel sáng nhất. Ngược lại, các giá trị dần dần căn chỉnh với các pixel có tần số thấp hơn hoặc các pixel ít quan trọng hơn trong khối.

2.2.1 Giải quyết vấn đề

Thông qua quá trình nén ảnh JPEG, rõ ràng là các pixel hầu như không thay đổi hoặc thay đổi rất ít trong quá trình nén JPEG là những pixel có tần số thấp hoặc độ sáng thấp. Vấn đề có thể được giải quyết bằng cách nhúng dữ liệu vào các pixel này vì dữ liệu sẽ không bị mất hoặc sẽ bị thay đổi trong phạm vi chấp nhận được khi truyền qua các kỹ thuật xử lý ảnh.



Hình 6. Nhúng dữ liệu bằng DCT[16]

2.2.2 Đánh giá hiệu quả

Miền biến đổi là một kỹ thuật biến đổi các pixel từ miền thời gian sang miền tần số. Về mặt kỹ thuật số, một hình ảnh về cơ bản là một tập hợp các pixel. Các pixel biên được biết là có tần số cao, trong khi các pixel không biên được biết là các pixel tần số thấp. Có nhiều kỹ thuật khác nhau trong miền biến đổi như DCT (Biến đổi Cosine rời rạc), DWT (Biến đổi Wavelet rời rạc) và DFT (Biến đổi Fourier rời rạc).

Mặc dù dữ liệu bí mật có thể được ẩn giấu rất tốt bằng kỹ thuật này nhưng quá trình nén hình ảnh là một quá trình toán học khiến nó khá phức tạp. Ngoài ra, nó chỉ có thể được sử dụng với một số định dạng hạn chế và phải liên quan đến việc nén hình ảnh, chẳng hạn như định dạng JPEG. Mặc dù vậy, nó vẫn được đánh giá cao do tính bảo mật cao hơn đáng kể so với các kỹ thuật miền không gian.

2.3 PATCHWORK (CHẮP VÁ)

PatchWork là một trong những kỹ thuật mạnh mẽ và an toàn, được phát triển bởi Walter Bender, Daniel Gruhl và Norishige Morimoto và được xuất bản

trên Tạp chí Hệ thống IBM vào năm 1996. Nó dựa trên các mô hình thống kê và một tập hợp các số giả ngẫu nhiên. PatchWork nổi bật là một trong những kỹ thuật mạnh mẽ nhất xét về độ bền và tính bảo mật[17].

2.3.1 Kỹ thuật PatchWork

a) Cơ chế hoạt động

PatchWork, kỹ thuật ẩn dữ liệu, có thể hiểu một cách đơn giản như sau:

Chọn ngẫu nhiên một cặp pixel A và B từ ảnh. Gọi độ sáng tại A là 'a' và tại B là 'b'.

Điều chỉnh độ sáng của một số điểm trong A đồng thời điều chỉnh độ sáng của các điểm tương ứng trong B. Cụ thể, ta sửa đổi độ sáng sao cho a′ = a + μ và b′ = b − μ sao cho với mỗi bit ta muốn nén:

- Nếu bit là ’ 1 ’, hãy đảm bảo rằng a ’ - b ’ > 0.
- Nếu bit là ’ 0 ’, hãy đảm bảo rằng a ’ - b ’ < 0.

Lặp lại quy trình này để phân phối dữ liệu ẩn trên nhiều vị trí trong ảnh.

Về quá trình trích xuất rất đơn giản: xác minh thứ tự của các cặp pixel được sử dụng để ẩn và kiểm tra lại các điều kiện được xác định trước.

b) Đánh giá vấn đề hạn chế

Nếu lặp lại với n pixel thì ta sẽ có tổng tương ứng với số lượng độ sáng đã thay đổi trong ảnh như sau :

$$S(n) = \sum_{i=1}^{n} 2 * (a_i - b_i)$$

Từ đó ta có thể thấy được rằng μ sẽ thay đổi theo khoảng cách độ sáng giữa 2 điểm A và B. Khoảng cách càng lớn thì μ càng lớn đồng nghĩa với việc sự thay đổi độ sáng càng lớn, điều đó dẫn đến việc hình ảnh có thể bị phá hủy.

Hiển nhiên điều kiện lý tưởng nhất chính là A và B có độ sáng bằng nhau hay a – b = 0.

2.3.1 Đánh giá hiệu quả

Tàng hình: Kỹ thuật này chắc chắn là một trong những phương pháp kín đáo nhất, vì trong trường hợp lý tưởng, mỗi pixel chỉ cần điều chỉnh độ sáng một cấp, khiến mắt người hầu như không thể nhận thấy được.

Bảo mật và mạnh mẽ: Việc sử dụng chuỗi số giả ngẫu nhiên đảm bảo dữ liệu được phân tán và ngẫu nhiên trong toàn bộ hình ảnh. Ngoài ra, nhiều pixel góp phần lưu trữ một bit dữ liệu, giúp kỹ thuật này có khả năng chống chịu cao với mọi sửa đổi hình ảnh và bảo vệ nó khỏi các cuộc tấn công.

Điểm yếu đáng kể chưa được giải quyết: Hạn chế chính, cũng hạn chế việc áp dụng và thảo luận rộng rãi về kỹ thuật PatchWork, là yêu cầu nghiêm ngặt để hình ảnh tuân thủ các điều kiện lý tưởng (trong đó hầu hết các pixel thể

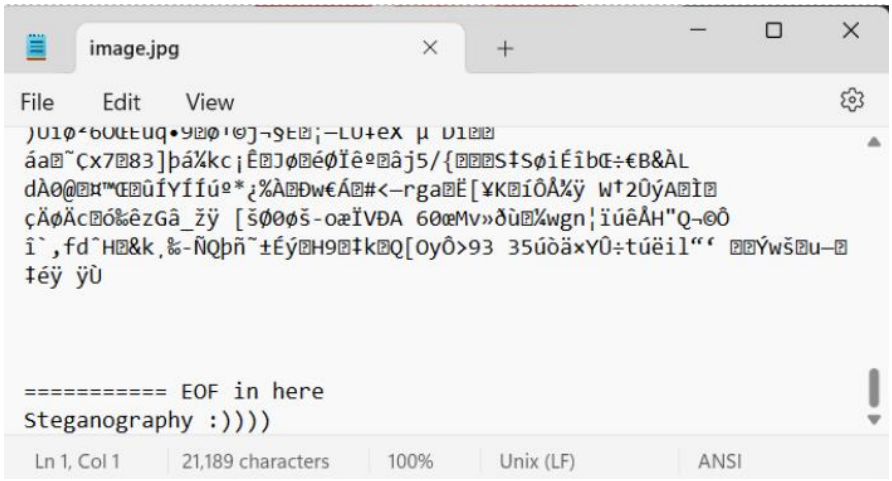
hiện độ sáng đồng đều) để có hiệu suất tối ưu. Hơn nữa, khả năng che giấu thông tin còn hạn chế cũng tạo nên một thiếu sót lớn khác.

2.4 KỸ THUẬT END OF FILE (EOF)

2.4.1 Hoạt động

Như chúng ta đã biết, tất cả các kỹ thuật steganography đều nhằm mục đích sửa đổi các thành phần hình ảnh để che giấu thông điệp bên trong chúng một cách khéo léo. Sự đánh đổi này liên quan đến việc hy sinh chất lượng hình ảnh để tăng cường khả năng tàng hình và bảo mật dữ liệu trong khi vẫn giữ được kích thước có thể quản lý được.

Tuy nhiên, phương pháp End-of-File (EOF) khai thác hành vi của các ứng dụng hiển thị hình ảnh, khiến chúng bỏ qua mọi dữ liệu ngoài thẻ EOF và chỉ đọc phần trước đó.



Hình 7. EOF trong file txt của hình ảnh

Điều này dẫn đến khả năng chèn hoàn toàn các tin nhắn ẩn đằng sau thẻ EOF bằng một lệnh đơn giản, như sau

```
Copy /b Cover.jpg+Message.txt Stego.jpg
```

Trong đó Cover.jpg là ảnh gốc còn Message.txt là file thông tin cần ẩn

2.4.2 Đánh giá hiệu quả

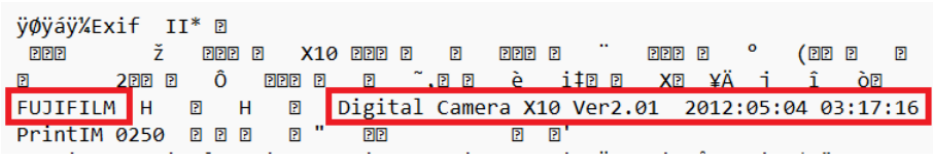
Bằng cách chèn thông báo phía sau thẻ EOF, kỹ thuật này không làm thay đổi bất kỳ thành phần nào trong hình ảnh. Do đó, chất lượng hình ảnh không thay đổi và có thể ẩn số lượng tin nhắn ẩn tùy ý. Sửa đổi duy nhất so với ảnh gốc là tăng kích thước tỷ lệ thuận với dữ liệu được nhúng.

Mặc dù cách tiếp cận này có vẻ đáng chú ý nhưng thật không may, nó không thể chống lại sự tấn công từ các chuyên gia hoặc bất kỳ hình thức xử lý hình ảnh nào (chẳng hạn như nén JPEG)

2.4.3 Thông tin tập tin mở rộng hình ảnh

Kỹ thuật tương tự như EOF liên quan đến việc ẩn thông tin trong thông tin tệp mở rộng (EXIF) của hình ảnh. Phần này được nhà sản xuất sử dụng để lưu trữ

thông tin chi tiết về hình ảnh, bao gồm độ phân giải, nhãn hiệu máy ảnh và thời gian chụp.



Hình 8. Cho biết ảnh được chụp bằng máy ảnh FUJIFILM và thời gian chụp.

Rõ ràng, kỹ thuật này cũng không đáng tin cậy, tương tự như EOF.

2.5 KỸ THUẬT SPREAD SPECTRUM

Ý tưởng của kỹ thuật này dựa trên kỹ thuật trải phổ trong video [17]. Về cơ bản, kỹ thuật này khai thác nhiễu kỹ thuật số tự nhiên trong hình ảnh (chẳng hạn như nhiễu trắng) để che giấu thông tin bên trong. Điều này làm cho các kỹ thuật quét hiện tại dễ dàng bỏ qua các điểm nhiễu chứa thông tin này.

2.5.1 Cách hoạt động

Công nghệ được sử dụng ở đây là Bộ tạo nhiễu giả ngẫu nhiên (PRNG), có độ tương quan yếu để tạo ra tín hiệu nhiễu giả tương tự như nhiễu tự nhiên.

Quá trình này được tóm tắt như sau:

a) Quá trình ẩn thông tin

Giá sử có thông tin cần gửi đi dưới dạng một chuỗi số bit riêng lẻ là :

$$a = \{a_1, a_2, \dots, a_k\} \quad \forall i \in \{1, \dots, k\} : a_i \in \{-1,1\}$$

Để có thể triển khai trải phổ trực tiếp, ta chia ảnh thành các trải phổ s_i trong đó với mỗi trải phổ sẽ có các tín hiệu nhiễu riêng s_{i_j} được tạo bởi bộ số giả ngẫu nhiên :

$$s_i \in s = \{s_1, \dots, s_M\}, \qquad M \geq k$$
$$\forall i \in \{1, \dots, M\}: s_i = \{s_{i_0}, \dots, s_{i_{n-1}}\}$$
$$\forall j \in \{0, \dots, n-1) : \ s_{i_j} \in \{-1,1\}$$

Khi đó hình ảnh gốc C sẽ được điều chế thành hình ảnh mới là V:

$$V = C + G * E$$

Trong đó G là một hệ số dương để mở rộng trải phổ, E là tín hiệu nhiễu được đưa vào :

$$E = \sum_{i=1}^{k} s_i * a_i$$

Hiểu đơn giản rằng, mỗi bit tin nhắn gửi đi sẽ được giấu tương ứng vào một trải phổ theo đúng thứ tự.

b) Quá trình khôi phục tin nhắn

Các tín hiệu nhiễu trong trải phổ được sử dụng là tương quan yếu với nhau, hay hệ số tương quan giữa chúng bằng 0 :

$$\forall i \neq j: \rho(s_i, s_j) \approx 0$$

Và bởi các tín hiệu nhiễu được tạo ra ngẫu nhiên nên chúng cũng không tương quan với ảnh gốc :

$$\forall i : \rho(s_i, C) \approx 0$$

Khi đó để có thể giải mã, ta sẽ dựa vào tính hệ số tương quan giữa từng trải phổ với ảnh sau khi được điều chế:

$$\rho(V, s_i) = \rho(C + G * E, s_i)$$

$$= \rho(C, s_i) + G * \rho(E, s_i) \approx G * \sum_{j=1}^{k} a_j * \sum_{k=0}^{n} \frac{s_{i_k} * s_{j_k}}{n}$$

Bởi trước đó ta đã có $\forall i \neq j: \rho(s_i, s_j) \approx 0$, lúc này phương trình trở thành :

$$\rho(V, s_i) \approx G * a_i * \frac{1}{n}$$

Có thể thấy hệ số tương quan lúc này sẽ phụ thuộc dấu vào giá trị của a_i, khi đó ta có thể tìm ra được a_i lúc đầu bằng phép toán :

$$a_i = sign(\rho(V, s_i)) = \begin{cases} +1, \rho(V, s_i) > 0 \\ -1, \rho(V, s_i) < 0 \end{cases}$$

2.5.2 Đánh giá

Kỹ thuật này hoàn toàn được kiểm nghiệm có thể chống lại hầu hết các cuộc tấn công và các kỹ thuật xử lý ảnh (như nén JPEG) nhưng nó lại khá khó tiếp cận bởi các lý luận toán học xung quanh.

2.6 ĐÁNH GIÁ KỸ THUẬT STEGANOGRAPHY HÌNH ẢNH

Tất cả các kỹ thuật nêu trên đều có điểm mạnh và điểm yếu khác nhau và tất cả các thuật toán steganographic đều phải tuân thủ một số yêu cầu cơ bản. Yêu cầu quan trọng nhất là thuật toán steganographic phải không thể nhận dạng được. Những yêu cầu này như sau:

	EOF	LSB	JPEG Compression	Patchwork	Spread Spectrum
Vô hình	Cao	Cao	Cao	Cao	Khá cao
Sức chứa khối hàng	Cao	Cao	Trung bình	Thấp	Trung bình
Chống lại các cuộc tấn công	Thấp	Thấp	Trung bình	Cao	Cao
Chống lại thao tác hình ảnh	Thấp	Thấp	Trung bình	Cao	Trung bình

Bảng 5. So sánh các kỹ thuật

2.7 BỘ SỐ GIẢ NGẪU NHIÊN(PRNG) – ĐỌC THÊM

Trình tạo số ngẫu nhiên giả (PRNG) đề cập đến một thuật toán sử dụng các công thức toán học để tạo ra chuỗi số gần giống với dãy số ngẫu nhiên.

Mặc dù trông nó giống với bộ số ngẫu nhiên nhưng với một seed sẽ được đưa ra từ trước, mỗi seed khác nhau sẽ đưa ra các chuỗi số khác nhau nhưng mỗi seed luôn sẽ random ra một dãy số ngẫu nhiên duy nhất không thay đổi về thứ tự.

Chính vì vậy chúng ta hoàn toàn có thể dựa vào đây để có thể truy cập vào các điểm giống nhau và đúng thứ tự để trích xuất dữ liệu miễn là có đúng seed.

CHƯƠNG 3. CÀI ĐẶT, ĐÁNH GIÁ VÀ NÂNG CAO ĐỘ HIỆU QUẢ

3.1 KỸ THUẬT LSB

3.1.1 Cài đặt

Các thư viện được sử dụng trong chương trình :

- System.Drawing : sử dụng để đưa ảnh về dạng BitMap

Chương trình sẽ gồm 4 phần chính :

1. Xử lý tin nhắn bí mật thành dữ liệu nhúng
2. Nhúng dữ liệu (quá trình giấu tin)
3. Giải mã dữ liệu
4. Xử lý dữ liệu thành tin nhắn bí mật

a) Xử lý tin nhắn cần nhúng

- Để có thể nhúng tin nhắn vào trong ảnh, ta phải xử lý tin nhắn bí mật đầu vào , cụ thể là đưa tin nhắn về dạng nhị phân (mỗi kí tự sẽ có độ dài cố định 8 bit)

```
binary.Append(Convert.ToString(b, 2).PadLeft(8, '0'));
```

- Tùy vào trường hợp nhúng bao nhiêu bit dữ liệu vào trong một pixel, ta sẽ tùy chỉnh sao cho độ dài của dữ liệu có thể phù hợp với số bit giấu vào mỗi pixel.
- Để phục vụ trong quá trình giải mã mà không cần phải truyền bí mật độ dài của tin nhắn qua con đường khác, ta sẽ trực tiếp nhúng độ dài tin nhắn vào trong 4 pixel đầu, tương đương với độ dài 12 bit.

```
Convert.ToString(mess.Length / 8, 2).PadLeft(12, '0');
```

- Trong trường hợp này quy định mỗi pixel giấu 3 bit hay mỗi màu trong hệ RGB sẽ giấu ở 1 bit ít quan trọng nhất. Vì vậy độ dài chuỗi sẽ phải chia hết cho 3. Nếu không chia hết sẽ thêm số 0 vào cuối tương ứng.

```
mess = mess.PadLeft((3 - (mess.Length % 3)) + mess.Length, '0');
```

b) Nhúng dữ liệu (Quá trình giấu tin)

- Trước khi nhúng, ta sẽ đưa ảnh về dạng BitMap.
- Bởi ta chỉ nhúng vào mỗi pixel 3 bit vì vậy sẽ tách lần lượt 3 phần tử trong chuỗi tin nhắn cần giấu để nhúng vào từng pixel
- Với mỗi pixel, ta sẽ gọi tới 3 màu RGB, sau đó dùng phép toán thao tác bit (phép AND) để chuyển bit cuối cùng của từng màu về 0 rồi cộng với bit tin nhắn cần giấu

```
Pixel.Red & 254 + Convert.ToInt32(mess[0], 2);
Pixel.Green & 254 + Convert.ToInt32(mess[0], 2);
Pixel.Blue & 254 + Convert.ToInt32(mess[0], 2);
```

- Từ pixel mới, ta đưa trở lại vào ma trận ảnh BitMap để tạo thành 1 ảnh mới.

c) Giải mã dữ liệu

- Từ ảnh đã được giấu tin, ta chuyển về dạng BitMap.
- Ta đã biết trước đó 4 pixel đầu tiên sẽ phụ trách lưu lại độ dài của tin nhắn đầu vào nên ta sẽ ưu tiên tách 4 pixel này trước.
- Sau khi có độ dài, từ pixel thứ năm ta sẽ lần lượt đọc 3 màu trong hệ RGB của từng pixel và sử dụng phép toán thao tác bit (phép AND) để lấy bit cuối cùng cho đến khi đủ độ dài trước đó.

```
Convert.ToString(Pixel.Red & 1)
Convert.ToString(Pixel.Green & 1)
Convert.ToString(Pixel.Blue & 1)
```

d) Xử lý dữ liệu đã giải mã thành tin nhắn bí mật

- Sau khi lấy được chuỗi dữ liệu đã giấu, ta sẽ bỏ đi phần kí tự thừa ở đầu đã thêm vào trước đó để dễ nhúng.
- Cuối cùng lấy lần lượt 8 bit trong chuỗi để đưa chuyển đổi trở lại thành tin nhắn cần giấu.

```
string binaryString = data.Substring(i * 8, 8);
  bytes[i] = Convert.ToByte(binaryString, 2);
```

3.1.2 Kết quả chạy

Chạy kiểm nghiệm với ảnh có kích thước 880 x 880 và độ dài tin nhắn cần giấu là 100 kí tự (tương đương với 800 bit cần giấu).

Hình 9. Ảnh gốc

Hình 10. Ảnh Stego

Kết quả có thể thấy, hoàn toàn không thể phân biệt 2 ảnh nếu nhìn bằng mắt thường dù ảnh bên phải đã nhúng đến 800 bit dữ liệu.

3.1.1 Đánh giá độ hiệu quả

Rõ ràng, bằng cách sử dụng 3 bit trên mỗi pixel để ẩn dữ liệu, về mặt lý thuyết, một hình ảnh có kích thước 800*600 có thể che giấu tổng cộng 1.440.000 bit (tương đương 180.000 byte hoặc 180.000 ký tự) - một con số rất lớn. Tuy nhiên, để ẩn được nhiều dữ liệu hơn, kỹ thuật này yêu cầu ảnh bìa phải lớn hơn.

Do đó, chúng tôi bắt đầu đánh giá một khía cạnh khác, đó là tăng số lượng bit ít quan trọng nhất được chọn trong pixel để thay thế dữ liệu.

Để đánh giá hiệu quả, chúng ta sẽ sử dụng thước đo **delta E** do International Commission on Illumination (CIE) tạo ra, có thể hiểu như sau:

- Delta E nhỏ hơn hoặc bằng 1.0: Gần giống với màu gốc.
- Delta E nhỏ hơn hoặc bằng 1 - 2: Gần giống với màu gốc cần kiểm tra kỹ mới có thể phát hiện.
- Delta E nhỏ hơn hoặc bằng 2 - 10: Khác biệt rõ rệt so với màu gốc.
- Delta E nhỏ hơn hoặc bằng 11 - 49: Gần như màu tương phản.
- Delta E nhỏ hơn hoặc bằng 50 - 100: Màu tương phản.

Sau khi trải qua nhiều lần thực nghiệm, ta có bảng đánh giá hiệu quả sử dụng delta E như sau:

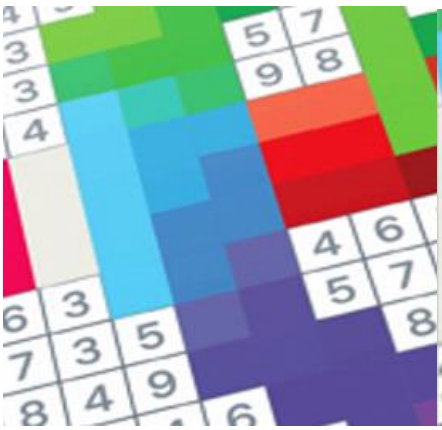
Số bit ít quan trọng được giấu trong mỗi pixel	**Lượng dữ liệu đã nhúng**	**Giá trị Delta E trung bình cho mỗi thay đổi màu**
1 bit	60000 ký tự	0.52912
2 bit	120000 ký tự	1.13968
3 bit	180000 ký tự	2.25129
4 bit	240000 ký tự	4.23711

Bảng 6. Bảng đánh giá hiệu quả sử dụng delta E

Ngoài ra ta còn đánh giá qua thực tế:

Hình 11. Ảnh gốc

Hình 12. 1 Bit được sử dụng



Hình 13. 2 Bit được sử dụng

Hình 14. 3 Bit được sử dụng

Hình 15. 4 Bit được sử dụng

3.2 KỸ THUẬT SPREAD SPECTRUM (TRẢI PHỔ)

3.2.1 Cài đặt

Thư viện được sử dụng trong chương trình bao gồm :

- OpenCV-Python: là một thư viện mà các nhà phát triển sử dụng để xử lý hình ảnh cho các ứng dụng thị giác máy tính
- Random : Phục vụ cho việc tạo dãy số PRNG

Chương trình gồm 2 phần chính :

- Nhúng dữ liệu vào từng trải phổ
- Giải mã dữ liệu từ ảnh stego

Lưu ý :

- Ảnh sẽ cần được chuyển về hệ màu **Ycbcr**, để tính toán theo độ sáng của pixel.
- Quy định độ rộng của một trải phổ (signa) = chiều cao ảnh / độ dài tin nhắn
- Tạo một ma trận bằng kích thước của ảnh để chứa các dãy PRNG dành riêng cho từng trải phổ.

```
        for i in range(height):
```

```
            random.seed( int(i / height) )
            for k in range(width):
                PRNG[k] = random.randint(0, 1)
                if (array[k] == 0): array[k] = -1
            array_PRNG[i] = PRNG
```

- Một dãy số PRNG chỉ chứa các phần tử {1,-1} có độ dài bằng chiều rộng của ảnh và dành riêng cho mỗi trải phổ với "seed" chính là thứ tự của trải phổ.

a) Nhúng dữ liệu vào từng trải phổ

- Tin nhắn mật đầu vào đã ở sẵn dạng bit
- Ta sẽ chuyển tin nhắn thành dạng {-1,1} để phục vụ quá trình nhúng

```
array_mess = [-1 if message[i] == '0' else 1 for i in range(message_length)]
```

- Quy định rằng mỗi trải phổ sẽ chỉ giấu 1 bit tin nhắn. Khi đó ta có mỗi pixel sẽ được tính lại như sau:

```
   pixel = pixel + G * array_PRNG[i, j] * array_mess[int(i / signa)]
```

- Sau khi thay đổi toàn bộ pixel, đưa nó trở thành ảnh mới và lưu lại *độ dài tin nhắn*.

b) Giải mã dữ liệu trong các trải phổ

- Từ *độ dài tin nhắn* đã lưu trước đó, ta tính ra độ rộng của từng trải phổ.
- Tương ứng với ma trận PRNG, ta tính hệ số tương quan của từng trải phổ trong tương ứng trong ma trận PRNG với từng phần trải phổ trong ảnh.

```
for i in range(message_length):
    row = image_ycbcr[signa * i : signa * (i + 1), :, 0]. flatten()
    array = array_PRNG[signa * j : signa * (j + 1)]. flatten()
    correlation_coefficient = np.corrcoef(array, row)[0, 1]
    ans += correlation_coefficient
```

- Cuối cùng sử dụng phép tính $sign(ans)$ để có thể lấy được bit tin nhắn ẩn giấu.

3.2.1 Kết quả chạy

Chạy kiểm nghiệm với ảnh có kích thước 880 x 880 và độ dài tin nhắn cần giấu là 100 kí tự (tương đương với 800 bit cần giấu).

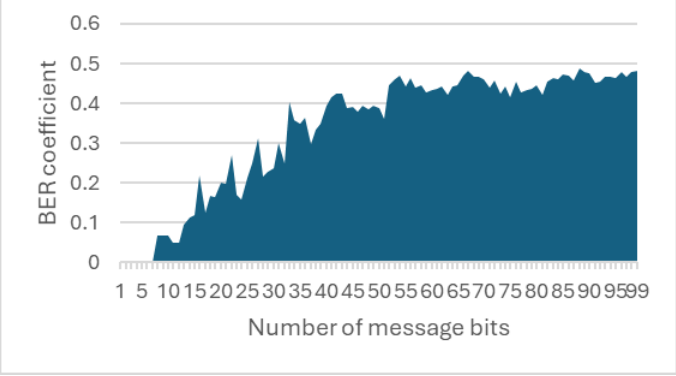
Hình 16. Bên trái là ảnh gốc, bên phải là ảnh stego

Tuy có thể nhìn thấy những đường sọc kẻ nhưng nó lại dễ đánh lừa thị giác máy tính khi máy tính có thể coi đó là nhiễu trắng, nhiễu sáng trong ảnh mà bỏ qua.

3.2.2 Đánh giá độ hiệu quả

Trong nghiên cứu, do mối tương quan yếu giữa phổ trải rộng xấp xỉ 0 và các lỗi tiềm ẩn khi ước tính giá trị của a_i nên tỷ lệ lỗi bit (BER) khá cao trong thực tế. Sau khi lặp lại thử nghiệm 100 lần với kích thước ảnh 480x360, việc đánh giá sự phụ thuộc vào giá trị G và số bit thông báo trong quá trình khôi phục như sau:

Hình 17. BER theo giá trị G, số bit thông báo được kiểm tra là 4



Hình 18. BER theo giá trị k thì giá trị của G được sử dụng là 4

3.3 PHÁT HIỆN DỮ LIỆU ẨN TRONG HÌNH ẢNH (CHI – SQUARE)

Steganalysis là nghệ thuật phát hiện các thông điệp ẩn trong ảnh stego. Thông qua phương pháp chủ động hoặc thụ động, kẻ tấn công sẽ phân tích và xác định xem hình ảnh có chứa thông điệp ẩn hay không. Dữ liệu được nhúng thường không làm thay đổi hình thức trực quan của ảnh bìa nhưng làm gián đoạn các thuộc tính thống kê nội bộ của nó.

Thông thường, kẻ tấn công không biết ảnh bìa gốc và chỉ có ảnh stego để đánh giá. Điều này được gọi là steganalysis mù hoặc steganalysis thống kê phổ quát. Phần này đề cập cụ thể đến một phương pháp thường được sử dụng: Kiểm tra Chi-Square để tấn công các hình ảnh sử dụng kỹ thuật LSB [19,20].

3.3.1 Ý tưởng thực hiện

Phương pháp này được đưa ra bởi Westfeld và Pfitzma [20], ý tưởng sử dụng thống kê Chi – Square để xác định các pixel có tuân theo một điều kiện xác định hay không.

Giả sử X_{2k} và X_{2k+1}lần lượt là tần suất xuất hiện của pixel mang giá trị 2k và 2k+1(ở đây ta đánh giá pixel theo độ sáng nên $k \in \{0, \dots, 127\}$). Khi đó mỗi khi nhúng 1 bit dữ liệu vào trong pixel (ta quy ước mỗi pixel chỉ nhúng 1 bit dữ liệu)sẽ có thể làm cho một giá trị thuộc 2k trở thành 2k+1 hoặc ngược lại.

Giải thích tại sao chọn 2k và 2k+1: Bởi giá trị 2k sẽ có bit tận cùng là "0", mà kỹ thuật LSB sẽ thay vào giá trị bit ít quan trọng nhất là bit cuối cùng, chính vì vậy nếu bit tin nhắn là "1" hay "0" thì giá trị 2k sẽ thành 2k+1 hoặc là giữ nguyên 2k và tương tự với giá trị 2k+1 cũng vậy.

Khi đó với dãy bit được nhúng vào ảnh với tỉ lệ xuất hiện của "0" và "1" là 50/50 sẽ làm cho X_{2k} và X_{2k+1} tiến đến gần nhau(hay tần suất xuất hiện giữa 2k và 2k + 1 là bằng nhau), lúc này ta sẽ coi tần suất kì vọng của X_{2k} và X_{2k+1}là :

$$X^*_{2k} = \frac{X_{2k} + X_{2k+1}}{2}$$

Áp dụng Chi – Square Test ta sẽ tính được phân phối

$$X^2 = \sum_{i=0}^{127} \frac{(X_{2i} - X^*_{2i})^2}{X^*_{2i}}$$

Với hệ số bậc tự do là $v = 127 - 1 -$ số phần tử độ sáng 2k và 2k + 1 có tần suất bằng 0.

Lúc này xác suất để thành lập sự nghi ngờ đây là ảnh stego là

$$P\left(X^2 \geq X^2_{v;a}\right) = p = 1 - \int_0^{X^2_{v;a}} \frac{e^{-\frac{x}{2}} * x^{\frac{v}{2}-1}}{2^{\frac{v}{2}} * \Gamma\left(\frac{v}{2}\right)} dx [21]$$

Và ta có thể tra được giá trị trong Chi – Square Test Table với a là ngưỡng xác suất đáp ứng đủ điều kiện.

3.3.1 Cài đặt

Thư viện được sử dụng trong chương trình bao gồm:

- Scipy.stats : hỗ trợ tra cứu bảng phân phối X^2

Chương trình sẽ được chia thành 2 phần chính:

- Chia ảnh thành các khối block 18 * 18
- Đánh giá từng khối block với giá trị tới hạn phải đặt 95%

Các ảnh khi thực hiện kiểm tra sẽ được chuyển qua hệ mày Ycrcb để tính toán dựa trên độ sáng.

a) Chia ảnh thành các khối block

- Để chia ảnh thành các khối Block ta nhờ sự trợ giúp của hàm split, khi đó để chia thành khối 18 * 18 ta sẽ có như sau:

```
blocks = np.split(img, len(img) // block_size)
```

- Mục đích để có thể kiểm tra từng phần ảnh, tăng độ chuẩn xác

b) Đánh giá từng khối block

- Để đánh giá từng khối block, ta sẽ góm nhóm tất cả các màu có mặt trong khối theo độ sáng, và tách riêng giá trị chẵn lẽ ra làm 2 mảng :

```
for c in block:
    if c % 2 == 0:
        auxX[c // 2] += 1
    else:
        auxY[c // 2] += 1
```

- Sau đó đánh giá lần lượt 0 – 255 giá trị (tương ứng với độ sáng) rồi tính phân phối X^2 với từng giá trị chẵn lẻ tương ứng :

```
chi_square = calcChiSquare(auxX[i], (auxX[i] + auxY[i]) / 2)
```

- Cuối cùng tra bảng phân phối X^2 với giá trị tới hạn là 0,95

3.3.1 Đánh giá độ hiệu quả

Trong paper này, chúng tôi thực hiện chia nhỏ ảnh thành các block 18 x 18 để tăng độ hiệu quả trong việc thực hiện thống kê Chi – Square. Với yêu cầu giá trị tới hạn phải trên 95%, dưới đây là thực nghiệm chúng tôi đưa ra với 100 ảnh khác nhau :

	Số % ảnh Stego được phân tích xác suất trên ngưỡng xác định	Số % ảnh gốc bị phát hiện nhầm trên ngưỡng xác định
Ngưỡng 20%	85%	27%
Ngưỡng 50%	66%	2%
Ngưỡng 70%	62%	0%

Từ thực nghiệm chúng tôi kết luận rằng việc phương pháp này hiệu quả tốt nhất với ảnh xám và dễ bị phân tích sai với các hình ảnh nhiều màu, đặc biệt phương pháp này vẫn sẽ hiệu quả với các phương pháp nhúng dữ liệu tuần tự khác nhưng lại kém hiệu quả với các phương pháp nhúng pixel ngẫu nhiên.

3.4 NÂNG CAO HIỆU QUẢ THÔNG QUA ỨNG DỤNG MÃ HÓA

Như chúng ta đã biết, để nhúng dữ liệu vào hình ảnh, chúng ta phải mã hóa dữ liệu thành dạng bit (thông thường mã hóa từng ký tự thành một chuỗi 8 bit). Không có sự can thiệp nào khác ngoài điều này. Do đó, việc áp dụng Steganography chỉ giúp che giấu sự hiện diện của dữ liệu và nếu kẻ tấn công tìm thấy dữ liệu ẩn trong ảnh, chúng có thể đọc trực tiếp mà không cần bất kỳ lớp bảo vệ thứ cấp nào.

Điều này dẫn đến ý tưởng rằng để tăng tính bảo mật cho hình ảnh Stego, một trong những phương pháp dễ tiếp cận nhất là mã hóa dữ liệu trước khi nhúng nó vào hình ảnh.

Các hệ thống mã hóa hiện nay thường được chia thành hai loại: Mã hóa đối xứng và Mã hóa bất đối xứng.

3.4.1 Hệ mã hóa đối xứng

Mã hóa đối xứng sử dụng một khóa duy nhất Kcho cả chức năng mã hóa và giải mã. Điều này có nghĩa là chúng ta phải tạo khóa K để mã hóa dữ liệu và cũng ẩn khóa K trong ảnh để người nhận có thể giải mã dữ liệu. Điều này vô tình làm tăng kích thước của dữ liệu được nhúng vào hình ảnh, khiến kẻ tấn công dễ dàng phát hiện hình ảnh steganography hơn.

3.4.2 Hệ mã hóa bất đối xứng

Mã hóa bất đối xứng sử dụng hai khóa cho quá trình mã hóa và giải mã. Hai khóa này đi thành một cặp, bao gồm khóa riêng và khóa chung. Khóa riêng được giữ bí mật và được sử dụng để giải mã dữ liệu, trong khi khóa chung được chia sẻ với mọi người và được sử dụng để mã hóa dữ liệu.

Sự tồn tại của hai khóa này vô tình đáp ứng đủ mọi điều kiện để nâng cao hiệu quả của kỹ thuật giấu tin. Bằng cách sử dụng khóa chung của người nhận, chúng ta có thể mã hóa dữ liệu mà không cần phải bí mật gửi khóa cho người nhận và điều đó cũng không làm tăng kích thước của dữ liệu được nhúng.

Một số hệ thống mật mã khóa công khai nổi tiếng bao gồm RSA, AES...

CHƯƠNG 4. KẾT LUẬN

4.1 KẾT LUẬN

Lĩnh vực giấu tin hình ảnh tiếp tục phát triển với những nghiên cứu đang diễn ra nhằm giải quyết cả việc phát triển các kỹ thuật mới và cải tiến các phương pháp hiện có. Các cuộc khảo sát và đánh giá cung cấp một cái nhìn tổng quan toàn diện về công nghệ tiên tiến nhất trong lĩnh vực này.Trong bài viết này, chúng tôi đã đạt được mục tiêu trình bày tổng quan về các kỹ thuật steganographic và đánh giá chúng từ nhiều góc độ khác nhau. Ngoài ra, chúng tôi đã giới thiệu kỹ thuật Chi-Square Test để tấn công LSB và đánh giá tính phù hợp của các hệ thống mã hóa khác nhau nhằm nâng cao hiệu quả của kỹ thuật giấu tin.Về tính mới của bài báo này, chúng tôi đã đánh giá hiệu quả của kỹ thuật LSB và trải phổ thông qua dữ liệu đo được và nâng cao hiệu quả của các cuộc tấn công Chi-Square bằng cách chia hình ảnh thành các khối để đánh giá. Hơn nữa, việc đánh giá sự phù hợp của từng hệ thống mã hóa cũng đóng góp đáng kể cho bài viết này.Từ kết quả thực nghiệm, rõ ràng các kỹ thuật thông dụng vẫn còn những hạn chế và trong tương lai, việc nâng cao chất lượng để đảm bảo hiệu quả trong liên lạc bí mật giữa các tổ chức hoặc đơn vị lớn hơn như quân đội là cần thiết.

4.2 HƯỚNG PHÁT TRIỂN CỦA ĐỒ ÁN TRONG TƯƠNG LAI

Về mặt lý thuyết thì báo cáo này đã đưa ra đầy đủ và tổng quan về các kỹ thuật giấu tin, nhưng lại chưa đưa ra các giải pháp cần có để có thể cải thiện hoặc nâng cao kỹ thuật mà chỉ đánh giá chung. Không chỉ vậy, rõ ràng việc đánh giá theo các số liệu đo được chỉ có thể mang tính chất tham khảo, cần phải thêm nhiều công cuộc nghiên cứu nữa trong tương lai để đưa ra được những kết quả chính xác nhất.

TÀI LIỆU THAM KHẢO

[1] Ker, A.D.: Steganalysis of lsb matching in grayscale images. IEEE Signal Processing Letters 12(6), 441–444 (2007)

[2] Kim, H.-K., Moon, T.: Jpeg steganography using syndrome-trellis codes. Proceedings of the IEEE International Conference on Image Processing, 344–348 (2016). IEEE

[3] Cox, I., Miller, M., Bloom, J., Fridrich, J., Kalker, T.: Digital watermarking and steganography. Morgan Kaufmann (2007)

[4] Holub, V., Fridrich, J.: Designing steganographic distortion using directional filters. Proceedings of the IEEE International Workshop on Information Forensics and Security, 234–239 (2012). IEEE

[5] Johnson, N.F., Jajodia, S.: Exploring steganography: Seeing the unseen. Computer 31(2), 26–34 (1998)

[6] Judge, J.C.: Steganography: past, present, future. SANS white paper 30 (2001)

[7] Kahn, D.: The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. Simon and Schuster (1996)

[8] Hartung, F., Girod, B.: Fast public-key watermarking of compressed video. In: Proceedings of International Conference on Image Processing, vol. 1, pp. 528–531 (1997). IEEE

[9] Artz, D.: Digital steganography: Hiding data within data. IEEE Internet Computing 5(3), 75–80 (2001)

[10] Majeed, M.A., Sulaiman, R., Shukur, Z., Hasan, M.K.: A review on text steganography techniques. Mathematics 9(21), 2829 (2021)

[11] KHALDI, A.: Steganographic techniques classification according to image format. International Annals of Science 8(1), 143–149 (2020)

[12] Erdem, E.: Graphics interchange format (gifs) as micro movies. Master's thesis, Bilkent Universitesi (Turkey) (2015)

[13] Krishna, V., Kumbharana, C.: An analysis of different image formats for steganography. Int. J. Emerg. Technol. Innov. Res.(JETIR) 8(10), 299–307 (2021)

[14] Wang, P., Zhong, H., Feng, Y., Gong, L., Tang, Y., Lu, Z.-M., Wang, L.: Covert communication through robust fragment hiding in a large number of images. Sensors 24(2), 627 (2024)

[15] Morkel, T., Eloff, J.H., Olivier, M.S.: An overview of image steganography. In: 26 ISSA, vol. 1, pp. 1–11 (2005)

[16] Dutta, S., Saini, K.: Securing data: A study on different transform domain techniques. WSEAS Transactions on Systems And Control 16(10.37394), 23203–2021 (2021)

[17] Bender, W., Gruhl, D., Morimoto, N., Lu, A.: Techniques for data hiding. IBM systems journal 35(3.4), 313–336 (1996)

[18] Hartung, F., Girod, B.: Fast public-key watermarking of compressed video. In: Proceedings of International Conference on Image Processing, vol. 1, pp. 528–531 (1997). IEEE

[19] Hamghalam, M., Mirzakuchaki, S.: Statistical restoration to resist chi-square attack based on heuristic algorithms in steganalytic systems. In: 2011 7th Iranian Conference on Machine Vision and Image Processing, pp. 1–4 (2011). IEEE

[20] Westfeld, A., Pfitzmann, A.: Attacks on steganographic systems: Breaking the steganographic utilities ezstego, jsteg, steganos, and s-tools-and some lessons learned. In: International Workshop on Information Hiding, pp. 61–76 (1999). Springer

[21] Daniel, W.W., Cross, C.L.: Biostatistics: a Foundation for Analysis in the Health Sciences. Wiley(2018)

[22] Panpaliya, R., Nemade, J., Zende, P., Bhoyalkar, S., Shaikh, S.: Secret communication using multi-image steganography for military purposes, 229–235 (2022)

[23] McAteer, I., Ibrahim, A., Zheng, G., Yang, W., Valli, C.: Integration of biometrics and steganography: A comprehensive review. Technologies 7(2), 34 (2019)

[24] Al-Dmour, H.S.T.: Enhancing information hiding and segmentation for medical images using novel steganography and clustering fusion techniques. PhD thesis (2018)

[25] Karampidis, K., Kavallieratou, E., Papadourakis, G.: A review of image steganalysis techniques for digital forensics. Journal of information security and applications 40, 217–235 (2018)

PHẦN CODE THAM KHẢO

Phần I. CODE KỸ THUẬT LSB

```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Net.NetworkInformation;
using System.Reflection;
using System.Reflection.Metadata;
using System.Text;
using ColorMine.ColorSpaces;
using ColorMine.ColorSpaces.Comparisons;
using static System.Reflection.Metadata.BlobBuilder;
using static System.Runtime.InteropServices.JavaScript.JSType;

class Program
{
    //Embed the message in the first 2 bits of the pixel
    public static void Stego(int index, string mess, ref int[,] stoArray)
    {
        Console.Write(stoArray[index, 2] + " " + mess[0] + " " +
stoArray[index, 3] + " " + mess[1] + " " + stoArray[index, 4] + " " +
mess[2]);
        stoArray[index, 2] &= 254;
        stoArray[index, 3] &= 254;
        stoArray[index, 4] &= 254;

        stoArray[index, 2] += Convert.ToInt32(mess.Substring(0, 1), 2);
        stoArray[index, 3] += Convert.ToInt32(mess.Substring(1, 1), 2);
        stoArray[index, 4] += Convert.ToInt32(mess.Substring(2, 1), 2);
        Console.WriteLine(" " + stoArray[index, 2] + " " + stoArray[index,
3] + " " + stoArray[index, 4]);
    }

    public static string Decrypt(int[,] StoArray)
    {
        StringBuilder len = new StringBuilder();
        //Console.WriteLine(StoArray[1, 0]);
        for (int i = 0; i <= 3; i++)
        {
            len.Append(Convert.ToString(StoArray[i, 0] - (StoArray[i, 0] &
254), 2).PadLeft(1, '0'));
            len.Append(Convert.ToString(StoArray[i, 1] - (StoArray[i, 1] &
254), 2).PadLeft(1, '0'));
            len.Append(Convert.ToString(StoArray[i, 2] - (StoArray[i, 2] &
254), 2).PadLeft(1, '0'));
        }
        int lenStego = Convert.ToInt32(len.ToString(), 2);

        StringBuilder mess = new StringBuilder();
        for (int i = 4; i < lenStego * 8 / 3 + 4 + 1; i++)
        {
            mess.Append(Convert.ToString(StoArray[i, 0] - (StoArray[i, 0] &
254), 2).PadLeft(1, '0'));
            mess.Append(Convert.ToString(StoArray[i, 1] - (StoArray[i, 1] &
254), 2).PadLeft(1, '0'));
            mess.Append(Convert.ToString(StoArray[i, 2] - (StoArray[i, 2] &
254), 2).PadLeft(1, '0'));
        }
        //Console.WriteLine(mess.ToString());
        return mess.ToString().Remove(0, (lenStego * 8 / 3 + 1) * 3 -
lenStego * 8);
```

```
}

public static string BinaryToString(string data)
{
    byte[] bytes = new byte[data.Length / 8];

    for (int i = 0; i < data.Length / 8; i++)
    {
        string binaryString = data.Substring(i * 8, 8);
        bytes[i] = Convert.ToByte(binaryString, 2);
    }

    string convertedString = Encoding.UTF8.GetString(bytes);
    return convertedString;
}
public static string StringToBinary(string data)
{
    byte[] bytes = Encoding.UTF8.GetBytes(data);
    StringBuilder binary = new StringBuilder();

    foreach (byte b in bytes)
    {
        binary.Append(Convert.ToString(b, 2).PadLeft(8, '0'));
    }
    return binary.ToString();
}

public static void Main(string[] args)
{
    Console.WriteLine("1. Hide information in photos");
    Console.WriteLine("2. Extract hidden information in images");
    Console.Write("Choose option: ");
    int ans = Convert.ToInt32(Console.ReadLine());

    if (ans == 1)
    {
        //Path to the cover image
        string imagePath = "image.png";
        Bitmap image = (Bitmap)Image.FromFile(imagePath);

        int width = image.Width;
        int height = image.Height;
        int index = 0;

        int[,] stoArray = new int[width * height, 5];

        for (int i = 0; i < width; i++)
        {
            for (int j = 0; j < height; j++)
            {
                Color pixel = image.GetPixel(i, j);

                stoArray[index, 0] = i;
                stoArray[index, 1] = j;
                stoArray[index, 2] = pixel.R;
                stoArray[index, 3] = pixel.G;
                stoArray[index, 4] = pixel.B;

                index++;
            }
        }

        //Path to file secret message
        string messPath = "Import.txt";
        string mess = File.ReadAllText(messPath);
```

```
            mess = StringToBinary(mess);
            string steLenMess = Convert.ToString(mess.Length / 8,
2).PadLeft(12, '0');
            mess = mess.PadLeft((3 - (mess.Length % 3)) + mess.Length, '0');
            index = 0;
            double total = 0.0;
            for (int i = 0; i < steLenMess.Length; i += 3)
            {
                Stego(index, steLenMess.Substring(i, 3), ref stoArray);

                var rgbColor2 = new Rgb { R = stoArray[index, 2], G =
stoArray[index, 3], B = stoArray[index, 4] };

                index++;
            }

            for (int i = 0; i < mess.Length; i += 3)
            {
                var rgbColor1 = new Rgb { R = stoArray[index, 2], G =
stoArray[index, 3], B = stoArray[index, 4] };
                Stego(index, mess.Substring(i, 3), ref stoArray);
                var rgbColor2 = new Rgb { R = stoArray[index, 2], G =
stoArray[index, 3], B = stoArray[index, 4] };
                var labColor1 = rgbColor1.To<Lab>();
                var labColor2 = labColor1.To<Lab>();
                total += labColor1.Compare(labColor2, new
Cie1976Comparison());
                index++;
            }

            Bitmap newImage = new Bitmap(width, height,
PixelFormat.Format24bppRgb);
            index = 0;

            for (int i = 0; i < width; i++)
            {
                for (int j = 0; j < height; j++)
                {
                    Color pixel = Color.FromArgb(stoArray[index, 2],
stoArray[index, 3], stoArray[index, 4]);
                    newImage.SetPixel(i, j, pixel);
                    index++;
                }
            }

            //Path to file secret mess
            string outputPath = "stego_image.png";
            newImage.Save(outputPath, ImageFormat.Png);
        }



        else if (ans == 2)
        {
            // Path to stego image
            string stegoImage = "stego_new_image.png";
            Bitmap image = (Bitmap)Image.FromFile(stegoImage);

            int width = image.Width;
            int height = image.Height;
            int index = 0;

            int[,] stoArray = new int[width * height, 3];
```

```
            for (int i = 0; i < width; i++)
            {
                for (int j = 0; j < height; j++)
                {
                    Color pixel = image.GetPixel(i, j);

                    stoArray[index, 0] = pixel.R;
                    stoArray[index, 1] = pixel.G;
                    stoArray[index, 2] = pixel.B;

                    index++;
                }
            }

            string mess = Decrypt(stoArray);

            // Path to decrypted message txt
            string messPath = "Export.txt";
            File.WriteAllText(messPath, BinaryToString(mess));
        }

        else
        {
            Console.WriteLine("ERROR!!!");
        }
    }
}
```

Phần II. CODE KỸ THUẬT SPREAD SPECTRUM

```
import numpy as np
import cv2
import random
import math

def main():
    print("1. Embed message in photo")
    print("2. Decode the image")
    n = int(input("Enter selection: "))

    if n == 1:
        with open("message.txt", "r") as file:
            message = file.read()
        message_length = len(message)
        G = 4

        image = cv2.imread('image.jpg')
        image_ycbcr = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
        height, width, _ = image.shape
        n = width
        array = np.zeros(n)
        message_length = 4
        array_mess = [-1 if bit == '0' else 1 for bit in message]

        signa = int(height / message_length)
```

```
        for j in range(height):
            if math.floor(j / signa) == message_length:
                break

            random.seed(j // signa)
            for i in range(width):
                array[i] = random.randint(0, 1)
                if array[i] == 0:
                    array[i] = -1

            for i in range(width):
                y = image_ycbcr[j, i, 0]
                y = y + G * array[i] * array_mess[int(j / signa)]
                y = np.clip(y, 0, 255)
                image_ycbcr[j, i, 0] = y

        dem = 0
        new_array = np.empty((height, width))
        for i in range(height):
            random.seed(i // signa)
            for k in range(width):
                array[k] = random.randint(0, 1)
                if(array[k] == 0):
                    array[k] = -1
            new_array[i] = array

        image_bgr = cv2.cvtColor(image_ycbcr, cv2.COLOR_YCrCb2BGR)
        cv2.imwrite('stego_image.jpg', image_bgr)

    elif n == 2:
        # Decode the stego image
        image_bgr = cv2.imread('stego_image.jpg')
        if image_bgr is None:
            print("Error: Could not load stego image.")
            return

        message_length = 4
        G = 4
        height, width, _ = image_bgr.shape
        signa = int(height / message_length)
        array = np.zeros(width)

        # Convert to YCbCr color space
        new_array = np.empty((height, width))
        for i in range(height):
            random.seed(i // signa)
            for k in range(width):
                array[k] = random.randint(0, 1)
                if(array[k] == 0):
                    array[k] = -1
            new_array[i] = array
```

```
        image_ycbcr = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2YCrCb)

        decoded_message = ""
        for j in range(message_length):
            ans = 0

            for i in range(message_length):
                row = image_ycbcr[signa*i : signa*(i+1), :, 0]
                row = row.flatten()
                array = new_array[signa*j : signa*(j+1)]
                array = array.flatten()
                correlation_coefficient = np.corrcoef(array, row)[0, 1]
                ans += correlation_coefficient

            sign_ans = np.sign(ans)
            decoded_message += '1' if sign_ans > 0 else '0'

        #print to file
        with open('decrypted_message.txt', 'w') as file:
            file.write(decoded_message)

if __name__ == "__main__":
    main()
```

Phần III. CODE KỸ THUẬT CHI – SQUARE TEST

```
import numpy as np
import cv2 as ocv
import scipy.stats as stats
import glob

def calcChiSquare(T, auxZ):
    chi_square = 0
    for observed, expected in zip(T, auxZ):
        if expected != 0:
            chi_square += ((observed - expected) ** 2) / expected
    return chi_square

def chiSquareAttack(image_path, block_size = 324):  # threshold for dof=2
    img_r = ocv.imread(image_path, ocv.IMREAD_GRAYSCALE)
    if img_r is None:
        raise ValueError("Image not found or unable to read")

    img = np.ndarray.flatten(img_r)
    padding = block_size - len(img) % block_size
    if padding != block_size:
        img = np.append(img, np.full(padding, 0))
```

```
    # Chia ảnh thành các block
    blocks = np.split(img, len(img) // block_size)

    suspicious_blocks = 0

    for block in blocks:
        auxX = [0] * 128
        auxY = [0] * 128

        for c in block:
            if c % 2 == 0:
                auxX[c // 2] += 1
            else:
                auxY[c // 2] += 1

        T = []
        auxZ = []
        v = -2
        for i in range(128):
            if auxX[i] + auxY[i] != 0:
                T.append(auxX[i])
                auxZ.append((auxX[i] + auxY[i]) / 2)
                v += 1
        chi_square = calcChiSquare(T, auxZ)
        #print(chi_square)

        if chi_square >= stats.chi2.ppf(0.95, df=v):
            suspicious_blocks += 1
    #print(f"Suspicious blocks: {suspicious_blocks}/{len(blocks)}")
    print(f"Percentage of suspicious blocks: {(suspicious_blocks /
len(blocks)) * 100:.2f}%")

# Đường dẫn tới ảnh cần kiểm tra
folder_path = "test/*.png"
image_paths = glob.glob(folder_path)
for image_path in image_paths:
    print(image_path.split('/')[-1])
    chiSquareAttack(image_path)
    print()
```