BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

ĐỒ ÁN TỐT NGHIỆP

NGHIÊN CỨU MỘT SỐ PHƯƠNG PHÁP TẠO HỘP THẾ VÀ ĐỀ XUẤT ỨNG DỤNG

Ngành: An toàn thông tin
Mã số: 7.48.02.02

Sinh viên thực hiện:
Nguyễn Xuân Hiệu
Lớp: AT17A

Người hướng dẫn:
ThS Hoàng Phương Thức
TTCNTT, Ngân hàng Agribank

Hà Nội, 2024

BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

ĐỒ ÁN TỐT NGHIỆP

NGHIÊN CỨU MỘT SỐ PHƯƠNG PHÁP TẠO HỘP THẾ VÀ ĐỀ XUẤT ỨNG DỤNG

Ngành: An toàn thông tin
Mã số: 7.48.02.02

Sinh viên thực hiện:
Nguyễn Xuân Hiệu
Lớp: AT17A

Người hướng dẫn:
ThS Hoàng Phương Thức
TTCNTT, Ngân hàng Agribank

Hà Nội, 2024

MỤC LỤC

DANH MỤC KÝ HIỆU VÀ VIẾT TẮT

$\oplus$	Phép XOR
AES	Advanced Encryption Standard
ANF	Algebraic Normal Form
DES	Data Encryption Standard
IDEA	International Data Encryption Algorithm
NIST	National Institute of Standards and Technology
PC	Propagation Criterion
SAC	Strict Avalanche Criterion
S-box	Substitution box
SPN	Substitution Permutation Network

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

LỜI CÁM ƠN

Trước hết em xin gửi lời cảm ơn chân thành đến các thầy, cô giáo trong khoa an toàn thông tin – Học viện Kỹ thuật mật mã đã tạo điều kiện tốt nhất để em hoàn thành đồ án tốt nghiệp. Đặc biệt, em xin gửi đến Thạc sĩ Hoàng Phương Thức – TTCNTT, Ngân hàng Agribank và Tiến sĩ Hoàng Đức Thọ – Chủ nhiệm Khoa ATTT đã tận tình hướng dẫn, giúp đỡ em hoàn thành báo cáo đồ án này lời cảm ơn sâu sắc nhất.

Vì kiến thức bản thân còn hạn chế, trong báo cáo này em không tránh khỏi những sai sót, kính mong nhận được những ý kiến đóng góp từ thầy cô để em có thể bổ sung kiến thức cho bản thân và thực hiện những nghiên cứu chuyên sâu sau này.

LỜI NÓI ĐẦU

Cùng với sự phát triển nhanh chóng của Internet, các ứng dụng giao dịch điện tử ngày càng phát triển thì nhu cầu bảo vệ thông tin trên các hệ thống ngày càng được quan tâm. Một trong những phương pháp phổ biến và hữu ích trong việc bảo vệ an toàn thông tin chính là sử dụng mật mã.

Các ứng dụng mật mã để bảo vệ thông tin được sử dụng trong nhiều lĩnh vực khác nhau như: an ninh, quân sự, ngân hàng, thương mại… Và ngày càng có nhiều những nghiên cứu về các thuật toán mật mã để ứng dụng và giải quyết vấn đề an toàn thông tin một cách hiệu quả. Phương pháp tạo hộp thế (S-box) là một trong những nghiên cứu đang được nhiều người quan tâm. Vì vậy em chọn đề tài "Nghiên cứu về một số phương pháp tạo hộp thế và đề xuất ứng dụng" để làm đề tài nghiên cứu trong đồ án

Mục tiêu của đề tài là: Nghiên cứu tổng quan về mật mã và hộp thế, tầm quan trọng của hộp thế trong mật mã để đảm bảo an toàn, an ninh thông tin. Nghiên cứu về một số phương pháp tạo hộp thế và đi sâu vào nghiên cứu phương pháp tạo hộp thế dựa trên ánh xạ hỗn loạn một chiều. Ứng dụng hộp thế trong thuật toán AES để mã hóa dữ liệu, hình ảnh và giải mã chúng, so sánh các điểm giống và khác nhau khi mã hóa ảnh bằng hộp thế mặc định và hộp thế được tạo ra.

Nội dung chính của đồ án bao gồm các chương sau:

Chương 1: Tổng quan về mật mã và hộp thế

Trong chương này sẽ giới thiệu tổng quan về mật mã, hệ mật mã khối và chỉ ra được vai trò của hộp thế (S-box) trong mật mã khối. Giới thiệu về hệ mật AES và hộp thế trong thuật toán mã hóa AES. Trình bày về hàm Boolean và các tính chất mật mã của hàm Boolean, các tính chất mật mã của hộp thế.

Chương 2: Một số phương pháp tạo hộp thế

Nội dung của chương 2 là tìm hiểu về một số phương pháp tạo hộp thế và đi sâu vào 1 phương pháp cụ thể: phương pháp tạo hộp thế động dựa trên ánh xạ hỗn loạn rời rạc một chiều pure

Chương 3: Xây dựng chương trình thực thi

Chương 3 sẽ bao gồm các nội dung: cài đặt chương trình thử nghiệm tạo S-box và áp dụng hộp thế được tạo ra vào thuật toán AES để mã hóa và giải mã dữ liệu, hình ảnh. Đánh giá một số tính chất mật mã của hộp thế và hình ảnh mã hóa được tạo ra và sự khác biệt của chúng so với hộp thế mặc định của AES

SINH VIÊN THỰC HIỆN ĐỒ ÁN

CHƯƠNG 1 TỔNG QUAN VỀ MẬT MÃ VÀ HỘP THẾ

1.1 Tổng quan về hệ mật

1.1.1 Định nghĩa

Hệ mật mã: Hệ mật mã được định nghĩa là một bộ năm (P,C,K,E,D). Trong đó : [4]

1. P là tập hợp hữu hạn các bản rõ
2. C là tập hữu hạn các bản mã
3. K là tập hữu hạn các khóa
4. E là tập các hàm mã hóa
5. D là tập các hàm giải mã

Với mỗi $k \in K$, có một hàm mã hóa $e_k \in E, e_k : P \to C$ và một hàm giải mã $d_k \in D, d : C_k \to P$ sao cho $d_k(e_k(x)) = x, \forall x \in P$

Bản rõ: Chứa các xâu ký tự gốc, thông tin trong bản rõ là thông tin cần mã hoá để giữ bí mật.

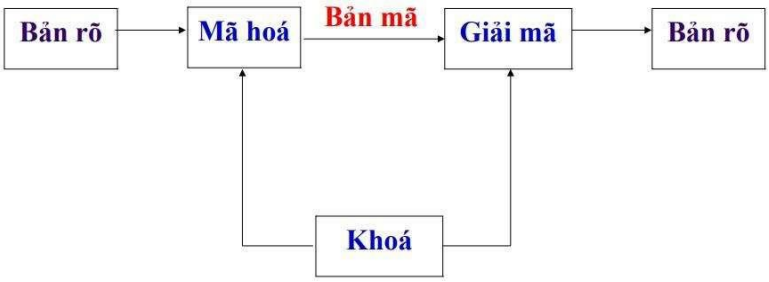
Bản mã: Chứa các ký tự sau khi đã được mã hoá, mà nội dung được giữ bí mật.

Mã hóa: là quá trình sử dụng những quy tắc được quy định trong một hệ mã để biến đổi thông tin ban đầu (bản rõ) thành bản mã.

Giải mã: là quá trình ngược lại với mã hóa, tức là sử dụng những quy tắc được quy định trong hệ mã để biến đổi nội dung bản mã về thông tin ban đầu.

Khoá K: là thông tin tham số dùng để mã hoá và giải mã chỉ có người gửi và người nhận biết. Khóa là độc lập với bản rõ và có độ dài phù hợp với yêu cầu bảo mật.

Quá trình mã hoá và giải mã được thể hiện trong hình 1.1



Hình 1.1: Quá trình mã hóa và giải mã thông tin

Thám mã là nghiên cứu cách phá các hệ mật nhằm phục hồi bản rõ ban đầu từ bản mã, nghiên cứu các nguyên lí và phương pháp giải mã mà không biết khóa.

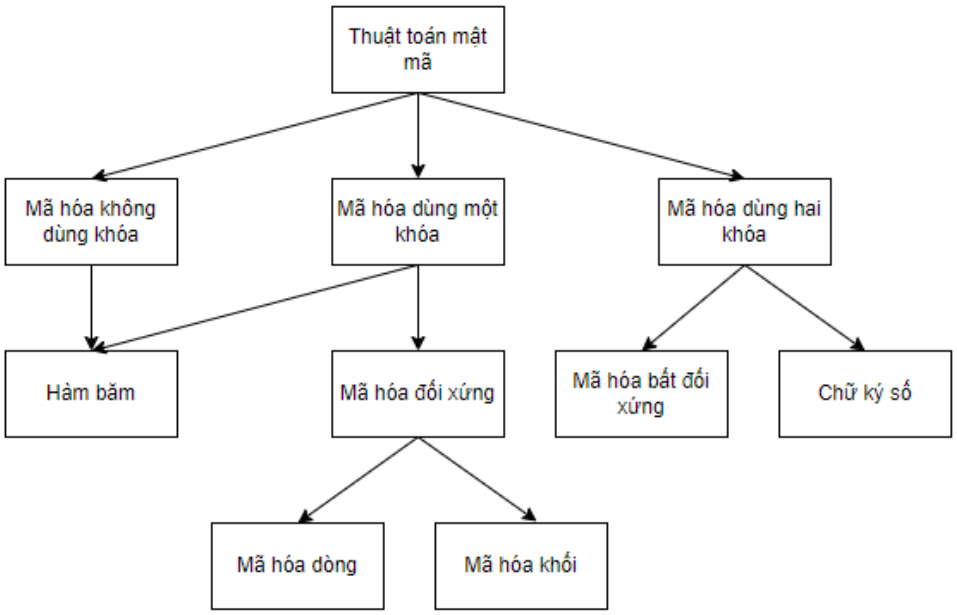
Thám mã có thể chia làm 2 loại:

Loại thám mã tích cực: là việc thám mã sau đó tìm cách làm sai lạc các dữ liệu truyền, nhận hoặc các dữ liệu lưu trữ phục vụ mục đích của người thám mã.

Loại thám mã thụ động: là việc thám mã để có được thông tin về bản rõ phục vụ mục đích của người thám mã.

1.1.2 Phân loại hệ mật

Dựa vào khóa mã hóa và cách thức tiến hành, hệ mật được phân loại thuật toán mật mã như trong hình 1.2 [4]



Hình 1.2: Sơ đồ phân loại thuật toán mật mã

Hàm hash (hàm băm): Trong ngành mật mã học, hàm băm có một số tính chất bảo mật nhất định (một chiều, kháng tiền ảnh…) để phù hợp việc sử dụng trong nhiều ứng dụng bảo mật thông tin đa dạng, chẳng hạn như kiểm tra tính nguyên vẹn của thông điệp (message integrity). Một hàm băm nhận đầu vào là một xâu ký tự có độ dài tùy ý và tạo ra kết quả là một xâu ký tự có độ dài cố định, đôi khi được gọi là tóm tắt thông điệp (message digest)

Mật mã đối xứng: Là một hệ mật trong đó một khóa giống nhau sẽ vừa được dùng để mã hóa, vừa được dùng để giải mã các thông điệp. Thực tế thì hai khóa (mã hóa, giải mã) có thể khác nhau, trong trường hợp này thì một khóa nhận được từ khóa kia bằng phép tính toán đơn giản.

Mật mã đối xứng thì được chia làm hai loại: mã dòng và mã khối.

Mã dòng: Mã hóa dòng làm việc trên từng bit của dòng dữ liệu và quá trình biến đổi thay đổi theo quá trình mã hóa.

Mã khối: Mã hóa khối là những thuật toán mã hóa đối xứng hoạt động trên những khối thông tin có độ dài xác định (gọi là block) với những chuyển đổi xác định. Chẳng hạn một thuật toán mã hóa khối có thể xử lý khối 128 bit đầu vào và biến nó thành khối 128 bit đầu ra. Quá trình chuyển đổi còn sử dụng thêm một tham số nữa: khóa bí mật để cá biệt hóa quá trình. Việc giải mã cũng diễn ra tương tự: xử lý khối mã hóa 128 bit cùng với khóa để trả về khối 128 bit bản rõ ban đầu.

Mật mã bất đối xứng: Có thể hiểu là người ta dùng hai khoá khác nhau để khoá và mở khóa thông tin bí mật. Public key sẽ được công khai, và được gửi đi đến đối tượng cần mã hoá thông tin, còn private key được giữ bí mật được dùng để giải mã thông điệp hoặc ký số

Chữ ký số: Là phương pháp ký một bức điện dưới dạng điện tử, để đảm bảo tính nguyên vẹn, xác thực và chống chối bỏ. Trong mật mã bất đối xứng, chữ ký điện tử cũng dùng thuật toán mật mã với hai khóa, khóa công khai tính dễ dàng từ khóa bí mật, còn khóa bí mật thì rất khó hầu như không tính được từ khóa công khai. Nhưng khác với mật mã bất đối xứng là quá trình ký bức điện dùng khóa mật,

còn quá trình kiểm tra bức điện dùng khóa công khai. Và khóa mật ngoài chủ của bức điện thì không ai biết được, nhờ tính chất này mà chữ ký điện tử có chức năng đảm bảo tính chống chối bỏ của thông tin

1.1.3 Vai trò của hệ mật

Các ứng dụng của mật mã học trong an toàn thông tin rất đa dạng và phong phú, tùy vào tính đặc thù của mỗi hệ thống bảo vệ thông tin mà ứng dụng sẽ có các tính năng với đặc trưng riêng. Mật mã đảm bảo những tính chất sau trong an toàn thông tin:

Tính bí mật (confidentiality/privacy): đảm bảo thông tin chỉ được tiết lộ cho những ai được phép.

Tính toàn vẹn (integrity): tính chất này đảm bảo thông tin không thể bị thay đổi mà không bị phát hiện. Tính chất này không đảm bảo thông tin không bị thay đổi, nhưng một khi nó bị thay đổi thì người nhận được thông tin có thể biết được là thông tin đã bị thay đổi. Hàm băm thường được dùng để đảm bảo tính toàn vẹn cho thông tin.

Tính xác thực (authentication): người gửi (hoặc người nhận) có thể chứng minh đúng họ. Người ta có thể dụng một password, một challenge dựa trên một thuật toán mã hóa hoặc một bí mật chia sẻ giữa hai người để xác thực. Sự xác thực này có thể thực hiện một chiều (one-way) hoặc hai chiều (mutual authentication).

Tính chống chối bỏ (non-repudiation): người gửi hoặc nhận sau này không thể chối bỏ việc đã gửi hoặc nhận thông tin. Thông thường điều này được thực hiện thông qua một chữ ký điện tử (electronic signature)

1.2 Tổng quan về mã khối

1.2.1 Giới thiệu chung

Các hệ mật mã cổ điển có đặc điểm chung là từng ký tự của bản rõ được mã hoá tách biệt. Điều này làm cho việc thám mã trở lên dễ dàng hơn. Chính vì vậy, trên thực tế người ta hay dùng một kiểu mật mã khác, trong đó từng khối ký tự của

bản rõ được mã hóa cùng một lúc như là một đơn vị mã hoá đồng nhất, gọi là mã khối.

Quá trình mã hoá bao gồm 2 thuật toán: Mã hoá - ký hiệu E và giải mã - ký hiệu D. Cả 2 thuật toán đều tác động lên một khối đầu vào n bit sử dụng một khoá k bit để cho ra một khối đầu ra n bit.

Độ dài của khối thông tin, ký hiệu là n, thông thường là cố định ở 64 hoặc 128 bit. Một số thuật toán có độ dài khối thay đổi nhưng không phổ biến. Tính đến trước những năm 1990 thì độ dài 64 bit thường được sử dụng. Từ đó trở về sau thì khối 128 bit được sử dụng rộng rãi hơn. Trong các chế độ mã hoá khối thì người ta thường phải bổ sung thêm một số bit cho văn bản (padding) để văn bản chứa số nguyên lần các khối.

Hầu hết các thuật toán mã hóa khối sử dụng lặp đi lặp lại các hàm đơn giản. Mỗi chu kỳ lặp được gọi là một vòng và thông thường các thuật toán có từ 4 tới 32 vòng.

Các thành phần sử dụng trong thuật toán là các hàm toán học, các hàm logic (đặc biệt là hàm XOR), hộp thế (S-box) và các phương pháp hoán vị.

1.2.2 Một số hệ mã khối

Có thể kể ra một số hệ mã sau: DES (1977), 3-DES (1985), IDEA (1990), GOST, AES (2001)…

Trong đó DES được sử dụng rộng rãi và có tầm quan trọng vô cùng to lớn. DES có vai trò bảo mật thông tin trong quá trình truyền và lưu trữ thông tin. DES cũng được sử dụng để kiểm tra tính xác thực của mật khẩu truy cập vào một hệ thống (hệ thống quản lý bán hàng, quản lý thiết bị viễn thông, ...), hay tạo và kiểm tính hợp lệ của một mã số bí mật (thẻ internet, thẻ điện thoại di động trả trước), hoặc của một thẻ thông minh (thẻ tín dụng, thẻ payphone...).

Tuy nhiên DES cũng có những điểm yếu nhất định của nó, đó chính là tính bù và khoá yếu. [4]

Tính bù: Ký hiệu $\overline{u}$ là phần bù của u (ví dụ: 0100101 và 1011010 là bù của nhau) thì DES có tính chất sau: $y = DES_z(x) \Rightarrow \overline{y} = DES_{\overline{z}}(\overline{x})$.

Cho nên nếu biết MÃ y được mã hóa từ TIN x với khoá z thì ta suy ra $\overline{y}$ được mã hoá từ TIN $\overline{x}$ với khoá $\overline{z}$. Tính chất này chính là một điểm yếu của DES bởi vì nhờ đó kẻ địch có thể loại trừ một nửa số khoá cần phải thử khi tiến hành phép thử - giải mã theo kiểu tìm kiếm vét cạn không gian khoá.

Khóa yếu: khóa yếu trong DES là các khóa đặc biệt khiến thuật toán có độ an toàn kém hơn so với bình thường. Với các khóa yếu, mã hóa hai lần bằng cùng một khóa sẽ trả về chính bản rõ ban đầu. Điều này tạo điều kiện cho các kiểu tấn công vì DES trở thành dạng "mã hóa đồng nhất" (self-inverse), tức là:

$$DES(DES(P,K),K) = P$$

DES có 4 khóa yếu là: 0000000000000000, FFFFFFFFFFFFFFFF, 0101010101010101 và FEFEFEFEFEFEFEFE

Hiện nay có thể nói rằng DES đã bị phá vỡ, và vượt lên trên các tấn công phân tích mã, lần lượt các hệ mã khối mới với ưu thế mới đã ra đời đáp ứng có độ an toàn chống lại được các tấn công thực tế và hiệu quả trong sử dụng. Tiêu chuẩn mã dữ liệu tiên tiến AES nhằm thay cho DES là một ví dụ.

Vào năm 2000, cơ quan quản lý về chuẩn và công nghệ của Mỹ, NIST (National Institute of Standard and Technology), đã tổ chức một cuộc thi để chọn một hệ mật mã mới thay thế cho DES. Hệ mã Rijndael đã được chọn và được công bố (2002) như là chuẩn mật mã mới thay thế cho DES, với tên gọi là Advanced Encryption Standard (AES). Hệ mã này được phát triển bởi 2 nhà khoa học Bỉ, Joan Daemen và Vincent Rijmen (vì vậy tên gọi Rijndael được tạo ra từ việc ghép tiền tố tên họ 2 người này).

Advanced Encryption Standard (AES) được xây dựng trên cấu trúc mạng thay thế - hoán vị (Substitution-Permutation Network - SPN). Số vòng lặp của AES là một tham số xác định trên cơ sở kích thước khoá: 10 vòng lặp cho khóa 128 bit, 12 cho 192 bit, 14 cho 256 bit.

Hệ mật AES hiện nay là một thuật toán mã hoá đang được sử dụng thịnh hành trên thế giới và dù cho có bị tấn công bởi "tấn công kênh kề", AES vẫn chưa bị phá vỡ bởi theo nghiên cứu thì tại thời điểm này nguy cơ tấn công này không thực sự nguy hiểm và có thể bỏ qua. AES hiện nay cần được cải tiến hơn nữa để giữ vị trí an toàn trong các thuật toán mã hoá thông dụng.

1.2.3 Độ an toàn của mã khối

Để mã hoá khối an toàn cần đảm bảo các điều kiện sau:

Độ an toàn của khối: Khối phải có độ dài đủ để chống lại các phương pháp phân tích thống kê và ngăn việc một số khối nào đó xuất hiện nhiều hơn các khối khác. Mặt khác sự phức tạp của thuật toán tăng theo hàm mũ với độ dài khối. Với khối có độ dài 64 bit là đủ độ an toàn.

Không gian khóa: Khoá phải có độ dài đủ để chống lại các phương pháp vét cạn khoá (chống khả năng thử các khoá được sinh ra từ (N) bit khoá cho trước). Tuy nhiên khóa phải đủ ngắn để việc tạo khóa, phân phối và lưu trữ khóa được dễ dàng

Khi thiết kế một hệ mã khối, phải đảm bảo hai yêu cầu là sự xáo trộn và sự khuếch tán. [4]

Sự xáo trộn (confusion): sự phụ thuộc giữa bản rõ và bản mã phải thực sự phức tạp để gây khó khăn đối với việc tìm quy luật thám mã. Mối quan hệ này tốt nhất là phi tuyến.

Sự khuếch tán (diffusion): mỗi bit của bản rõ và khóa phải ảnh hưởng lên càng nhiều bit của bản mã càng tốt.

1.3 Một số tính chất mật mã của hàm boolean

Một trong các thành phần cơ bản của hệ thống mật mã đối xứng là các phép biến đổi phi tuyến, nó xác định độ an toàn chống lại các phương pháp thám mã. Các phép biến đổi phi tuyến được xây dựng trên cơ sở các hàm Boolean phi tuyến. Để đảm bảo an toàn, các hàm Boolean, được dùng trong các hộp thế, phải thỏa mãn

các tính chất nhất định. Các tham số cơ bản phản ánh độ an toàn của các hàm Boolean phi tuyến là:

1. Tính chất cân bằng

2. Độ phi tuyến

3. Tiêu chuẩn phân phối và hiệu ứng thác chặt

4. Bậc đại số

Sau đây là một số định nghĩa và khái niệm cơ bản liên quan:

Định nghĩa 1.3.1: Tập hợp F cùng với 2 phép toán cộng "+" và nhân "$*$" được gọi là một trường nếu thỏa mãn các tính chất sau:

1. F là một nhóm Abel với phép cộng "+"

2. F là đóng đối với phép nhân và tập $F \setminus \{0\}$ cùng với phép nhân "$*$" là một nhóm Abel.

3. Phép nhân "$*$" có tính chất phân phối đối với phép cộng "+", tức là $(a+b)*c = a*c + b*c$ và $a*(b+c) = a*b + a*c$

Trường F có q phần tử, được gọi là trường hữu hạn hay là trường Galois và được ký hiệu là $GF(q)$ hoặc F_q

Tập gồm 2 phần tử 0 và 1 cùng với 2 phép toán "$\otimes$" – phép nhân theo modulo 2 và phép toán "$\oplus$" – phép cộng theo modulo 2 được gọi là trường $GF(2)$

Định nghĩa 1.3.2: Đa thức $f(x) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_n$ với hệ số $a_i \in GF(2)$ được gọi là đa thức dưới trường $GF(2)$. Số nguyên lớn nhất n sao cho $a_n \neq 0$, được gọi là bậc của đa thức $f(x)$. Tập tất cả các đa thức dưới trường đã được tạo ra nào đó được ký hiệu là $F(x)$.

Định nghĩa 1.3.3: Cho $f(x)$ là đa thức bất khả quy (hoặc nguyên thủy) bậc n dưới trường $GF(2)$. Khi đó tập các đa thức theo modulo của đa thức $f(x)$ là trường hữu hạn với bậc 2^n.

Định nghĩa 1.3.4: Ánh xạ $f(x): GF(2n) \to GF(2)$ được gọi là hàm Boolean từ n biến. Tập tất cả các hàm Boolean n biến được ký hiệu là F_n.

Bảng chân trị của hàm Boolean $f(x)$ là vector nhị phân chứa đầu ra của hàm với mỗi giá trị có thể có của vector x.

Định nghĩa 1.3.5: Chuỗi các giá trị vô hướng của hàm Boolean f từ tập giá trị {0,1} có dạng $f(b_0)f(b_1)...f(b_{2n-1})$, trong đó $b_i \in GF(2n)$ được sắp xếp theo thứ tự, được gọi là bảng chân trị của hàm Boolean f.

Ví dụ 1: Ví dụ đối với hàm Boolean $f(x)$ với 3 biến: 10101100. Kết quả hiển thị trong bảng 1.1

Bảng 1.1: Bảng chân trị của hàm boolean 3 biến

x_1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1
$f(x_1, x_2, x_3)$	0	0	1	1	0	1	0	0

Đối với mỗi hàm Boolean f có thể được mô tả một cách duy nhất dưới dạng một đa thức với các hệ số từ trường $GF(2)$, bậc của đa thức này theo mỗi biến là không vượt quá 1. Đa thức này được gọi là đa thức Zyganskie hay dạng chuẩn đại số (Algebraic Normal Form – ANF).

$$f(x) = a_0 \sum_{i=1}^{n} a_i x_i \oplus \sum_{1 \le i < j \le n} a_{ij} x_i x_j \oplus ... \oplus a_{1...n} x_1 ... x_n 0 \quad (1.1)$$

trong đó tất cả 2^n hệ số $a_i, a_{ij}, ..., a_{1...n}$ đều thuộc trường $GF(2)$.

Định nghĩa 1.3.6: Hàm Boolean f được gọi là hàm affine nếu nó có dạng

$$f(x) = a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus ... \oplus a_n x_n \quad (1.2)$$

trong đó $a_i (i = 0...n) \in GF(2)$.

Trong trường hợp hàm tuyến tính thì $a_0 = 0$. Tập tất cả các hàm affine n biết được ký hiệu là A_n.

Có một cách để nhận được dạng ANF từ bảng chân trị đã biết của hàm Boolean. Nó được thực hiện bởi thuật toán sau đây:

Thuật toán tính đa thức Zhegalkin từ hàm Boolean được mô tả dưới dạng bảng chân trị:

Đầu vào: Vector giá trị của hàm boolean f.

Thuật toán: Vector chứa các hệ số của đa thức Zhegalkin tương ứng.

Bước 1: Sắp xếp bảng chân trị từ 2^n giá trị của hàm Boolean theo thứ tự tăng dần của các giá trị của biến đầu vào. Chúng ta ký hiệu chúng là P_f và mô tả dưới dạng cột:

$$P = \begin{bmatrix} f(b_0) \\ f(b_0) \\ ..... \\ f(b_m) \end{bmatrix} \qquad (1.3)$$

Trong đó $m = 2^n - 1;$

Bước 2: Tính vector cột v_f với 2^n các hệ số chưa biết của đa thức Zhegalkin cần tìm

$$v = \begin{bmatrix} v_0 \\ v_1 \\ ... \\ v_m \end{bmatrix} \qquad (1.4)$$

Trong đó $m = 2^n - 1;$

Bước 3: Xây dựng ma trận L_n theo cách đệ quy như sau:

$$L_0 = [1]; L_n = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes L_{n-1} = \begin{bmatrix} L_{n-1} & 0 \\ L_{n-1} & L_{n-1} \end{bmatrix}, \qquad (1.5)$$

Bước 4: Tính các hệ số của đa thức:

$$v = L_n P_n \qquad (1.6)$$

Ví dụ 2: Cho các hàm Boolean $f_1 f_2$ với các hệ số của bảng chân trị như sau:

$f_1 = [0,0,1,0,0,1,1,1,0,0,0,1,1,1,0,1]$

$f_2 = [0,1,1,0,1,1,1,0,1,0,0,1,0,0,1,0]$

$$P_{f1} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \qquad P_{f1} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$L_0 = [1] \qquad L_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad L_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$L_4 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}$$

Các hệ số và đa thức Zhegalkin tương ứng nhận được bởi thuật toán trên là:

$$v_1 = L_4 P_{f1} = [0,0,1,1,0,1,0,0,0,0,1,0,1,1,1,0]$$

$$p_1 = x_1 \oplus x_0x_1 \oplus x_0x_2 \oplus x_1x_3 \oplus x_2x_3 \oplus x_0x_2x_3 \oplus x_1x_2x_3$$

$$v_2 = L_4 P_{f2} = [0,1,1,0,1,1,1,1,1,0,0,0,0,0,1,0]$$

$$p_2 = x_0 \oplus x_1 \oplus x_2 \oplus x_0x_2 \oplus x_1x_2 \oplus x_0x_1x_2 \oplus x_3 \oplus x_1x_2x_3$$

Định nghĩa 1.3.7: Trọng số Hamming của vector $v \in GF(2^n)$ là số hệ số khác không, và được ký hiệu là $Wt_H(v)$.

Định nghĩa 1.3.8: Trọng số Hamming $Wt_H(f)$ của hàm Boolean f là số giá trị 1 trong bảng chân trị của hàm f.

Định nghĩa 1.3.9: Khoảng cách Hamming giữa 2 hàm Boolean f và g là $dist(f,g)$ được xác định bởi biểu thức sau:

$$dist(f,g) = Wt_H(f \oplus g) \qquad (1.7)$$

Ví dụ 3: Cho hàm Boolean f có bảng chân trị là: {01011001}. Hàm g : {10001001}. Khi đó, khoảng cách Hamming giữa f và g là:

$$dist(f,g) = Wt_H(f \oplus g) = Wt_H(11010000) = 3$$

Định nghĩa 1.3.10: Hai hàm $f(x_1, x_2, ..., x_n)$ và $g(x_1, x_2, ..., x_n) \in F_n$ được gọi là tương đương affine với nhau nếu tồn tại ma trận khả nghịch A sao cho:

$$g(Ax \oplus a) = \begin{cases} f(x) \\ f(x) \oplus 1 \end{cases} \qquad (1.8)$$

Để mô tả các thuộc tính của hệ thống mật mã, cũng như xác định độ an toàn của nó chống lại các phương pháp thám mã đã biết, các tính chất sau đây của hàm Boolean là rất quan trọng:

Tính cân bằng: Hàm Boolean f là cân bằng nếu $Wt_H(f) = 2^n - 1$, tức là số giá trị 1 và 0 trong bảng chân trị của nó bằng nhau.

Ví dụ 4: Hàm f 3 biến có bảng chân trị: {11000101}.

Trọng số Hamming $Wt_H(f) = 4 = 2^{3-1}$ là hàm cân bằng. Có 4 giá trị bằng 1 và 4 giá trị bằng 0.

Hàm g 3 biến có bảng chân trị: {11110101}.

Trọng số Hamming $Wt_H(g) = 6$. Đây là hàm không cân bằng. Có 6 giá trị bằng 1 và 2 giá trị bằng 0.

Độ phi tuyến: Khoảng cách Hamming nhỏ nhất của hàm f với tập tất cả các hàm affine A_n được gọi là độ phi tuyến của hàm f và được ký hiệu là N_f.

Ví dụ 5: Xác định độ phi tuyến của hàm f 2 biến

Cho hàm f với bảng chân trị là: {1011}. Ta sẽ viết tập các hàm affine A_n dưới dạng ANF:

$$\varphi_0 = 0, \quad \varphi_1 = 1, \quad \varphi_2 = x_1, \quad \varphi_3 = x_1 \oplus 1, \quad \varphi_4 = x_2,$$

$$\varphi_5 = x_2 \oplus 1, \quad \varphi_6 = x_2 \oplus 1, \quad \varphi_7 = x_2 \oplus x_1 \oplus 1$$

Mô tả các hàm affine này dưới dạng bảng chân trị trong bảng 1.2.

Bảng 1.2: Bảng chân trị

x_2	x_1	φ_0	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	φ_7	f
0	0	0	1	0	1	0	1	0	1	1
0	1	0	1	1	0	0	1	1	0	1
1	0	0	1	0	1	1	0	1	0	0
1	1	0	1	1	0	1	0	0	1	1

Tính khoảng cách Hamming giữa f với các hàm affine:

$dist(f,\varphi_0)=3,\quad dist(f,\varphi_1)=1,\quad dist(f,\varphi_2)=1,\quad dist(f,\varphi_3)=3$

$dist(f,\varphi_4)=3,\quad dist(f,\varphi_5)=1,\quad dist(f,\varphi_6)=3,\quad dist(f,\varphi_7)=1$

Như vậy độ phi tuyến của hàm f là: $N_f=\min\{dist(f,A_n)\}=1$

Định nghĩa 1.3.11: Hàm nhận giá trị số nguyên trên $GF(2^n)$ được gọi là phép biến đổi Hadamard và được xác định như sau:

$$W_{f(u)}=\sum(-1)^{f(x)\oplus<x,u>}\quad x\in GF(2^n)\qquad(1.9)$$

Đối với mỗi $u\in GF(2^n)$ giá trị $W_{f(u)}$ được gọi là hệ số Hadamard.

Có một cách khác (thuận tiện hơn cho việc lập trình) để định độc phi tuyến của hàm Boolean là tính toán thông qua hệ số Hadamard.

Độ phi tuyến của hàm Boolean f có thể được tính thông qua biến đổi Hadamard theo công thức sau đây:

$$N_f=2^{n-1}-\frac{1}{2}\max_{u\in GF(2^n)}\left|W_{f(u)}\right|\qquad(1.10)$$

Bậc đại số: Bậc đại số của hàm Boolean f là bậc của số hạng dài nhất trong hàm đó khi nó được biểu diễn dưới dạng ANF.

Ví dụ 6: Hàm $f(x)$ 3 biến: $f(x)=1\oplus x_1\oplus x_3\oplus x_1x_3\oplus x_1x_2x_3$. Số hạng có độ dài lớn nhất là: $x_1x_2x_3$, tức là bậc của nó bằng 3. Vậy bậc đại số của f là $deg(f)=3$.

Một tiêu chuẩn nữa cũng rất quan trọng là tiêu chuẩn phân phối (*PC - Propagation Criterion*) và hiệu ứng lan truyền thác chặt (*SAC – Strict Avalanche Criterion*). Nội dung chính của các tiêu chuẩn này là đánh giá xác suất thay đổi giá trị hàm Boolean phụ thuộc vào một phần xác định các đối số đầu vào.

Định nghĩa 1.3.12: Hàm Boolean f trên trường $GF(2^n)$ thỏa mãn:

- Tiêu chuẩn PC với vector α, $PC(\alpha)$ nếu hàm $f(x)\oplus f(x\oplus\alpha)$ là cân bằng, với $x\in GF(2^n)$, $x=(x_1,x_2,x_3,...,x_n)$.

- Tiêu chuẩn PC bậc k, $PC(k)$, nếu thỏa mãn tiêu chuẩn PC với mọi vector $\alpha\in GF(2^n)$ mà $1\le Wt_H(\alpha)\le k$.

- Tiêu chuẩn hiệu ứng lan truyền thác chặt SAC, nếu f thỏa mãn tiêu chuẩn PC bậc 1.

Nếu hàm f thỏa mãn $PC(k)$, điều này có nghĩa là thay đổi t bit đầu vào $1\le t\le k$ dẫn đến thay đổi mỗi bit đầu ra với xác suất bằng 1/2

Ví dụ 7: Cho $f:\{1\,0\,1\,1\}$. Ta sẽ xem xét f có thỏa mã $PC(1)$ hay không. Đối với các vector: $\alpha: wt(\alpha)=1$:

$\alpha=1:\sum f(x)\oplus f(x\oplus 1),\ x\in GF(2^2)$

$=\{f(0)\oplus f(0\oplus 1)\}+\{f(1)\oplus f(1\oplus 1)\}+\{f(2)\oplus f(2\oplus 1)\}+\{f(3)\oplus f(3\oplus 1)\}$

$=\{1\oplus 1\}+\{1\oplus 1\}+\{0\oplus 1\}+\{1\oplus 0\}=2$

$\alpha=2:\sum f(x)\oplus f(x\oplus 2),\ x\in GF(2^2)$

$=\{f(0)\oplus f(0\oplus 2)\}+\{f(1)\oplus f(1\oplus 2)\}+\{f(2)\oplus f(2\oplus 2)\}+\{f(3)\oplus f(3\oplus 2)\}$

$=\{1\oplus 0\}+\{1\oplus 1\}+\{0\oplus 1\}+\{1\oplus 1\}=2$

Như vậy f thỏa mãn $PC(1)$ và hệ quả là thỏa mãn tiêu chuẩn SAC. Kiểm tra với $PC(2)$ Với các vector $\alpha : wt(\alpha) = 2$

$\alpha = 3 : \sum f(x) \oplus f(x \oplus 3),\ x \in GF(2^2)$

$= \{f(0) \oplus f(0 \oplus 3)\} + \{f(1) \oplus f(1 \oplus 3)\} + \{f(2) \oplus f(2 \oplus 3)\} + \{f(3) \oplus f(3 \oplus 3)\}$

$= \{1 \oplus 1\} + \{1 \oplus 0\} + \{0 \oplus 1\} + \{1 \oplus 1\} = 2$

Vậy f thỏa mãn $PC(2)$

Trong các ứng dụng mật mã với hệ mật mã dòng, độ phi tuyến của hàm Boolean là rất quan trọng đối với việc tổng hợp của các thanh ghi dịch hồi quy tuyến tính khi xây dựng các bộ tạo dòng khóa. Các hàm Boolean này phải thỏa mãn các tính chất xác định, ví dụ: tính cân bằng, độ phi tuyến, bậc đại số cao, vv…, để tăng độ an toàn chống lại các kiểu thám mã. Các hàm Boolean cũng được sử dụng trong các hệ mật mã khối như là các hàm thành phần của hộp thế (S-box). Tiếp theo, chúng ta sẽ xem xét một số khái niệm và định nghĩa liên quan đến hộp thế (S-box).

1.4 Hộp thế và một số tính chất mật mã của hộp thế

1.4.1 Định nghĩa

Trong mã khối, bản rõ được chia thành các khối có kích thước cố định, các phép biến đổi của thuật toán sẽ thực hiện trên các khối đó, kết quả đầu ra là một khối bản mã. Nguyên tắc chính trong xây dựng hệ mật mã khối đã được đề xuất bởi Shannon trong công trình được công bố năm 1949 theo nguyên lý này, hệ mật mã khối là sự kết hợp của 2 khối hoạt động – khối xáo trộn (Confusion) và khối khuếch tán (Difusion). Trong đó, "xáo trộn" làm phức tạp hóa sự phụ thuộc của bản mã đối với bản rõ, gây cảm giác xáo trộn đối với kẻ thám mã có ý định phân tích tìm quy luật để phá mã. Quan hệ giữa mã – tin là phi tuyến. "Khuếch tán" làm khuếch tán những mẩu văn bản mang đặc tính thống kê (gây ra do dư thừa ngôn ngữ) lẫn vào toàn bộ văn bản. Nhờ đó gây khó khăn cho kẻ phá hoại trong việc dò mã trên cơ sở thống kê các mẫu lặp lại cao. Sự thay đổi 1 bit trong 1 khối bản rõ phải dẫn tới sự thay đổi hoàn toàn trong khối mã tạo ra. [4]

S-box (Substitution-box) là một thành phần căn bản của thuật toán mã hóa đối xứng, có chức năng thay thế. Trong hệ mã khối, S-box có chức năng che giấu mối quan hệ giữa bản mã và bản rõ nhằm tăng tính xáo trộn của thuật toán.

Về mặt tổng quát, S-box nhận đầu vào là 1 chuỗi m bit, sau đó nó sẽ biến đổi và tạo ra đầu ra là một chuỗi n bit (m và n không nhất thiết phải bằng nhau)

Hộp thế thực hiện thay thế khối đầu vào bằng khối đầu ra trong quá trình mã hóa, sao cho không thể tìm được sự phụ thuộc của đầu ra từ đầu vào bằng các phép toán tuyến tính nào đó. Vì vậy các hộp thế (S-box) có vai trò rất quan trọng đối với độ an toàn của các hệ mật và việc lựa chọn hộp thế là bước quan trọng trong việc xác định sức mạnh của hệ thống mật mã chống lại một số tấn công. Điều cần thiết là phải hiểu thiết kế và tính chất của S-box cho các ứng dụng mã hóa.Chất lượng được cải thiện của S-Box để tăng cường khả năng tạo sự xáo trộn trong mạng hoán vị thay thế (SPN) nhất định đã là một thách thức đối với các nhà nghiên cứu.

Độ an toàn của hầu hết các mật mã đối xứng hiện đại phụ thuộc rất nhiều vào các tính chất của các hộp thế được lựa chọn, các tính chất này như là phần tử cơ bản xác định độ an toàn của hệ mật chống lại các kiểu thám mã.

Có thể mô tả hộp thế dưới dạng một hệ hàm Boolean. Hộp thế $n \times m$ (n – đầu vào, m – đầu ra) được viết như là m hàm Boolean n biến. Nghĩa là các hộp thế được mô tả như các ánh xạ hàm Boolean.

Định nghĩa 1.4.1.1: Ánh xạ hàm Boolean (hoặc đơn giản là ánh xạ) $S: GF(2n) \to GF(2m),\ m > 0,\ n > 0$ được gọi là hộp thế kích thước $n \times m$. Thông thường, hộp thế được sử dụng là $n = m$, còn khi $m = 1$ ánh xạ này trở thành hàm Boolean. Như vậy, nếu xem rằng đầu ra của hộp thế là các hàm Boolean f_i n biến. thì $S(x) = (f_1(x), f_2(x), ..., f_m(x))$, với $x \in GF(2^n)$

Định nghĩa 1.4.1.2: Ánh xạ $F = (f_1, f_2, ..., f_m): GF(2^n) \to GF(2^m)$ được gọi là (n, m, t) – bất biến, nếu bất kỳ hàm nào nhận được từ F bằng cách cố định t bit đầu vào là hàm cân bằng.

Định nghĩa 1.4.1.3: Ánh xạ $S:GF(2n)\to GF(2m)$ được gọi là đồng nhất sai phân δ (*differential δ-uniform*), δ là số nguyên, nếu phương trình

$S(x)\oplus S(x\oplus a)=b\ \forall\ a\in GF(2^n), a\neq 0, b\in GF(2^m)$ có không quá δ nghiệm, nghĩa là:

$$\left|\{x\in GF(2n): S(x)\oplus S(x\oplus a)=b\}\right|\leq\delta$$

Các tính chất mật mã cơ bản của hộp thế cho biểu diễn độ an toàn của hệ mật bao gồm: tính cân bằng, độ phi tuyến, bậc đại số, điểm bất động, giá trị xấp xỉ vi sai cực đại, giá trị xấp xỉ tuyến tính cực đại.

1.4.2 Các tính chất mật mã của hộp thế

Trong các hệ mật mã khối, các hộp thế thực hiện việc biến đổi n đầu vào thành m bit đầu ra. Trong trường hợp:

Nếu $n=m$, hộp thế được gọi là hộp thế song ánh

Nếu $n>m$, hộp thế được gọi là hộp thế nén

Nếu $n<m$, hộp thế được gọi là hộp thế mở rộng.

Có thể mô tả hộp thế dưới dạng một hệ hàm Boolean. Hộp thế $n\times m$ với (n – *đầu vào*, m – *đầu ra*) được viết như là m hàm Boolean n biến. Nghĩa là các hộp thế được mô tả như các ánh xạ hàm Boolean.

Các tính chất mật mã cơ bản của hộp thế cho biểu diễn độ an toàn của hệ mật là:

1. Độ phi tuyến
2. Tính cân bằng
3. Bậc đại số
4. Điểm bất động
5. Giá trị xấp xỉ tuyến tính cực đại
6. Giá trị xấp xỉ vi sai cực đại

…

a. Độ phi tuyến

Tham số mật mã quan trọng nhất của một hộp thế là độ phi tuyến:

$$S(x) = (f_1(x), f_2(x), ..., f_m(x))$$

Độ phi tuyến nhỏ nhất của hàm Boolean trong tập gồm m hàm Boolean tạo ra hộp thế và tất cả các tổ hợp tuyến tính của m hàm Boolean đó:

$$N_s = min\{N_{f1}, N_{f2}, \dots, N_{fm}, N_{f1 \oplus f2}, \dots, N_{f1 \oplus f2 \dots \oplus fm}\}$$

Hay là:

$$N_s = \min\{N_{f1}, N_{f2}, ..., N_{fm}, N_{f1 \oplus f2}, ..., N_{f1 \oplus f2 \oplus \dots \oplus fm}\}$$

Trong đó $(c_1, c_2, ..., c_n) \in GF(2^n) \setminus (0,0,...,0)$.

Ví dụ 8: Xác định độ phi tuyến của hàm f với 2 biến:

Cho hàm f với bảng chân trị là: {1011}. Ta sẽ viết tập các hàm affine A_n dưới dạng ANF:

$\varphi_0 = 0$, $\varphi_1 = 1$, $\varphi_2 = x_0$, $\varphi_3 = x_0 \oplus 1$, $\varphi_4 = x_1$,

$\varphi_5 = x_1 \oplus 1$, $\varphi_6 = x_2 \oplus 1$, $\varphi_7 = x_2 \oplus x_1 \oplus 1$

Độ phi tuyến được thể hiện trong bảng 1.3

Bảng 1.3: Ví dụ độ phi tuyến của hàm f 2 biến

x_0	x_1	f_1	f_2	$f_1 \oplus f_2$
0	0	1	0	1
1	0	1	1	0
0	1	0	0	0
1	1	1	0	1

Như vậy độ phi tuyến của các hàm là:

$$N_{f1} = 1, N_{f2} = 1, N_{f1 \oplus f2} = 2$$

$$N_s = \min\{N_{f1}, N_{f2}, N_{f1 \oplus f2}\} = 1$$

Hay $N_s = \min\{N \mid f = c_1 f_1 \oplus c_2 f_2 \oplus c_3 f_3\}$

Để đảm bảo độ an toàn, hộp thế được tạo với độ phi tuyến càng lớn càng tốt để chống lại các tấn công thám mã vi sai và tấn công tuyến tính.

b. Tính cân bằng

Hệ hàm Boolean $S(x)$ được gọi là cân bằng nếu như các hàm Boolean f cân bằng, tức là trọng số Hamming

$$Wt_H(f_1) = Wt_H(f_2) = ... = Wt_H(f_m) = 2^n - 1$$

Hộp thế có tính chất cân bằng sẽ có khả năng chống lại tấn công thống kê, giúp cho thuật toán đảm bảo tính an toàn. [6]

c. Bậc đại số

Bậc đại số của hộp thế $deg(S)$ được định nghĩa là:

$$deg(S) = \min\{deg(f_1),...,deg(f_m),deg(f_1 \oplus f_2),...,deg(f_1 \oplus ... \oplus f_m)\}$$

Hay là:

$$deg(S) = \min\{deg(f) \mid f = c_1 f_1 \oplus c_2 f_2 \oplus ... \oplus c_m f_m\}$$

Trong đó $(c_1, c_2, ..., c_n) \in GF(2^n) \setminus (0,0,...,0)$.

Ví dụ 9: Cho hàm f với bảng chân trị là {1001}, ta có các hàm f 2 biến như trong bảng 1.4

Bảng 1.4: Tập các hàm f 2 biến

x_0	x_1	f_1	f_2	$f_1 \oplus f_2$
0	0	1	0	1
1	0	1	1	0
0	1	0	0	0
1	1	1	0	1

$$P_{f1}=\begin{bmatrix}1\\1\\0\\1\end{bmatrix},\quad P_{f2}=\begin{bmatrix}0\\1\\0\\0\end{bmatrix},\quad P_{f3}=P_{f1\oplus f2}=\begin{bmatrix}1\\0\\0\\1\end{bmatrix},\qquad L=\begin{bmatrix}1&0&0&0\\1&1&0&0\\1&0&1&0\\1&1&1&1\end{bmatrix}$$

$v_1 = LP_{f1} = [1, 0, 1, 1],\qquad P_1 = 1 \oplus x_1 \oplus x_0 x_1$

$deg(S) = \min\{deg(f_1)\} = 2$

Hay là $deg(S) = \min\{deg(f) \mid f = c_1 f_1\} = 2$

$v_2 = LP_{f2} = [0, 1, 0, 0],\quad P_2 = x_0 \oplus x_0 x_1$

$deg(S) = \min\{deg(f_2)\} = 2$

Hay là $deg(S) = \min\{deg(f) \mid f = c_2 f_2\} = 2$

$v_3 = LP_{f3} = [0, 1, 0, 0],\quad P_3 = 1 \oplus x_1$

$deg(S) = \min\{deg(f_3)\}$

$= \min\{deg(f_1), deg(f_2), deg(f_1 \oplus f_2)\} = 1$

Hay là $deg(S) = \min\{deg(f) \mid f = c_1 f_2 \oplus c_2 f_2\} = 1$

Bậc đại số của hộp thế càng cao càng tốt. Bậc đại số có ý nghĩa chống lại tấn công thám mã vi sai và tấn công đại số.

d. Điểm bất động

Điểm bất động của hộp thế s-box là một giá trị đầu vào mà sau khi đi qua hộp thế, đầu ra của nó vẫn giữ nguyên như đầu vào. Nói cách khác nếu $S(x)$ là hàm ánh xạ của hộp thế s-box thì x là một điểm bất động nếu: $S(x) = x$

Điểm bất động trong s-box có thể ảnh hưởng đến tính bảo mật của một hệ thống mã hóa. Các điểm bất động có thể làm giảm tính phi tuyến của S-box, dẫn đến sự dễ dàng hơn trong việc tấn công mã hóa bằng cách giảm độ phức tạp của hộp thế. Các thuật toán mã hóa mạnh thường loại bỏ các điểm bất động trong thiết kế s-box để tăng cường tính bảo mật.

e. Giá trị xấp xỉ tuyến tính

Giá trị xấp xỉ tuyến tính của hộp thế là phương pháp phân tích nhằm tìm các quan hệ tuyến tính xấp xỉ giữa đầu vào và đầu ra của S-box. Các S-box được thiết kế để đảm bảo phi tuyến tính nhằm bảo mật cao, do đó việc tìm được một xấp xỉ tuyến tính có thể hỗ trợ trong các tấn công dạng phân tích mật mã như tấn công tuyến tính.

Để tìm giá trị xấp xỉ tuyến tính cho S-box, người ta thường sử dụng bảng tuyến tính (linear approximation table – LAT). Cặp giá trị (α, β) tương ứng với xác xuất $Pr[\alpha.x = \beta.S(x)]$. Trong đó α và β là vector trọng số của các bit đầu vào và đầu ra, $S(x)$ là phép biến đổi của S-box

$$Pr[\alpha.x = \beta.S(x)] = \frac{1}{2} + e(\alpha, \beta),$$

Trong đó $e(\alpha, \beta)$ được gọi là độ lệch. Biến đổi công thức trên ta được một công thức khác:

$$2^{m+1} Pr[\alpha.x = \beta.S(x)] = 2^m + \hat{S}(\alpha, \beta).$$

Trong đó $\hat{S}(\alpha, \beta)$ được gọi là hệ số Fourier của S-box

f. Giá trị xấp xỉ vi sai

Giả sử $S(x)$ là một S-box với x là đầu vào nhị phân. Đối với hai giá trị đầu vào x và x', gọi $\Delta x = x \oplus x'$ là sự khác biệt đầu vào, và $\Delta y = S(x) \oplus S(x')$ là sự khác biệt đầu ra. Giá trị xấp xỉ vi sai được tính bằng cách xác định xác suất mà với một sự khác biệt đầu vào Δx, sự khác biệt đầu ra Δy xảy ra, tức là:

$$Pr(S(x) \oplus S(x') = \Delta y \mid x \oplus x' = \Delta x)$$

Giá trị xấp xỉ vi sai cực đại của S-box, ký hiệu là δ, được định nghĩa là xác suất lớn nhất của tất cả các cặp khác biệt đầu vào và đầu ra có thể:

$$\delta = \max_{\Delta x \neq 0, \Delta y} Pr(S(x) \oplus S(x') = \Delta y \mid x \oplus x' = \Delta x)$$

Trong đó, Δx và Δy lần lượt là chênh lệch đầu vào và đầu ra. Trong điều kiện lý tưởng, S-box cho thấy sự đồng nhất khác biệt. Xác suất xấp xỉ vi sai càng nhỏ thì S-box càng mạnh.

1.5 Tổng quan hệ mật AES và hộp thế trong thuật toán mã hóa AES

1.5.1 Tổng quan hệ mật AES

AES là một thuật toán mã hóa đối xứng, nghĩa là cùng một khóa được sử dụng để mã hóa và giải mã dữ liệu. Vì vậy, cả người gửi và người nhận phải biết và giữ bí mật khóa này. AES là một thuật toán mã hóa khối, mã hóa dữ liệu theo từng khối bit và hỗ trợ ba độ dài khóa khác nhau: AES-128 sử dụng khóa 128 bit (16 byte). AES-192 sử dụng khóa 192 bit (24 byte) và AES-256 sử dụng khóa 256 bit (32 byte). Khóa càng dài thì độ bảo mật càng cao, nhưng sẽ tốn nhiều tài nguyên tính toán hơn.

AES hoạt động trên một mảng dữ liệu 4x4 byte (được gọi là trạng thái) và sử dụng một quá trình mã hóa được chia thành các vòng lặp, với số vòng phụ thuộc vào độ dài khóa: AES-128: 10 vòng, AES-192: 12 vòng và AES-256: 14 vòng. Mỗi vòng gồm bốn bước chính:

1. SubBytes (Thay thế byte): Mỗi byte trong trạng thái được thay thế bằng một byte khác dựa trên bảng thay thế (S-box), nhằm tăng tính không thể dự đoán được.
2. ShiftRows (Dịch hàng): Các hàng của trạng thái được dịch vòng một số lượng nhất định các vị trí, tăng sự xáo trộn các byte.
3. MixColumns (Trộn cột): Các cột của trạng thái được kết hợp với nhau bằng cách sử dụng phép toán toán học, tạo ra sự trộn lẫn giữa các cột để làm phức tạp thêm cấu trúc dữ liệu.
4. AddRoundKey (Thêm khóa vòng): Mỗi byte của trạng thái được kết hợp với khóa vòng hiện tại bằng phép XOR. Khóa này là một phần của khóa chính, được tạo ra từ một thuật toán mở rộng khóa.

Quy trình mã hóa AES

Bước 1: Chuẩn bị khóa: Khóa chính được mở rộng thành nhiều khóa con (key schedule), mỗi khóa con được sử dụng trong mỗi vòng của thuật toán.

Bước 2: AddRoundKey ban đầu: Khối dữ liệu ban đầu được kết hợp với khóa chính bằng phép XOR.

Bước 3: Vòng mã hóa chính: Các vòng mã hóa được thực hiện, bao gồm SubBytes, ShiftRows, MixColumns và AddRoundKey.

Bước 4: Vòng cuối cùng: Vòng cuối cùng giống như các vòng trước, nhưng bỏ qua bước MixColumns.

Giải mã AES: giải mã AES là quá trình nghịch đảo của các bước mã hóa. Các bước tương tự được thực hiện theo thứ tự ngược lại với các phép toán nghịch đảo của SubBytes, ShiftRows và MixColumns, sử dụng khóa con ngược để khôi phục lại dữ liệu gốc.

1.5.2 Hộp thế trong thuật toán mã hóa AES

S-box có vai trò vô cùng quan trọng ảnh hưởng đến độ an toàn của thuật toán mật mã, vì vậy việc xây dựng các hộp thế là một trong những vấn đề quan trọng nhất trong lĩnh vực ứng dụng các phương pháp mật mã để bảo đảm an toàn thông tin.

a. Kiến trúc của Rijndael S-box

Rijndael là một mật mã khối, với kích thước khối thay đổi, độ dài khóa thay đổi và số vòng thay đổi. Độ dài khối và độ dài khóa có thể được chỉ định độc lập cho bất kỳ bội số nào của 32 bit từ 128 bit đến 256 bit. Mật mã Rijndael đã được chọn làm tiêu chuẩn mã hóa nâng cao (AES).

Độ dài khối và độ dài khóa có thể được chỉ định độc lập là 128, 192 hoặc 256 bit. Số vòng lặp phụ thuộc vào các giá trị tương ứng với độ dài khối đầu vào và độ dài khóa của mật mã, ví dụ: cho độ dài khối 128 bit, số vòng là 10. Một vòng Rijndael bao gồm hai lớp: một lớp phi tuyến tính (phép biến đổi SubByte) và một lớp tuyến tính. Mỗi byte trong khối đầu vào được thay thế theo từng byte bằng

phép biến đổi SubByte sử dụng hộp S thay thế. Hộp thế s-box tiêu chuẩn của hệ mật AES được thể hiện ở hình 1.3

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Hình 1.3: Hộp thế AES tiêu chuẩn

b. Hộp thế nghịch đảo trong AES

Nếu như S-box đóng vai trò quan trọng trong việc mã hoá AES thì S-box nghịch đảo cũng là một thành phần không thể thiếu trong pha giải mã của thuật toán AES.

S-box nghịch đảo của AES trong pha giải mã được tạo ra bằng cách áp dụng quy luật nghịch đảo lên S-box tiêu chuẩn của AES. Quá trình tạo S-box nghịch đảo này nhằm đảo ngược việc thay thế byte trong quá trình mã hóa.

Để tạo S-box nghịch đảo, ta áp dụng các bước sau:

Bước 1: Lấy giá trị của byte đầu vào trong quá trình giải mã.

Bước 2: Chia byte đầu vào thành hai phần: phần cao 4 bit (MSB) và phần thấp 4 bit (LSB).

Bước 3: Phần cao 4 bit của byte đầu vào sẽ xác định hàng trong S-box nghịch đảo, trong khi phần thấp 4 bit sẽ xác định cột.

Bước 4: Từ vị trí hàng và cột xác định được, ta truy xuất giá trị tương ứng từ S-box tiêu chuẩn của AES.

Bước 5: Giá trị được truy xuất từ S-box tiêu chuẩn sẽ được sử dụng để thay thế byte ban đầu.

Quá trình trên sẽ được lặp lại cho tất cả các byte trong khối dữ liệu đã được giải mã. Điều này đảm bảo rằng trong quá trình giải mã, các byte dữ liệu được đảo ngược từ quá trình thay thế trong quá trình mã hóa. S-box nghịch đảo của AES trong pha giải mã giúp khôi phục dữ liệu gốc từ dữ liệu đã được mã hóa.

Hộp thế nghịch đảo của thuật toán mã hóa AES được thể hiện tronh hình 1.4

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Hình 1.4: Hộp thế nghịch đảo tiêu chuẩn trong AES

CHƯƠNG 2 MỘT SỐ PHƯƠNG PHÁP TẠO HỘP THẾ

2.1 Một số phương pháp tạo hộp thế

Hiện nay có nhiều phương pháp khác nhau để tạo hộp thế. Phần này sẽ giới thiệu tóm lược một số phương pháp tạo hộp thế điển hình

2.1.1 Phương pháp ngẫu nhiên

Đây là một phương pháp đơn giản nhưng khá hiệu quả trong việc tạo ra hộp thế. Ý tưởng của phương pháp này là sắp xếp ngẫu nhiên các giá trị từ 0 đến 255 để tạo ra một mảng các byte có tính phi tuyến. Mỗi giá trị trong khoảng này chỉ xuất hiện một lần trong S-box, đảm bảo rằng mọi giá trị đầu vào đều có một ánh xạ duy nhất trong bảng.

Các bước thực hiện phương pháp ngẫu nhiên

Bước 1: Khởi tạo một mảng chứa các giá trị từ 0 đến 255. Mảng này sẽ chứa các phần tử là duy nhất, không trùng lặp

Bước 2: Sử dụng các thuật toán ngẫu nhiên (ví dụ, hàm xáo trộn của ngôn ngữ lập trình như random.shuffe trong python) để thay đổi thứ tự của các phần tử. Điều này đảm bảo các giá trị từ 0 đến 255 vẫn nằm trong mảng nhưng ở vị trí ngẫu nhiên

Bước 3: Ta sẽ đặt ra các chỉ số tối thiểu cho các tính chất của hộp thế chả hạn như độ phi tuyến tối thiểu của s-box tạo ra phải lớn hơn 90, bậc đại số phải lớn hơn 5, không có điểm bất động nào …

Bước 4: Kiểm tra các chỉ số của các tính chất mật mã của bước 3, nếu hộp thế đạt tiêu chuẩn thì ta sẽ đi xây dựng hộp thế nghịch đảo và đưa vào sử dụng. Nếu hộp thế không đạt tiêu chuẩn thì ta sẽ quay lại bước 2, tiếp tục tạo ra hộp thế ngẫu nhiên

Ưu điểm của cách làm này khá đơn giản, không theo quy luật tuyến tính nào nhưng nhược điểm lại khá lớn, khá khó để tạo ra hộp thế lý tưởng thỏa mãn nhiều tiêu chí

2.1.2 Sử dụng hàm băm (hash function)

S-box có thể được tạo bằng cách sử dụng hàm băm để ánh xạ các khối dữ liệu đầu vào thành các khối dữ liệu đầu ra. Các hàm băm như SHA-3, SHA-2 hoặc MD5 có thể được sử dụng để tạo S-box.

Giả sử chúng ta muốn tạo một S-box với kích thước đầu vào là 8-bit và kích thước đầu ra cũng là 8-bit, chúng ta có thể sử dụng hàm băm SHA-256 để tạo S-box.

Bước 1: Chọn một chuỗi ngẫu nhiên để làm khóa cho hàm băm SHA-256, ví dụ như chuỗi kí tự "SecretKey".

Bước 2: Tạo ra một bảng 256x8-bit với giá trị ban đầu là các giá trị từ 0 đến 255.

Bước 3: Áp dụng hàm băm SHA-256 lên các giá trị từ 0 đến 255 để tạo ra các giá trị 256-bit tương ứng.

Bước 4: Chia các giá trị 256-bit tương ứng thành 8 phần bằng nhau, mỗi phần có độ dài 32-bit.

Bước 5: Ánh xạ mỗi phần 32-bit thành một giá trị từ 0 đến 255 bằng cách chuyển đổi giá trị thập phân từ phần 32-bit đó sang hệ cơ số 256.

Bước 6: Thay thế các giá trị trong bảng 256x8-bit ban đầu bằng các giá trị 8-bit tương ứng thu được từ bước 5.

Kết quả là một S-box 8-bit được tạo ra bằng cách sử dụng hàm băm SHA-256. Trong quá trình mã hoá, các giá trị đầu vào 8-bit sẽ được ánh xạ sang các giá trị tương ứng 8-bit trong S-box, và giá trị đầu ra sẽ được tính toán từ các giá trị tương ứng này. S-box được tạo ra bằng phương pháp này có độ khó bị tấn công cao, do tính ngẫu nhiên của hàm băm SHA-256.

Ví dụ **:**

Chọn một hàm băm an toàn như SHA-256 hoặc SHA-3 để tạo ra một giá trị băm cho mỗi khóa đầu vào. Ví dụ, giả sử chúng ta có khóa đầu vào là "$ABCDEF123456$".

Sử dụng giá trị băm đó để tạo ra một chuỗi các giá trị ngẫu nhiên. Ví dụ, giả sử giá trị băm đó là:

"0x7c4611f0d2657b4e4f03354efc4a4d3d4f3944ebe20c545b559cd9d18d9e70ca"

Chia chuỗi các giá trị ngẫu nhiên đó thành các khối có độ dài bằng nhau (ví dụ, 8 hoặc 16 bytes). Ví dụ, giả sử chúng ta chia chuỗi đó thành các khối 8 bytes.

Áp dụng một hàm biến đổi không tuyến tính đơn giản lên mỗi khối để tạo ra các giá trị S-box. Ví dụ, giả sử chúng ta sử dụng hàm biến đổi như sau:

$$S(x) = (5 * x_3 + 7 * x_2 + 11 * x + 13) \, mod \, 256$$

Trong đó "mod" biểu thị phép chia lấy dư.

Sử dụng các giá trị S-box này để thay thế các giá trị ban đầu trong S-box của bạn. Ví dụ, giả sử chúng ta sử dụng các khối 8 bytes của giá trị ngẫu nhiên được tạo ra bằng giá trị băm "$ABCDEF123456$" để tạo ra các giá trị S-box như sau:

$$S(0x7c4611f0d2657b4e) = 0x39$$
$$S(0x4f03354efc4a4d3d) = 0xa4$$
$$S(0x4f3944ebe20c545b) = 0x1d$$
$$S(0x559cd9d18d9e70ca) = 0x2a$$

Ta có thể sử dụng các giá trị S-box này để thay thế các giá trị ban đầu trong S-box

2.1.3 Sử dụng mã hóa ngược (inverted encryption):

S-box có thể được tạo bằng cách sử dụng một thuật toán mã hóa ngược, trong đó khối dữ liệu đầu vào được mã hoá ngược và sau đó được giải mã để tạo ra khối dữ liệu đầu ra tương ứng.

Ví dụ : Giả sử chúng ta muốn tạo một S-box động có kích thước 16x16 cho mã hóa AES. Để tạo ra S-box này, chúng ta có thể sử dụng inverted encryption như sau:

Bước 1: Chọn một khóa bí mật ngẫu nhiên kích thước 128 bit.

Bước 2: Tạo một mảng 16x16 ngẫu nhiên, đó là S-box ban đầu.

Bước 3: Thực hiện mã hoá AES với khóa bí mật và S-box ban đầu để tạo ra một S-box mới.

Bước 4: Sử dụng inverted encryption để giải mã S-box mới, sử dụng cùng khóa bí mật.

Bước 5: So sánh S-box ban đầu với S-box giải mã được, nếu chúng khác nhau, quay lại bước 2 và lặp lại quá trình cho đến khi tìm được S-box thích hợp.

2.1.4 Sử dụng biến đổi tuyến tính (linear transformation)

S-box có thể được tạo bằng cách sử dụng đa tuyến tính, trong đó một số lượng lớn các phép biến đổi tuyến tính được áp dụng lên khối dữ liệu đầu vào để tạo ra khối dữ liệu đầu ra.

Ví dụ về S-box được tạo bằng đa tuyến tính có thể như sau:

Bước 1: Khối dữ liệu đầu vào được chia thành các khối con, mỗi khối có kích thước bằng với kích thước của S-box.

Bước 2: Mỗi khối con được biểu diễn dưới dạng vector cột.

Bước 3: Một ma trận A có kích thước bằng với kích thước của S-box được tạo ra.

Bước 4: Mỗi vector cột của khối dữ liệu đầu vào được nhân với ma trận A để tạo ra vector cột của khối dữ liệu đầu ra tương ứng.

Bước 5: Các vector cột của khối dữ liệu đầu ra được ghép lại để tạo thành khối dữ liệu đầu ra hoàn chỉnh.

Việc sử dụng đa tuyến tính để tạo S-box mang lại nhiều ưu điểm, như tính đa dạng trong khối dữ liệu đầu ra, khả năng kháng lại các cuộc tấn công phổ biến như tấn công tuyến tính và tấn công phi tuyến, đồng thời cũng đơn giản và hiệu quả trong việc thực hiện các phép biến đổi tuyến tính.

2.1.5 Phương pháp nghịch đảo trên trường hữu hạn GF(2^8)

Đây chính là kỹ thuật cốt lõi được sử dụng trong thuật toán mã hóa AES. Phương pháp này tạo ra s-box có độ phi tuyến cao, giúp đảm bảo khả năng chống

lại các cuộc tấn công tuyến tính và vi phân. Cụ thể quá trình tạo s-box trong AES dựa trên 2 bước chính:

1. Tính phép nghịch đảo trong trường $GF(2^8)$

Trường hữu hạn $GF(2^8)$ là trường nhị phân gồm 256 phần tử, đại diện cho các giá trị từ 0 đến 255 (các byte 8 bit). Trong phương pháp này, mỗi byte đầu vào x trong khoảng từ 0 đến 255 sẽ được thay thế bằng phép nghịch đảo cộng dồn của nó trong trường $GF(2^8)$

Giá trị 0 không có nghịch đảo thì được ánh xạ với 0. Các giá trị khác sẽ được tính sao cho $x.x^{-1} = 1$

2. Biến đổi Affine

Sau khi tính nghịch đảo, AES tiếp tục thực hiện phép biến đổi affine để tăng cường tính phi tuyến và che khuất cấu trúc gốc của dữ liệu. Phép biến đổi affine này là một phép toán tuyến tính, nhưng khi áp dụng với nghịch đảo trên $GF(2^8)$, nó tạo ra tính phi tuyến mạnh hơn. Công thức của phép biến đổi như sau:

$$y = Ax^{-1} + b$$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_0 \\ x_0 \\ x_0 \\ x_0 \\ x_0 \\ x_0 \\ x_0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Đầu vào của phép biến đổi là x và đầu ra là y, với mỗi giá trị $x(0 \le x \le 255)$ thì ta sẽ nhận được một giá trị y tương ứng

Tiếp sau đây sẽ đi sâu vào nghiên cứu phương pháp tạo S-box động dựa trên ánh xạ hỗn loạn rời rạc một chiều pure.

2.2 Thiết kế S-box với ánh xạ hỗn loạn một chiều pure

2.2.1 Cơ sở lý thuyết của ánh xạ hỗn loạn rời rạc một chiều pure

Ánh xạ hỗn loạn rời rạc một chiều pure là một loại ánh xạ rời rạc một chiều đặc biệt, có tính chất ngẫu nhiên, không thể dự đoán và thường được sử dụng trong mã hóa thông tin, bảo mật và các ứng dụng khác. [5]

Công thức tổng quát :

$$x_{n+1} = f(x_n)$$

Trong đó, x_n là giá trị hiện tại của ánh xạ, x_{n+1} là giá trị kế tiếp, và hàm số $f(x)$ là hàm ánh xạ hỗn loạn rời rạc một chiều pure. Hàm số này thường được chọn để đảm bảo tính ngẫu nhiên và khó dự đoán của giá trị tiếp theo.

Ví dụ : Giả sử chúng ta muốn tạo một S-box với kích thước đầu vào và đầu ra đều là 8-bit, và sử dụng phương pháp ánh xạ hỗn loạn rời rạc một chiều pure để tạo ra S-box.

Bước 1: Chọn một giá trị seed (hạt) ngẫu nhiên để khởi tạo hệ thống dynamic chaotic.

Bước 2: Sử dụng công thức để tạo ra một chuỗi số ngẫu nhiên dài bằng với số lượng giá trị trong S-box:

$$x[i] = (a * x[i-1] + b) \% c$$

với $x[0] = seed$; a, b, c là các giá trị số nguyên tố được chọn ngẫu nhiên.

Bước 3: Sắp xếp các giá trị trong chuỗi số ngẫu nhiên theo thứ tự tăng dần, và ánh xạ các giá trị này vào S-box.

Bước 4: Thực hiện các phép biến đổi ngược lại trên S-box để tăng tính ngẫu nhiên của S-box.

Kết quả là một S-box 8-bit được tạo ra bằng cách sử dụng phương pháp ánh xạ hỗn loạn rời rạc một chiều pure. Trong quá trình mã hoá, các giá trị đầu vào 8 bit sẽ được ánh xạ sang các giá trị tương ứng 8-bit trong S-box, và giá trị đầu ra sẽ được tính toán từ các giá trị tương ứng này. S-box được tạo ra bằng phương pháp

này có độ khó bị tấn công tương đối cao, do tính ngẫu nhiên của chuỗi số ngẫu nhiên được tạo ra bằng công thức ánh xạ hỗn loạn.

2.2.2 Một số loại ánh xạ hỗn loạn rời rạc một chiều pure

Có nhiều loại ánh xạ hỗn loạn rời rạc một chiều pure được sử dụng trong các ứng dụng bảo mật và mã hóa thông tin, sau đây là một số ví dụ:

a. Ánh xạ Logistic

Ánh xạ Logistic là một ví dụ của ánh xạ rời rạc một chiều pure, nó được sử dụng rộng rãi trong các ứng dụng về động học học địa chất, sinh học, kinh tế, v.v. Công thức của nó được biểu diễn như sau:

$$x_{n+1} = r * x_n * (1 - x_n)$$

Trong đó x_n là giá trị của biến x tại thời điểm n; x_{n+1} là giá trị của biến x tại thời điểm $n+1$ và r là hằng số được gọi là tham số r. Tham số r có giá trị từ 0 đến 4 và ảnh hưởng đến hình dạng của ánh xạ. Trong ánh xạ logistic, giá trị của tham số r ảnh hưởng trực tiếp đến tính chất của hệ thống. Khi r thay đổi, hệ thống có thể chuyển từ trạng thái tương đối ổn định và dự đoán được sang trạng thái hỗn loạn và không thể dự đoán được.

Tuy nhiên, nếu r quá nhỏ, hệ thống sẽ có xu hướng ổn định và dễ dự đoán, còn nếu r lớn, hệ thống sẽ tăng sự hỗn loạn và không thể dự đoán được.

Nghiên cứu cho thấy, khoảng giá trị của r từ 3.569 đến 4 được coi là đủ để tạo ra tính chất hỗn loạn mà vẫn giữ được tính dễ dàng phân tích và tính đơn giản của hệ thống. Do đó, khi sử dụng ánh xạ logistic để tạo S-box động, ta nên chọn giá trị của r trong khoảng này để đảm bảo tính an toàn và hiệu quả của hệ thống.

Ví dụ : nếu chúng ta muốn tính toán 100 giá trị tiếp theo của x từ giá trị ban đầu là $x_0 = 0.1$ và tham số $r = 3.8$, chúng ta sẽ áp dụng công thức ánh xạ Logistic như sau:

$$x_1 = r * x_0 * (1 - x_0) = 3.8 * 0.1 * (1 - 0.1) = 0.342$$

$$x_2 = r * x_1 * (1 - x_1) = 3.8 * 0.342 * (1 - 0.342) = 0.821$$

$x_3 = r * x_2 * (1 - x_2) = 3.8 * 0.821 * (1 - 0.821) = 0.499$

$x_4 = r * x_3 * (1 - x_3) = 3.8 * 0.499 * (1 - 0.499) = 0.949$

v.v.

Các giá trị tiếp theo của x có thể tính được bằng cách lặp lại công thức trên với giá trị mới tính được x_{n+1} được sử dụng để tính toán giá trị tiếp theo.

b. Ánh xạ Tent

Ánh xạ Tent là một ánh xạ rời rạc một chiều pure, được đặt tên là "Tent" bởi vì đồ thị của nó trông giống như một cái lều (từ "tent" trong tiếng Anh). Ánh xạ Tent được xác định trên khoảng $[0,1]$ bởi công thức sau:

$$x_{n+1} = \begin{cases} r * x_n & (0 \le x_n < 0.5) \\ r * x_{n-1} & (0.5 \le x_n \le 1) \end{cases}$$

Trong đó: x_n là giá trị đầu vào của ánh xạ tại thời điểm n, và x_{n+1} là giá trị đầu ra của ánh xạ tại thời điểm $n+1$ và r là tham số ngẫu nhiên.

Khi thay đổi giá trị của r, chuỗi số được tạo ra có thể có tính chất khác nhau như chu kỳ, hoặc tính ngẫu nhiên. Ví dụ, khi $r = 1.8$, chuỗi số sẽ có tính chất hội tụ đến giá trị xấp xỉ 0.66, trong khi khi $r = 3.5$, chuỗi số sẽ có tính chất hỗn loạn và không có chu kỳ rõ ràng. Việc chọn giá trị của r phụ thuộc vào mục đích sử dụng của bạn. Ánh xạ Tent có hai điểm bất động: $x = 0$ và $x = 1/2$. Nó cũng thỏa mãn các tính chất của ánh xạ hỗn loạn rời rạc một chiều pure như tính toán ngẫu nhiên và nhạy cảm đến điều kiện ban đầu. Ánh xạ Tent cũng được sử dụng trong các ứng dụng mã hóa để tạo ra các số học ngẫu nhiên.

Ví dụ : chúng ta có thể sử dụng Ánh xạ Tent để tạo ra một chuỗi các số học ngẫu nhiên. Ta bắt đầu với một giá trị khởi đầu x_0 nằm trong khoảng $[0,1]$ và áp dụng công thức ánh xạ Tent để tạo ra giá trị tiếp theo x_1. Sau đó, ta tiếp tục áp dụng công thức này lặp đi lặp lại để tạo ra các giá trị tiếp theo x_2, x_3,.. nếu chúng ta chọn $x_0 = 0.1$ và $r = 2$, ta có thể tính các giá trị của chuỗi x_n từ x_0 đến x_6 như sau:

$x_0 = 0.1$ (giá trị đầu vào ban đầu)

$x_1 = 2 * x_0 = 0.2$

$x_2 = 2 * x_1 = 2 * 0.2 = 0.4$

$x_3 = 2 * x_2 = 2 * 0.4 = 0.8$

$x_4 = 2 * (1 - x_3) = 2 * (1 - 0.8) = 0.4$

$x_5 = 2 * x_4 = 2 * 0.4 = 0.8$

$x_6 = 2 * (1 - x_5) = 2 * (1 - 0.8) = 0.4$

……

c. Ánh xạ Sine

Ánh xạ Sine là một loại ánh xạ rời rạc một chiều pure được định nghĩa bằng công thức:

$$x_{n+1} = \alpha * \sin(x_n)$$

Trong đó, α là một tham số và x_n là giá trị của ánh xạ tại thời điểm n. Ánh xạ sine được gọi là ánh xạ sine do công thức của nó chứa hàm sin.

Ánh xạ sine có tính chất hỗn loạn và có thể hiển thị các hình dạng phức tạp, bao gồm các hình dạng hỗn loạn và không định hướng. Nó cũng có thể hiển thị các chu kỳ bất thường và các vùng không ổn định. Ánh xạ sine đã được sử dụng trong nghiên cứu về động lực học và các ứng dụng liên quan như mã hóa ảnh và nén dữ liệu.

Ví dụ : khi $\alpha = 1.4$ và giá trị ban đầu $x_0 = 0.2$. Ánh xạ này có thể được tính bằng cách áp dụng công thức:

$$x_{n+1} = \alpha * \sin(x_n)$$

Với giá trị ban đầu $x_0 = 0.2$, ta có:

$x_1 = \alpha * \sin(x_0) = 1.4 * \sin(0.2) = 0.3917$

$x_2 = \alpha * \sin(x_1) = 1.4 * \sin(0.3917) = 0.7954$

$x_3 = \alpha * \sin(x_2) = 1.4 * \sin(0.7954) = 0.9902$

$x_4 = \alpha * \sin(x_3) = 1.4 * \sin(0.9902) = 0.8349$

$x_5 = \alpha * \sin(x_4) = 1.4 * \sin(0.8349) = 0.9623$

…….

Kết quả cho thấy rằng giá trị của ánh xạ tăng và giảm theo một cách không định hướng, tạo ra một hình dạng phức tạp và hỗn loạn. Các giá trị này thường được biểu diễn trên đồ thị Lyapunov, một công cụ thường được sử dụng để phân tích các loại hình dạng và tính chất của các ánh xạ rời rạc.

2.2.3 Thuật toán tạo S-box động dựa trên ánh xạ hỗn loạn rời rạc một chiều pure

Dưới đây sẽ thực hiện nghiên cứu phương pháp tạo S-box động dựa trên hàm ánh xạ Logistic được giới thiệu qua ở mục 2.2.2, phương pháp này được công bố trong tài liệu số [8].

Qúa trình thực hiện tính toán giá trị của S-box có thứ tự sau:

Bước 1: Chọn một điểm khởi đầu x_0 với giá trị nằm trong khoảng (0,1), một tham số số học r với giá trị nằm trong khoảng (0,4) để sử dụng trong ánh xạ logistic.

Bước 2: Đặt $i = 0$, khởi tạo mảng X rỗng, biến $h = null$, mảng $S = null$

Bước 3: Sử dụng vòng lặp while và lặp cho đến khi nào độ dài mảng $S = 256$

Bước 3.1: Xây dựng hàm logistic: $X[i+1] = r * X[i] * (1 - X[i])$

Bước 3.2: Chuyển đổi giá trị $X[i]$ sang kiểu dữ liệu hexa và lấy 2 giá trị từ vị trí thứ 7 đến 9 và lưu vào biến h

Bước 3.3: Kiểm tra nếu h nằm trong S thì tăng i $(i++)$, ngược lại thì lưu giá trị h vào mảng S và vẫn tăng i

Bước 4: Chuyển đổi mảng S thành ma trận 16x16 và trả ra kết quả

Sau khi đã xây dựng hàm tạo S-box, ta sẽ sử dụng s-box tạo ra làm giá trị khởi tạo để tính hộp thế nghịch đảo. Các bước để tạo hộp thế nghịch đảo như sau:

Bước 1: Nhận hộp thế s-box vừa tạo ra làm tham số đầu vào

Bước 2: Khởi tạo biến $a = b = 0$. Khởi tạo ma trận s-box nghịch đảo inv_sbox có độ dài 16x16

Bước 3: Dùng vòng lặp for $i = 0$ tới 15

Bước 3.1: Dùng vòng lặp for $j = 0$ tới 15

Bước 3.2: Gán a và b bằng ký tự đầu và thứ 2 của ma trận $S[i, j]$ và chuyển a và b sang số nguyên

Bước 3.3: chuyển đổi i và j sang kiểu hex, nối i và j thành chuỗi và gán cho inv_sbox

CHƯƠNG 3 XÂY DỰNG CHƯƠNG TRÌNH THỰC THI

3.1 Yêu cầu đối với chương trình thực thi

3.1.1 Công cụ và môi trường thực thi

Khi xây dựng chương trình thuật toán cần lựa chọn ngôn ngữ và công cụ lập trình phù hợp. Trong phạm vi đề tài đồ án, em đã sử dụng ngôn ngữ Python, công cụ Visual Studio Code và một số công cụ phân tích như Sage Math và Matlab để xây dựng chương trình thực thi.

Yêu cầu về các extention: Python, Python Extension Pack

Yêu cầu về thư viện:

Numpy: pip install numpy. Được sử dụng cho tính toán khoa học, chủ yếu tập trung vào việc xử lý các mảng (arrays) và ma trận (matrix)

PyCryptodome: pip install pycryptodome. Cần cài đặt trong môi trường ảo .venv. Thư viện này được sử dụng để mã hóa mạnh mẽ, cung cấp các công cụ để mã hóa và giải mã và thao tác với các thuật toán mật mã khác nhau

PIL: viết tắt của Python Imaging Library, là một thư viện mạnh mẽ cho việc xử lý hình ảnh trong Python. Nó cung cấp các chức năng để mở, thao tác và lưu trữ hình ảnh trong nhiều định dạng khác nhau.

Os: là một thư viện trong python cung cấp các hàm, sử dụng để tương tác với hệ điều hành

Ngoài ra còn sử dụng một số thư viện phụ hỗ trợ khác

3.1.2 Yêu cầu đối với chương trình

Xây dựng chương trình tạo ra s-box động dựa trên ánh xạ hỗn loạn 1 chiều, cụ thể là sử dụng ánh xạ logistic

Chương trình xây dựng dựa trên thuật toán đã được đề xuất ở mục 2.2.3, để có thể đưa ra được giá trị của hộp thế s-box (sử dụng trong quá trình mã hóa) và s-box nghịch đảo(dùng trong quá trình giải mã)

Sau khi tạo được s-box, áp dụng s-box tạo được vào thuật toán AES để mã hóa và giải mã dữ liệu, hình ảnh. Đánh giá một số tính chất của hộp thế và hình ảnh mã hóa. So sánh hình ảnh mã hóa bằng thuật toán mã hóa AES với hộp thế mặc định và hộp thế được tạo ra

3.2 Xây dựng chương trình và thử nghiệm

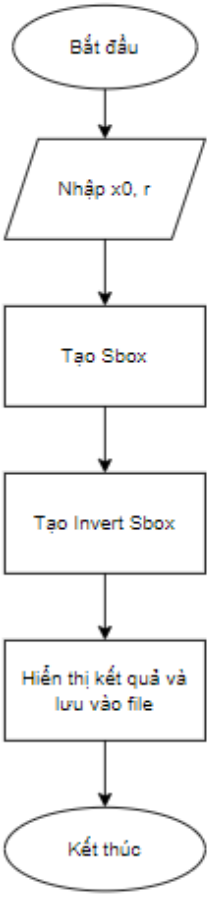
3.2.1 Xây dựng lược đồ tổng quan chương trình

a. Lược đồ tổng quan xây dựng hộp thế s-box

Chương trình xây dựng hộp thế s-box được xây dựng dựa trên các bước sau:

1. Xây dựng thuật toán hàm ánh xạ
2. Xây dựng hộp thế s-box và s-box nghịch đảo
3. Hiển thị kết quả và ghi vào file

Lược đồ tổng quan xây dựng s-box được thể hiện trong hình 3.1



Hình 3.1: Lược đồ tổng quan xây dựng hộp thế s-box

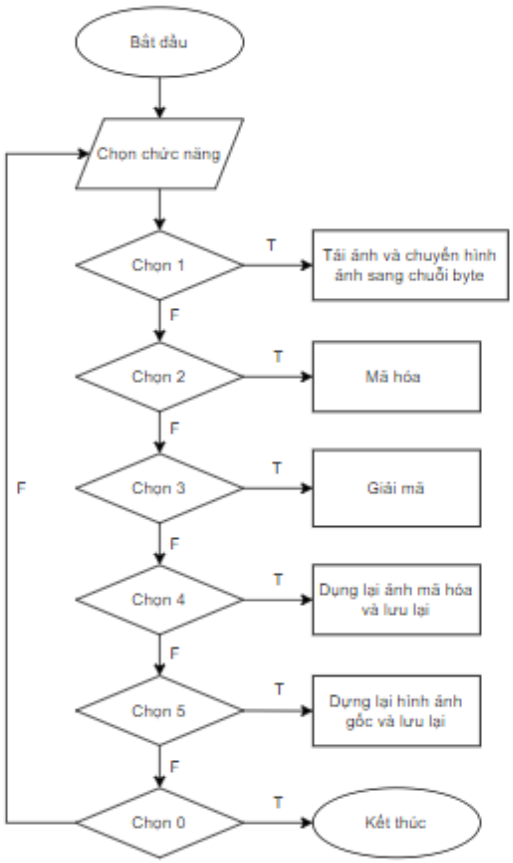
b. Lược đồ tổng quan mã hóa và giải mã hình ảnh

Sau khi ta tạo thành công hộp thế s-box, ta cần xây dựng thư viện AES của riêng mình để có thể thay đổi s-box mặc định bằng s-box tạo ra. Một số hàm tạo thư viện AES sẽ trình bày chi tiết ở Phụ lục 1

Chương trình mã hóa và giải mã hình ảnh được xây dựng dựa trên các bước sau:

1. Tải hình ảnh, chuyển đổi hình ảnh thành các chuỗi byte
2. Mã hóa chuỗi byte
3. Giải mã chuỗi byte
4. Dựng lại hình ảnh mã hóa từ chuỗi byte mã hóa, lưu lại hình ảnh
5. Dựng lại hình ảnh ban đầu bằng chuỗi byte giải mã, lưu lại hình ảnh

Lược đồ tổng quan xây dựng chương trình mã hóa, giải mã hình ảnh được thể hiện trong hình 3.2



Hình 3.2: Lược đồ tổng quan chương trình mã hóa, giải mã hình ảnh

3.2.2 Xây dựng chương trình

a. Xây dựng chương trình tạo s-box

- Import thư viện numpy

```
import numpy as np
```

Thư viện numpy sử dụng để xử lý mảng và ma trận

- Dựa theo thuật toán tạo s-box đã nêu ở mục 2.2.3, ta tiến hành xây dựng thuật toán tạo hộp thế s-box

```
def create_sbox(x0, r):
    arrS = []
    i = 0
    arrX = [x0]
    while len(arrS) < 256:
        before = arrX[i]
        value = r * before * (1 - before)
        arrX.append(value)
        h = before.hex()[8:10]
        if h in arrS:
            i += 1
        else:
            arrS.append(h)  # Lưu giá trị hex vào mảng
            i += 1
    sbox = np.array(arrS).reshape(16, 16)
    sbox_tuple = tuple(int(value, 16) for row in sbox for value in row)
    sbox_format = '\n'.join([' '.join(row) for row in sbox])
    return sbox, sbox_tuple, sbox_format
```

Xây dựng hàm tạo hộp thế và truyền vào 2 tham số: x0 và r

Tạo mảng S rỗng, gán i = 0, khởi tạo mảng X là mảng trung gian có 1 phần tử là giá trị khởi tạo

Cho vòng lặp chạy cho đến khi nào mảng S đủ 256 phần tử thì dừng lại. Trong vòng lặp, với mỗi lần lặp ta sẽ gán lại giá trị trước đó là phần tử cuối cùng của mảng trung gian X, sử dụng hàm logistic để tính giá trị tiếp theo. Khởi tạo biến h để lưu trữ giá trị của s-box, h sẽ có kiểu dữ liệu hexa và có giá trị là 2 ký tự từ 7 đến 9 của giá trị trước đó. Ta tiến hành kiểm tra nếu như h đã tồn tại trong mảng S thì ta sẽ bỏ qua, nếu h chưa tồn tại trong mảng S thì ta sẽ thêm h vào cuối mảng. Bước này được sử dụng để tránh các giá trị trùng lặp trong mảng

Tạo biến sbox, sbox_tuple, sbox_fomat để lưu trữ s-box dưới dạng ma trận, dạng kiểu dữ liệu tuple (sử dụng để thay thế S-box mặc định của AES) và dạng ma trận để hiển thị

- Sau khi tạo xong hộp thế, ta tiếp tục tạo hộp thế nghịch đảo

```
def create_invsbox(sbox):
    inv_sbox = np.zeros((16, 16), dtype='U2')
    for i in range(16):
        for j in range(16):
            a, b = sbox[i, j]
            a = int(a, 16)
            b = int(b, 16)
            i_hex = hex(i)[2:]
            j_hex = hex(j)[2:]
            inv_sbox[a, b] = '{}{}'.format(i_hex, j_hex)
    inv_sbox_tuple = tuple(int(value, 16) for row in inv_sbox for value in row)
    inv_sbox_format = '\n'.join([' '.join(row) for row in inv_sbox])
    return inv_sbox_tuple, inv_sbox_format
```

Xây dựng hàm tạo hộp thế nghịch đảo, nhận s-box tạo ra làm tham số đầu vào

Khởi tạo mảng inv_sbox dưới dạng ma trận 16x16, mỗi phần tử gồm có 2 ký tự là các giá trị 0

Sử dụng vòng lặp lồng nhau, trong vòng lặp ta sẽ lấy từng ký tự của mỗi phần tử s-box và gán vào a và b. Ta chuyển đổi biến a và b thành số thập phân để làm chỉ mục hàng và cột của hộp thế nghịch đảo. Với mỗi chỉ số i và j của vòng lặp, ta sẽ sẽ chuyển đổi sang kiểu dữ liệu hexa và chuyển thành chuỗi và gán vào từng phần tử của hộp thế nghịch đảo

Tạo biến inv_sbox_tuple lưu giá trị của inv_sbox dưới kiểu dữ liệu tuple để sử dụng thay thế cho hộp thế nghịch đảo của thuật toán AES. Tạo biến inv_sbox_format để lưu s-box nghịch đảo dưới dạng ma trận quen thuộc

- Lưu s-box và s-box nghịch đảo vào file

```
    with open('sbox_tuple.txt', 'w') as file:
        file.write("sbox_tuple = " + str(sbox_tuple) + "\n")
        file.write("inv_sbox_tuple = " + str(inv_sbox_tuple) + "\n")
    print("sbox_tuple và inv_sbox_tuple đã được lưu vào file sbox_tuple.txt")
```

b. Xây dựng chương trình mã hóa và giải mã hình ảnh bằng thuật toán AES với s-box được tạo ra

Đầu tiên ta cần phải tạo ra được thư viện AES của riêng mình, do thư viện AES được cung cấp với ngôn ngữ lập trình Python không cung cấp các thuộc tính cũng như phương thức cho phép thay thế hộp thế s-box mặc định. Chi tiết về việc xây dựng thư viện AES sẽ trình bày cụ thể trong phần Phụ lục 1

- Xây dựng chương trình

  Tạo đường dẫn đến ảnh trên máy tính:

```
# Đường dẫn đến ảnh trên máy tính
    image_path = 'D:/Do_an/do_an/image/beach.png'
    encrypted_image_path =
'D:/Do_an/do_an/image/s_box_other/encrypted_image.png'
    decrypted_image_path =
'D:/Do_an/do_an/image/s_box_other/decrypted_image.png'
```

Biến image_path chứa đường dẫn hình ảnh ban đầu, encrypted_image_path chứa đường dẫn tới hình ảnh đã được mã hóa và decrypted_image_path chứa đường dẫn tới hình ảnh được giải mã

Sử dụng vòng lặp while để xây dựng menu

```
while True:
    print("=================================================================\n")
    print("1. Tải ảnh, chuyển sang chuỗi byte")
    print("2. Mã hóa chuỗi byte hình ảnh")
    print("3. Giải mã chuỗi byte hình ảnh")
    print("4. Dựng hình ảnh từ chuỗi bytes đã được mã hóa và lưu hình ảnh")
    print("5. Dựng hình ảnh từ chuỗi bytes đã được giải mã và lưu hình ảnh")
    print("0. Thoát chương trình")
    condition = input("Nhập lựa chọn: ")
```

Xây dựng các case tương ứng để xử lý trường hợp dựa trên dữ liệu nhập vào

```
if (condition == '1'):
    # 1. Tải ảnh và chuyển sang chuỗi byte
    original_img, img_bytes = load_image_as_bytes(image_path)
    print("Chuyển đổi hình ảnh thành bytes thành công!")
elif (condition == '2'):
    # 2. Mã hóa ảnh
    encrypted_bytes = encrypt(img_bytes, 'ECB', 128)
    print("Đã mã hóa dữ liệu")
elif (condition == '3'):
    # 3. Giải mã ảnh
```

```
    decrypted_bytes = decrypt(encrypted_bytes, 'ECB')
    print("Đã giải mã dữ liệu")
elif (condition == '4'):
    # 4. Lưu ảnh mã hóa dưới dạng ảnh
    encrypted_image = encrypted_bytes_to_image(encrypted_bytes,
original_img.size[0], original_img.size[1])
    encrypted_image.save(encrypted_image_path)
    print("Lưu hình ảnh mã hóa thành công")
elif (condition == '5'):
    # 5. Lưu ảnh đã giải mã dưới dạng ảnh
decrypted_image = decrypted_bytes_to_image(original_img, decrypted_bytes)
    decrypted_image.save(decrypted_image_path)
    print("Lưu hình ảnh giải mã thành công")
elif (condition == '0'):
    break
else:
    print("Giá trị nhập vào không hợp lệ, vui lòng nhập lại!")
```

Các hàm tự tạo được sử dụng trong chương trình sẽ trình bày cụ thể ở Phụ lục 2

3.2.3 Thực nghiệm

a. Tạo hộp thế s-box và s-box nghịch đảo

```
sbox, sbox_tuple, sbox_format = create_sbox(0.0131, 3.64103)
inv_sbox_tuple, inv_sbox_format = create_invsbox(sbox)
```

Với giá trị khởi tạo x0 = 0.0131 và r = 3.64103 ta thu được s-box và s-box nghịch đảo có nội dung như trong hình 3.3 và 3.4

```
Ma trận Sbox:
c3 83 5a dc 66 ff f2 69 2d 80 39 20 86 6e b4 f0
79 46 60 0d 3a fd b6 ab 84 b2 c1 5b bb 29 31 48
b5 5e d4 4f e2 43 4e 11 6f 1b f3 3c c5 b8 e0 87
99 2e 28 ea c0 bc 6a 4b a9 d1 4c 05 3f 37 a3 fc
b7 c2 3e 67 3d fa 59 25 7d 53 95 a0 cb 17 c6 1f
64 4a ec 77 a6 00 08 c4 f5 26 d6 21 6d 0c 1e 73
af 07 ad 0f 32 aa ed 2f a4 44 03 41 e6 24 68 a1
3b 91 d7 04 2b 13 89 7f 19 df 49 0b 4d e1 18 a8
62 ac d9 ee d5 7b cf 14 de 6b e8 5c 10 61 02 96
45 51 34 12 a5 2a 82 94 35 fb be 38 8d 56 e9 1c
57 e7 eb 63 e3 81 b3 65 76 cc b9 7a 9e ae 2c 16
52 15 fe 90 ce b1 0a 0e d2 98 78 97 93 36 09 42
5d 72 1d 8f c9 f1 8e f6 7c cd c7 9d 22 40 06 ca
c8 01 8c ba 8a 75 db a7 9a ef 70 f9 5f 74 d8 a2
1a 9f 47 33 7e 58 bd f7 da 92 9b dd d0 d3 27 85
e5 e4 bf 50 55 71 23 88 9c 8b 6c f4 30 54 b0 f8
```

Hình 3.3: Hộp thế s-box tạo ra với giá trị x0 = 0.0131 và r = 3.64103

```
Ma trận Sbox đảo ngược:
55 d1 8e 6a 73 3b ce 61 56 be b6 7b 5d 13 b7 63
8c 27 93 75 87 b1 af 4d 7e 78 e0 29 9f c2 5e 4f
0b 5b cc f6 6d 47 59 ee 32 1d 95 74 ae 08 31 67
fc 1e 64 e3 92 98 bd 3d 9b 0a 14 70 2b 44 42 3c
cd 6b bf 25 69 90 11 e2 1f 7a 51 37 3a 7c 26 23
f3 91 b0 49 fd f4 9d a0 e5 46 02 1b 8b c0 21 dc
12 8d 80 a3 50 a7 04 43 6e 07 36 89 fa 5c 0d 28
da f5 c1 5f dd d5 a8 53 ba 10 ab 85 c8 48 e4 77
09 a5 96 01 18 ef 0c 2f f7 76 d4 f9 d2 9c c6 c3
b3 71 e9 bc 97 4a 8f bb b9 30 d8 ea f8 cb ac e1
4b 6f df 3e 68 94 54 d7 7f 38 65 17 81 62 ad 60
fe b5 19 a6 0e 20 16 40 2d aa d3 1c 35 e6 9a f2
34 1a 41 00 57 2c 4e ca d0 c4 cf 4c a9 c9 b4 86
ec 39 b8 ed 22 84 5a 72 de 82 e8 d6 03 eb 88 79
2e 7d 24 a4 f1 f0 6c a1 8a 9e 33 a2 52 66 83 d9
0f c5 06 2a fb 58 c7 e7 ff db 45 99 3f 15 b2 05
```

Hình 3.4: Hộp thế s-box nghịch đảo

```
sbox, sbox_tuple, sbox_format = create_sbox(0.00131, 3.64103)
inv_sbox_tuple, inv_sbox_format = create_invsbox(sbox)
```

Thay thế giá trị khởi tạo x0 = 0.00131 và r = 3.64103 ta thu được hộp thế s-box và s-box nghịch đảo có nội dung như trong hình 3.5 và hình 3.6

```
Ma trận Sbox:
9c 48 b0 95 e9 fa 17 49 8e 3e 55 75 f1 df 06 c1
69 58 36 46 5d c5 33 80 32 7d ea 8d c0 4f 0a e3
de ab f5 81 a2 39 9f 76 da 05 aa 4a 50 5c 4b 2c
b4 88 56 08 d1 7e b8 67 ec e0 2d b5 c6 73 ba fd
3f 09 c4 89 61 35 e6 0f 18 51 a0 d5 20 a3 65 3b
cf 72 19 60 1e 5e c7 40 97 f7 66 52 c9 98 a8 cb
31 74 f3 07 8b bd a5 42 e8 fe 86 70 59 bf 6b 12
7a 10 d2 ff 62 37 29 0e 38 0d 1f 04 00 21 27 9e
2e 4d d0 ee e5 bc d8 79 f9 cc b6 b7 5b f8 6f fc
1c 96 3c 6c 53 b3 03 9b 3d 1a d7 e7 82 90 a6 b9
25 28 ed 91 ac 9a 0c 83 d4 15 a9 a7 ce 02 e4 f4
f2 eb d6 2a 34 9d d3 57 af 4c dd 3a dc ad 77 fb
bb 5a 23 f6 68 7f 94 a4 78 26 11 14 c3 92 7c c8
f0 30 63 24 7b 99 e1 45 43 41 8f d9 8c 2f 22 01
85 47 2b db b2 5f ca 93 6e e2 1d 6d 54 16 8a 4e
84 be 6a 71 87 44 64 13 ef 1b 0b b1 cd ae c2 a1
```

Hình 3.5: Hộp thế s-box tạo ra với giá trị khởi tạo khác

```
Ma trận Sbox đảo ngược:
7c df ad 96 7b 29 0e 63 33 41 1e fa a6 79 77 47
71 ca 6f f7 cb a9 ed 06 48 52 99 f9 90 ea 54 7a
4c 7d de c2 d3 a0 c9 7e a1 76 b3 e2 2f 3a 80 dd
d1 60 18 16 b4 45 12 75 78 25 bb 4f 92 98 09 40
57 d9 67 d8 f5 d7 13 e1 01 07 2b 2e b9 81 ef 1d
2c 49 5b 94 ec 0a 32 b7 11 6c c1 8c 2d 14 55 e5
53 44 74 d2 f6 4e 5a 37 c4 10 f2 6e 93 eb e8 8e
6b f3 51 3d 61 0b 27 be c8 87 70 d4 ce 19 35 c5
17 23 9c a7 f0 e0 6a f4 31 43 ee 64 dc 1b 08 da
9d a3 cd e7 c6 03 91 58 5d d5 a5 97 00 b5 7f 26
4a ff 24 4d c7 66 9e ab 5e aa 2a 21 a4 bd fd b8
02 fb e4 95 30 3b 8a 8b 36 9f 3e c0 85 65 f1 6d
1c 0f fe cc 42 15 3c 56 cf 5c e6 5f 89 fc ac 50
82 34 72 b6 a8 4b b2 9a 86 db 28 e3 bc ba 20 0d
39 d6 e9 1f ae 84 46 9b 68 04 1a b1 38 a2 83 f8
d0 0c b0 62 af 22 c3 59 8d 88 05 bf 8f 3f 69 73
```

Hình 3.6: Hộp thế s-box nghịch đảo

Như vậy chỉ cần thay đổi nhỏ ở tham số đầu vào, ta sẽ thu được kết quả hoàn toàn khác

b. Mã hóa dữ liệu

Khởi tạo dữ liệu đầu vào

```
# Dữ liệu đầu vào cần mã hóa
    my_datas = 'Nguyen Xuan Hieu - AT170119'
    encrypt_datas = encrypt(my_datas.encode('utf-8'), 'ECB', 128)
    decrypt_datas = decrypt(encrypt_datas, 'ECB')
    print('Dữ liệu khi được mã hóa: ', encrypt_datas)
    print('Dữ liệu sau khi được giải mã: ', decrypt_datas.decode('utf-8'))
```

Mã hóa và giải mã, thu được kết quả như trong hình 3.7

```
ECB encryption: 100%|                                                    | 2/2 [00:00<00:00, 2003.97it/s]
Cipher Key: 362bb94d8f9db56b11b8bc3ed0c6bbab
ECB decryption: 100%|                                                    | 2/2 [00:00<00:00, 2007.32it/s]
Dữ liệu khi được mã hóa:  bytearray(b'\x8b\xd5+\x8f6\x03\xc1\xdc\xc9co\x06\xe3#\xbc+[\xae\xe
fB\xba\x19\x13\x98\xcd\xe8\x97\x15Ph+\xfa')
Dữ liệu sau khi được giải mã:  Nguyen Xuan Hieu - AT170119
```

Hình 3.7: Kết quả thu được sau khi mã hóa và giải mã dữ liệu

Thay đổi dữ liệu đầu vào

```
# Dữ liệu đầu vào cần mã hóa
    my_datas = 'Nguyen Xuan Hieu - AT170119 - Do an tot nghiep'
    encrypt_datas = encrypt(my_datas.encode('utf-8'), 'ECB', 192)
    decrypt_datas = decrypt(encrypt_datas, 'ECB')
    print('Dữ liệu khi được mã hóa: ', encrypt_datas)
    print('Dữ liệu sau khi được giải mã: ', decrypt_datas.decode('utf-8'))
```

Mã hóa và giải mã, thu được kết quả trong hình 3.8

```
PS D:\Do_an\do_an> py .\new_aes2.py
ECB encryption: 100%|                                                    | 3/3 [00:00<00:00, 3007.39it/s]
Cipher Key: 79bf468ce988d94e13772fc3b1e9179a117e990936a60589
ECB decryption: 100%|                                                    | 3/3 [00:00<00:00, 3005.95it/s]
Dữ liệu khi được mã hóa:  bytearray(b'\xc3\xaa\xae?G>\xffu\xcf0\xde\xb5\x82\x01en\xc6\xff\x8
ft\xca\xaa*\x95\xe0\x0e\xc4#\x82\xe0\x1d\xc0ln&\xb64#*c\xf4\xf3R\xee\x9dn\xaf\x0c')
Dữ liệu sau khi được giải mã:  Nguyen Xuan Hieu - AT170119 - Do an tot nghiep
```

Hình 3.8: Kết quả thu được sau khi mã hóa và giải mã dữ liệu với đầu vào khác

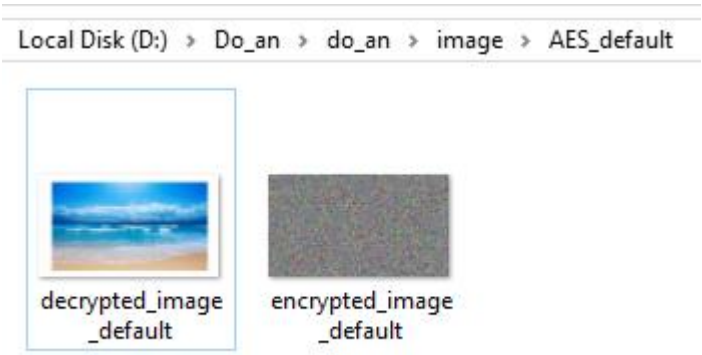
c. Mã hóa hình ảnh bằng sbox mặc định của AES

Khởi tạo dữ liệu đầu vào

```
    # Đường dẫn đến ảnh trên máy tính
    image_path = 'D:/Do_an/do_an/image/beach.png'
    encrypted_image_path =
'D:/Do_an/do_an/image/s_box_other/encrypted_image.png'
    decrypted_image_path =
'D:/Do_an/do_an/image/s_box_other/decrypted_image.png'

    # Khóa AES 128-bit và IV
    key = b'nguyen xuan hieu'
    iv = get_random_bytes(16)
```

Kết quả sau khi chạy chương trình thu được kết quả như trong hình 3.7



Hình 3.9: Hình ảnh mã hóa và giải mã được lưu lại

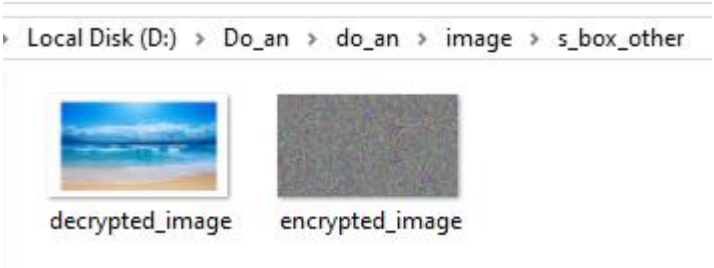
Hình ảnh đã được mã hóa và giải mã và lưu lại đúng đường dẫn

d. Mã hóa hình ảnh bằng s-box tạo ra

Khởi tạo giá trị đầu vào

```
    # Đường dẫn đến ảnh trên máy tính
    image_path = 'D:/Do_an/do_an/image/beach.png'
    encrypted_image_path =
'D:/Do_an/do_an/image/s_box_other/encrypted_image.png'
    decrypted_image_path =
'D:/Do_an/do_an/image/s_box_other/decrypted_image.png'
```

Kết quả sau khi chạy chương trình thu được kết quả như trong hình 3.8



Hình 3.10: Hình ảnh mã hóa và giải mã được lưu lại

3.3 Đánh giá các tính chất của S-box

Ta sẽ thực hiện phân tích và so sánh một số tính chất mật mã của hộp thế được tạo ra và s-box tiêu chuẩn được sử dụng trong AES. Các giá trị so sánh được thể hiện trong bảng

Để giúp thuận tiện hơn trong việc đánh giá tính chất của hộp thế S-box, trong phần này sẽ dùng một công cụ có sẵn đó là Sage Math để đánh giá, chi tiết về công cụ này sẽ được trình bày trong phần Phụ lục 3 của đồ án.

Sau khi phân tích và tính toán, ta thu được kết quả như trong bảng 3.1

Bảng 3.1: Bảng giá trị của các tính chất mật mã của s-box

Tính chất của S-Box	**S-Box tiêu chuẩn AES**	**S-box tạo ra**
Tính cân bằng	Có	Có
Độ phi tuyến	112	94

Bậc đại số	7	7
Điểm bất động	0	0
Giá trị xấp xỉ vi sai cực đại	4	12
Giá trị xấp xỉ tuyến tính cực đại	16	34

Theo bảng trên ta có thể thấy cụ thể các tính chất của hộp thế được tạo ra như sau:

- Về tính cân bằng:

S-box thỏa mãn tính cân bằng, vì vậy có khả năng chống lại tấn công thống kê, giúp cho thuật toán đảm bảo được an toàn hơn.

- Về độ phi tuyến:

Như chúng ta đã biết độ phi tuyến là yếu tố quan trọng nhất của hộp thế trong mỗi thuật toán mã hóa. Độ phi tuyến càng cao thì càng tốt. Dựa vào kết quả được đưa ra trong bảng ta có thể nhận thấy được độ phi tuyến của hộp thế là 94, được coi là khá tốt cho một S-box trong các thuật toán mã hóa.

- Về bậc đại số:

Bậc đại số của S-box là một chỉ số quan trọng trong việc đánh giá độ bảo mật và tính hiệu quả của các thuật toán mã hóa. Như ta biết, bậc đại số càng cao càng tốt. Bậc đại số có ý nghĩa đặc trưng cho việc chống lại tấn công thám mã vi sai và tấn công đại số. Bậc đại số là 7 được coi là rất tốt, cho thấy độ phức tạp cao

- Về điểm bất động của S-box:

Số điểm bất động là số cặp đầu vào – đầu ra khi qua S-box là như nhau. Càng nhiều điểm bất động, kẻ thám mã sẽ càng dễ dàng tìm ra quy luật để tấn công hơn. Đối với S-box số điểm bất động đều bằng 0. Như vậy tính an toàn được đảm bảo hoàn toàn.

- Về giá trị xấp xỉ vi sai cực đại:

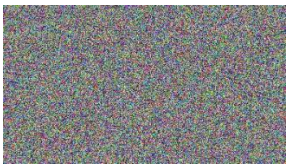
Giá trị xấp xỉ vi sai của hộp thế càng thấp thì khả năng chống lại tấn công vi sai càng tốt. Chỉ số này đo lường mức độ "cân bằng" của S-box, tức là khả năng của nó trong việc tạo ra đầu ra khác nhau cho những đầu vào khác nhau.

- Về giá trị xấp xỉ tuyến tính:

Xác xuất xấp xỉ tuyến tính càng thấp, khả năng chống lại thám mã vi sai càng lớn. Khả năng chống lại tấn công tuyến tính của hộp thế AES là khá tốt, giá trị xấp xỉ tuyến tính của AES là 16, nhỏ hơn nhiều so với S-box được tạo ra là 34

3.4 Đánh giá các tính chất của hình ảnh mã hóa, so sánh giữa hình ảnh được mã hóa bởi hộp thế mặc định của AES với hộp thế được tạo ra

Phần này sẽ tập trung đi vào phân tích entropy, histogram và correlation của ảnh gốc, ảnh mã hóa bằng hộp thế mặc định, ảnh mã hóa bằng hộp thế tạo ra. Toàn bộ code thực thi phân tích bằng công cụ Matlab sẽ trình bày ở phần Phụ lục 4

Ảnh gốc	Ảnh mã hóa default	Ảnh mã hóa other
		

3.4.1 Entropy analysis

Entropy được định nghĩa là độ thông tin hay phép đo tính ngẫu nhiên của dữ liệu trong một hình ảnh. Nếu giá trị của entropy cao, điều đó có nghĩa là dữ liệu trong hình ảnh càng bị rối loạn. Giá trị entropy tối đa trên lý thuyết là 8, xảy ra khi tất cả giá trị pixel từ 0 đến 255 xuất hiện với xác xuất bằng nhau trong ảnh, thể hiện sự ngẫu nhiên hoàn toàn

Entropy H của ảnh được tính dựa trên xác suất xuất hiện của mỗi giá trị pixel trong ảnh. Với ảnh 8-bit (các giá trị pixel từ 0 đến 255), công thức entropy là:

$$H = -\sum_{i=0}^{255} P(i)\, log_2\, P(i)$$

Trong đó: $P(i)$ là xác suất xuất hiện của giá trị pixel i trong ảnh

Sau khi phân tích bằng công cụ Matlab, thu được kết quả như trong bảng 3.2

Bảng 3.2: Bảng giá trị entropy toàn cục

Entropy	Ảnh gốc	Ảnh mã hóa (S-box AES)	Ảnh mã hóa (S-box tạo ra)
R (red)	Entropy = 7.0816	Entropy = 7.9977	Entropy = 7.9976
G (green)	Entropy = 6.5849	Entropy = 7.9978	Entropy = 7.9980
B (blue)	Entropy = 6.3828	Entropy = 7.9977	Entropy = 7.9982

Phân tích entropy của 10 ô vuông có kích thước 44x44 không chồng lên nhau rồi lấy giá trị trung bình, ta thu được kết quả như trong bảng 3.3

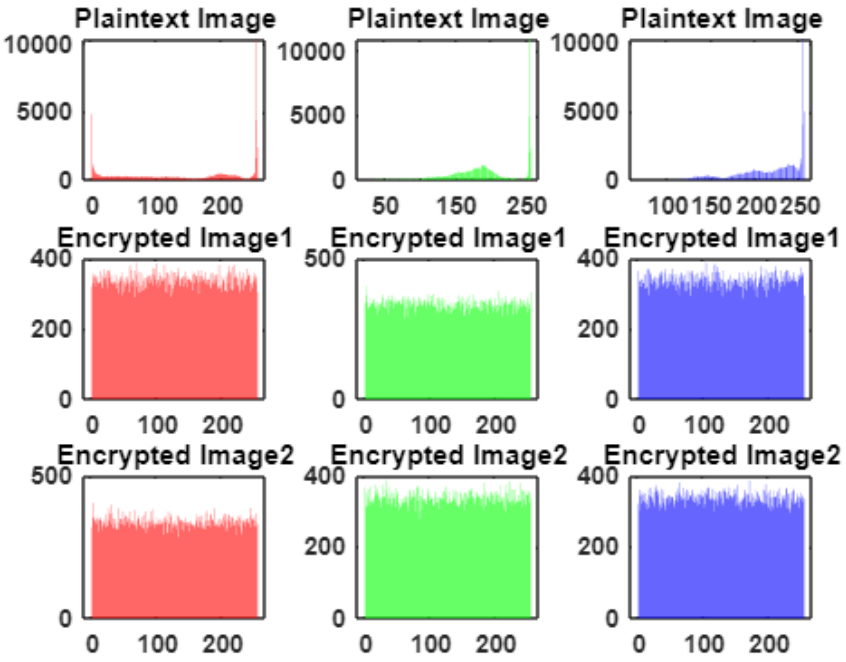
Bảng 3.3: Bảng giá trị entropy cục bộ

Entropy	Ảnh gốc	Ảnh mã hóa (S-box AES)	Ảnh mã hóa (S-box tạo ra)
R (red)	Entropy = 6.5286	Entropy = 5.8853	Entropy = 5.5577
G (green)	Entropy = 7.9038	Entropy = 7.9003	Entropy = 7.9092
B (blue)	Entropy = 7.9077	Entropy = 7.9048	Entropy = 7.9048

Như vậy ta có thể thấy entropy của hình ảnh mã hóa từ 2 hộp thế có giá trị khá tương đồng nhau và gần đạt giá trị tuyệt đối, khẳng định rằng dữ liệu của ảnh khó đoán và có khả năng kháng lại việc giải mã mà không có khóa phù hợp

3.4.2 Histogram analysis

Histogram của hình ảnh mã hóa cho biết tần suất xuất hiện của các giá trị màu trong hình ảnh bản rõ và bản mã. Biểu đồ hình ảnh bản mã thường được phân bố đồng đều và ít biến động trong khi đó biểu đồ hình ảnh bản rõ thường biến động, kẻ tấn công thường sử dụng thuộc tính tấn công này để thám mã. Để tránh nguy cơ này, mã hóa phải loại bỏ được thuộc tính này của hình ảnh bản rõ trong quá trình mã hóa. Phân tích biểu đồ bằng công cụ Matlab, ta thu được kết quả như trong hình 3.9



Hình 3.11: Biểu đồ histogram các kênh màu của hình ảnh

Phương sai của hình ảnh là một chỉ số đo lường sự phân bố độ sáng trong hình ảnh, cho biết mức độ biến đổi của các giá trị pixel so với trung bình. Giá trị phương sai càng cao nghĩa là hình ảnh có độ biến đổi cao giữa các vùng sáng và tối, tức là nhiều chi tiết hoặc độ tương phản cao.

Giá trị phương sai được tính bằng công thức:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(I_i - \mu)^2$$

Trong đó: N là tổng số pixel trong ảnh

μ là giá trị trung bình độ sáng của toàn bộ ảnh

I_i là giá trị độ sáng của pixel thứ i

Tính toán giá trị phương sai của các kênh R, G, B ta thu được kết quả trong bảng 3.4

Bảng 3.4: Bảng giá trị phương sai của hình ảnh

Phương sai	Ảnh gốc	Ảnh mã hóa (S-box AES)	Ảnh mã hóa (S-box tạo ra)
R	Variance = 5.8119e+05	Variance = 369.2863	Variance = 376.3608
G	Variance = 6.4886e+05	Variance = 339.2863	Variance = 311.0667
B	Variance = 6.4304e+05	Variance = 362.2980	Variance = 288.1490

3.4.3 Correlation Analysis

Nó được định nghĩa là mối quan hệ giữa các pixel tạo nên hình ảnh thực tế và các pixel tạo nên hình ảnh được mã hóa. Dựa trên mối tương quan, sự giống nhau giữa hình ảnh thực tế và hình ảnh được mã hóa sẽ được đánh giá. Để có một quá trình mã hóa tốt nhất, phải có mối tương quan thấp

Ta có công thức tính độ tương quan như sau:

$$r = \frac{\sum_{i=1}^{N} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N} (X_i - \bar{X})^2 \sum_{i=1}^{N} (Y_i - \bar{Y})}}$$

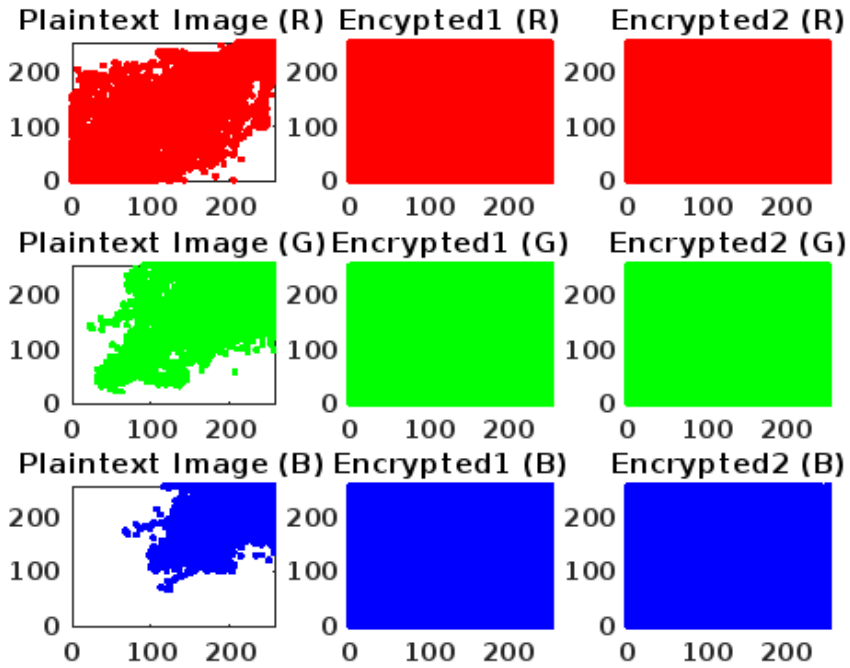
Trong đó:

X_i và Y_i là các giá trị pixel tương ứng trong ảnh gốc và ảnh hóa tại vị trí i

$\bar{X}$ và $\bar{Y}$ là giá trị trung bình của các pixel trong ảnh gốc và ảnh mã hóa

N là tổng số pixel trong ảnh

Phân tích bằng công cụ Matlab ta thu được kết quả trong hình 3.10:



Hình 3.12: Mối tương quan correlation các kênh màu của hình ảnh

Tính toán giá trị độ tương quan kênh màu R (red) ta thu được kết quả trong bảng 3.5

Bảng 3.5: Độ tương quan của kênh màu red

	Plaintex image	Encrypted image 1 (S-box AES)	Encrypted image 2 (S-box tạo ra)
Horizontal	0.9945	-0.0006	0.0025
Vertical	0.9843	-0.0013	0.0003
Diagonal	0.9795	-0.0054	-0.0013

Tính toán giá trị độ tương quan kênh màu G (green) ta thu được kết quả như trong bảng 3.6

Bảng 3.6: Độ tương quan của kênh màu green

	Plaintex image	Encrypted image 1 (S-box AES)	Encrypted image 2 (S-box tạo ra)
Horizontal	0.9855	-0.0025	-0.0016
Vertical	0.9561	0.0008	0.0027
Diagonal	0.9443	0.0008	0.0029

Tính toán giá trị độ tương quan kênh màu B (blue) ta thu được kết quả trong bảng 3.7

Bảng 3.7: Độ tương quan của kênh màu blue

	Plaintex image	Encrypted image 1 (S-box AES)	Encrypted image 2 (S-box tạo ra)
Horizontal	0.9828	0.0047	-0.0037
Vertical	0.9408	0.0055	-0.0030
Diagonal	0.9279	0.0058	0.0043

Giá trị tương quan nằm trong khoảng từ -1 đến 1. Với $r = 1$ là tương quan hoàn hảo, nghĩa là 2 hình ảnh tương đồng với nhau. Với $r = -1$ là tương quan âm hoàn hảo, tức là ảnh gốc và ảnh mã hóa là sự nghịch đảo của nhau. Với $r = 0$ tức là không có sự tương quan lẫn nhau, tức là không có bất kỳ sự liên quan nào giữa bản rõ và bản mã

Dựa vào 3 bảng giá trị tính toán được ở bên trên ta có thể thấy được hình ảnh mã hóa được tạo bởi 2 hộp thế khác nhau đều có giá trị xấp xỉ bằng 0. Như vậy cả 2 hình ảnh mã hóa đều được coi là mã hóa hóa rất tốt

Tổng kết

Trong báo cáo, đề xuất một S-box được cấu trúc bởi một thuật toán cực kỳ đơn giản và trực tiếp. Dựa trên một công thức toán học được sử dụng để xây dựng AES S-box tiêu chuẩn.

Về ưu điểm : Thuật toán làm tăng độ phức tạp cho thuật toán mã hóa, mỗi một phiên mã hoá dữ liệu ta có thể sử dụng một S-box khác nhau mà vẫn đảm bảo các S-box có các tính chất giống như S-box tiêu chuẩn.

Về nhược điểm: đó là một số tính chất mật mã của hộp thế mới thấp hơn hộp thế tiêu chuẩn của AES dẫn đến khả năng bảo mật thấp hơn. Bên cạnh đó, nếu muốn áp dụng hộp thế s-box tự tạo để mã hóa và giải mã, giữa người gửi và người nhận dữ liệu cần xây dựng một kênh truyền dữ liệu an toàn để trao đổi về các thông số đầu vào cũng như khoá.

KẾT LUẬN

Trong bài báo cáo này đã giới thiệu tổng quan về hệ mật mã, các tính chất mật mã, hàm Boolean, đánh giá một số tính chất hộp thế và trình bày về thuật toán tạo S-box đựa trên ánh xạ logistic và ứng dụng hộp thế tạo ra trong việc mã hóa và giải mã dữ liệu, hình ảnh

Ba chương của đồ án đã thể hiện được rằng những mục tiêu đặt ra khi thực hiện đồ án đều đã đạt được. Cụ thể:

Chương 1 đã giới thiệu tổng quan về mật mã, hệ mật mã khối và chỉ ra được vai trò của hộp thế (S-box) trong mật mã khối. Giới thiệu về hệ mật AES và hộp thế trong thuật toán mã hóa AES. Trình bày về hàm Boolean và các tính chất mật mã của hàm Boolean, các tính chất mật mã của hộp thế.

Chương 2 đã giới thiệu và tìm hiểu một số phương pháp tạo hộp thế, đi sâu vào nghiên cứu phương pháp tạo hộp thế dựa trên ánh xạ hỗn loạn một chiều pure. Xây dựng thuật toán tạo s-box động dựa trên ánh xạ logistics

Chương 3 đã cài đặt chương trình thử nghiệm tạo S-box động và áp dụng hộp thế được tạo ra vào thuật toán AES để mã hóa và giải mã dữ liệu, hình ảnh. Đánh giá một số tính chất mật mã của hộp thế và hình ảnh mã hóa được tạo ra và sự khác biệt của chúng so với hộp thế mặc định của AES

Tuy nhiên, do hạn chế về thời gian, kinh nghiệm và kiến thức chưa được sâu sắc nên trong bài báo cáo về đề tài của em không tránh khỏi những thiếu sót, mong quý thầy cô góp ý thêm để em có thể bổ sung kiến thức cho bản thân để thực hiện những nghiên cứu chuyên sâu hơn sau này.

TÀI LIỆU THAM KHẢO

[1]. **Nguyễn Văn Kiên**, *Mật mã khối và mật mã khóa đối xứng*, Trường Đại học Bách khoa Hà nội, 2008.

[2]. **Trần Văn Trường, Trần Quang Kỳ**, *Giáo trình mật mã học nâng cao*, Học viện Kỹ thuật mật mã, 2007.

[3]. **Nguyễn Đình Vinh, Trần Quang Kỳ**, *Giáo trình cơ sở an toàn thông tin*, Học viện Kỹ thuật mật mã, 2006.

[4]. **Nguyễn Bình, Hoàng Thu Phương**, *Cơ sở lý thuyết mật mã*, Giáo trình Học viện kỹ thuật mật mã, 2013.

[5]. **V. Afraimovich, S. Bykov**, *Discrete chaotic maps and their applications*, Springerplus, 2016.

[6]. **Daemen J, Rijmen V**, *The design of Rijndael-AES: the advanced encrytion standard*, Berlin Springer, 2020.

[7]. **Ramamoorthy V, Silaghi MC, Matsui T, Hirayama K, Yokoo M**, *The design of cryptographic S-boxes using CSPs*, 2011.

https://link.springer.com/chapter/10.1007/978-3-642-23786-7_7

[8]. **Rasool S. Salman**, **Alaa K. Farhan**, **Ali Sharkir** *Creation of S-Box based One-Dimensional Chaotic Logistic Map*, 2017

https://inass.org/wp-content/uploads/2022/05/2022103133-3.pdf

[9]. **The Sage Development Team**, *S-Boxes and Their Algebraic Representations*,

https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/sbox.html

[10]. **Nguyễn Quân**, *[AES] Bài 5 - Tối ưu logic tính S-Box dựa trên biến đổi toán học*, 2019

https://nguyenquanicd.blogspot.com/2019/10/aes-bai-5-toi-uu-logic-tinh-s-box-dua.html

[11]. **X. Wang, G. Chen,** *Chaotic maps and their applications in secure communications*, 2012

[12]. **John D.Cook, PhD**, The AES (Rijndael) S-box – John D.Cook, 2019 https://www.johndcook.com/blog/2019/05/25/aes-s-box/

[13]. **Ramirez-Torres MT, Murguia JS, Carlos MM**, *Image encryption with an improved cryptosystem based on a matrix approach*, 2014.

[14]. **Carlet C, Ding C**, *Highly nonlinear mappings. J Complex*, 2004.

[15]. **Tran MT, Bui DK, Doung AD**, *Gray S-box for advanced encryption standard. In: International conference computational intelligence and security*, 2008.

[16]. **Benvenuto CJ**, *Galois field in cryptography*, Seattle: University of Washington, 2012.

[17]. **Biham E, Shamir A**, *Differential cryptanalysis of DES-like cryptosystems*, J Cryptol, 1991.

[18]**. Indrajit Das, Sanjoy Roy, Subhrapratim Nath, Subhash Mondal**, *Random S-Box Generation in AES by changing Irreducible polynomial*, Meghnad Saha Institute of Technology, Kolkata, India, 2012

[19]. **Praveen Agarwal, Amandeep Singh, Adem Kilicman**, *Development of key-dependent dynamic S-Boxes with dynamic irreducible polynomial and affine constant*, 2018

[20]. **S. Das, J.K.M.S. Uz Zaman, R. Ghosh**, *Generation of AES S-Boxes with various modulus and additive constant polynomials and testing their randomization*, 2013

PHỤ LỤC 1

Một số hàm chính tạo thư viện AES

1. Hàm sub_bytes và inv_sub_bytes

```
    def sub_bytes(self, s):
        for i in range(self.Nb):
            for j in range(4):
                s[i][j] = self.Sbox[s[i][j]]

    def inv_sub_bytes(self, s):
        for i in range(self.Nb):
            for j in range(4):
                s[i][j] = self.InvSbox[s[i][j]]
```

2. Hàm shift_rows và inv_shift_row

```
    def shift_rows(self, s):
        s[0][1], s[1][1], s[2][1], s[3][1] = s[1][1], s[2][1], s[3][1], s[0][1]
        s[0][2], s[1][2], s[2][2], s[3][2] = s[2][2], s[3][2], s[0][2], s[1][2]
        s[0][3], s[1][3], s[2][3], s[3][3] = s[3][3], s[0][3], s[1][3], s[2][3]
```

3. Hàm mix_column và inv_mix_column

```
    def mix_columns(self, s):
        for i in range(self.Nb):
            self.mix_one_column(s[i])

    def inv_mix_columns(self, s):
        for i in range(self.Nb):
            u = self.xtime(self.xtime(s[i][0] ^ s[i][2]))
            v = self.xtime(self.xtime(s[i][1] ^ s[i][3]))
            s[i][0] ^= u
            s[i][1] ^= v
            s[i][2] ^= u
            s[i][3] ^= v
        self.mix_columns(s)
```

4. Hàm add_round_key

```
    def add_round_key(self, s, k):
        for i in range(self.Nb):
            for j in range(4):
                s[i][j] ^= k[i][j]
```

5. Hàm mở rộng khóa key_expansion

```
    def key_expansion(self, key):
```

```
        self.round_keys = self.key
        for i in range(self.Nk, self.Nb * (self.Nr + 1)):
            self.round_keys.append([0, 0, 0, 0])
            temp = self.round_keys[i - 1][:]
            # word is multiple of Nk
            if i % self.Nk == 0:
                self.rotate_word(temp)
                self.sub_word(temp)
                temp[0] = temp[0] ^ self.Rcon[i // self.Nk]
            elif self.Nk > 6 and i % self.Nk == 4:
                self.sub_word(temp)
            for j in range(4):
                self.round_keys[i][j] = self.round_keys[i - self.Nk][j] ^ temp[j]
```

6. Hàm mã hóa và giải mã

```
def cipher(self, text):
    self.state = self.text2matrix(text)
    self.add_round_key(self.state, self.round_keys[:4])
    for i in range(1, self.Nr):
        self.sub_bytes(self.state)
        self.shift_rows(self.state)
        self.mix_columns(self.state)
        self.add_round_key(self.state, self.round_keys[self.Nb * i : self.Nb * (i
+ 1)])
    self.sub_bytes(self.state)
    self.shift_rows(self.state)
    self.add_round_key(self.state, self.round_keys[len(self.round_keys) - 4:])
    return self.matrix2text(self.state)

def decipher(self, text):
    self.encrypted_state = self.text2matrix(text)
    self.add_round_key(self.encrypted_state, self.round_keys[len(self.round_keys)
- 4:])
    for i in range(self.Nr - 1, 0, -1):
        self.inv_shift_rows(self.encrypted_state)
        self.inv_sub_bytes(self.encrypted_state)
        self.add_round_key(self.encrypted_state, self.round_keys[self.Nb * i :
self.Nb * (i + 1)])
        self.inv_mix_columns(self.encrypted_state)
    self.inv_shift_rows(self.encrypted_state)
    self.inv_sub_bytes(self.encrypted_state)
    self.add_round_key(self.encrypted_state, self.round_keys[:4])
    return self.matrix2text(self.encrypted_state)
```

7. Hàm pad và unpad

```
def pad(block, block_length):
    bytes_to_pad = block_length - len(block) // 2
    for _ in range(bytes_to_pad):
        block += format(bytes_to_pad, '02x')
    return block

def unpad(block):
    bytes_to_unpad = int(block[-2:], 16)
    return block[:-bytes_to_unpad*2]
```

Ngoài ra em còn xây dựng một số class để chọn chế độ mã hóa như ECB, CBC và CTR và chọn độ dài khóa 128, 192, 256 bit

PHỤ LỤC 2

Một số hàm sử dụng trong việc mã hóa và giải mã hình ảnh

1. Hàm tải hình ảnh và chuyển hình ảnh thành chuỗi byte

```
def load_image_as_bytes(filepath):
    img = Image.open(filepath)
    img_data = np.array(img)
    img_bytes = img_data.tobytes()  # Chuyển ảnh thành chuỗi byte
    return img, img_bytes
```

2. Hàm mã hóa hình ảnh

```
def aes_encrypt_image(image_bytes, key, iv):
    print(len(image_bytes))
    cipher = AES.new(key, AES.MODE_CBC, iv)
    padded_data = pad(image_bytes, AES.block_size)
    encrypted_data = cipher.encrypt(padded_data)
    return encrypted_data
```

3. Hàm giải mã hình ảnh

```
def aes_decrypt_image(image_bytes, key, iv):
    print(len(image_bytes))
    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_padded = cipher.decrypt(image_bytes)
    decrypted_data = unpad(decrypted_padded, AES.block_size)
    return decrypted_data
```

4. Hàm dựng hình ảnh từ các bytes dữ liệu mã hóa

```
def encrypted_bytes_to_image(encrypted_bytes, width, height):
    encrypted_array = np.frombuffer(encrypted_bytes, dtype=np.uint8)[:width *
height * 3]
    encrypted_image = encrypted_array.reshape((height, width, 3))
    return Image.fromarray(encrypted_image)
```

5. Hàm dựng hình ảnh gốc từ các bytes dữ liệu đã giải mã

```
def decrypted_bytes_to_image(img, decrypted_bytes):
    img_data = np.frombuffer(decrypted_bytes,
dtype=np.uint8).reshape(img.size[1], img.size[0], -1)
    decrypted_image = Image.fromarray(img_data)
    return decrypted_image
```

6. Hàm mã hóa AES

```
def encrypt(img_bytes, block_cipher_mode, key_length):
    key = aes.random_key_generator(int(key_length))
    if key_length == 128:
        AES = aes.AES(key, 128)
```

```
    elif key_length == 192:
        AES = aes.AES(key, 192)
    elif key_length == 256:
        AES = aes.AES(key, 256)
    if block_cipher_mode == "ECB":
        bcm = aes.ECB(AES)
    elif block_cipher_mode == "CBC":
        bcm = aes.CBC(AES, 16)
    elif block_cipher_mode == "CTR":
        bcm = aes.CTR(AES)
    encrypted_data = bcm.cipher(img_bytes)
    write_key(key)
    print("Cipher Key:", key)
    return encrypted_data
```

7. Hàm giải mã AES

```
def decrypt(encrypted_data, block_cipher_mode):
    key = read_key()
    key_length = len(key) * 4

    if key_length == 128:
        AES = aes.AES(key, 128)
    elif key_length == 192:
        AES = aes.AES(key, 192)
    elif key_length == 256:
        AES = aes.AES(key, 256)

    if block_cipher_mode == "ECB":
        bcm = aes.ECB(AES)
    elif block_cipher_mode == "CBC":
        bcm = aes.CBC(AES, 16)
    elif block_cipher_mode == "CTR":
        bcm = aes.CTR(AES)
    decrypted_data = bcm.decipher(encrypted_data)
    return decrypted_data
```

PHỤ LỤC 3

Kiểm tra các tính chất mật mã của hộp thế bằng công cụ Sage Math

1. Kiểm tra tính cân bằng của S-box

- Sử dụng hàm is_balanced().

- Trả lại True nếu S-box là cân bằng

- Một S-box được gọi là cân bằng nếu tất cả chức năng thành phần của nó cân bằng

```
sage: from sage.crypto.sbox import SBox
sage: S = SBox([195, 131, 90, 220, 102, 255, 242, 105, 45, 128, 57, 32, 134, 110, 180, 240, 121, 70, 96,
....: 13, 58, 253, 182, 171, 132, 178, 193, 91, 187, 41, 49, 72, 181, 94, 212, 79, 226, 67, 78, 17, 111,
....: 27, 243, 60, 197, 184, 224, 135, 153, 46, 40, 234, 192, 188, 106, 75, 169, 209, 76, 5, 63, 55, 163,
....:  252, 183, 194, 62, 103, 61, 250, 89, 37, 125, 83, 149, 160, 203, 23, 198, 31, 100, 74, 236, 119, 1
....: 66, 0, 8, 196, 245, 38, 214, 33, 109, 12, 30, 115, 175, 7, 173, 15, 50, 170, 237, 47, 164, 68, 3, 6
....: 5, 230, 36, 104, 161, 59, 145, 215, 4, 43, 19, 137, 127, 25, 223, 73, 11, 77, 225, 24, 168, 98, 172
....: , 217, 238, 213, 123, 207, 20, 222, 107, 232, 92, 16, 97, 2, 150, 69, 81, 52, 18, 165, 42, 130, 148
....: , 53, 251, 190, 56, 141, 86, 233, 28, 87, 231, 235, 99, 227, 129, 179, 101, 118, 204, 185, 122, 158
....: , 174, 44, 22, 82, 21, 254, 144, 206, 177, 10, 14, 210, 152, 120, 151, 147, 54, 9, 66, 93, 114, 29,
....:  143, 201, 241, 142, 246, 124, 205, 199, 157, 34, 64, 6, 202, 200, 1, 140, 186, 138, 117, 219, 167,
....:  154, 239, 112, 249, 95, 116, 216, 162, 26, 159, 71, 51, 126, 88, 189, 247, 218, 146, 155, 221, 208
....: , 211, 39, 133, 229, 228, 191, 80, 85, 113, 35, 136, 156, 139, 108, 244, 48, 84, 176, 248])
sage: S.is_balanced()
True
```

2. Tính độ phi tuyến của S-box

- Sử dụng hàm nonlinearity( )

- Trả về độ phi tuyến của S-box

- Tính phi tuyến của một S-Box được định nghĩa là độ phi tuyến tính tối thiểu của tất cả các hàm thành phần của nó.

```
sage: from sage.crypto.sbox import SBox
sage: S = SBox([195, 131, 90, 220, 102, 255, 242, 105, 45, 128, 57, 32, 134, 110, 180, 240, 121, 70, 96,
....: 13, 58, 253, 182, 171, 132, 178, 193, 91, 187, 41, 49, 72, 181, 94, 212, 79, 226, 67, 78, 17, 111,
....: 27, 243, 60, 197, 184, 224, 135, 153, 46, 40, 234, 192, 188, 106, 75, 169, 209, 76, 5, 63, 55, 163,
....:  252, 183, 194, 62, 103, 61, 250, 89, 37, 125, 83, 149, 160, 203, 23, 198, 31, 100, 74, 236, 119, 1
....: 66, 0, 8, 196, 245, 38, 214, 33, 109, 12, 30, 115, 175, 7, 173, 15, 50, 170, 237, 47, 164, 68, 3, 6
....: 5, 230, 36, 104, 161, 59, 145, 215, 4, 43, 19, 137, 127, 25, 223, 73, 11, 77, 225, 24, 168, 98, 172
....: , 217, 238, 213, 123, 207, 20, 222, 107, 232, 92, 16, 97, 2, 150, 69, 81, 52, 18, 165, 42, 130, 148
....: , 53, 251, 190, 56, 141, 86, 233, 28, 87, 231, 235, 99, 227, 129, 179, 101, 118, 204, 185, 122, 158
....: , 174, 44, 22, 82, 21, 254, 144, 206, 177, 10, 14, 210, 152, 120, 151, 147, 54, 9, 66, 93, 114, 29,
....:  143, 201, 241, 142, 246, 124, 205, 199, 157, 34, 64, 6, 202, 200, 1, 140, 186, 138, 117, 219, 167,
....:  154, 239, 112, 249, 95, 116, 216, 162, 26, 159, 71, 51, 126, 88, 189, 247, 218, 146, 155, 221, 208
....: , 211, 39, 133, 229, 228, 191, 80, 85, 113, 35, 136, 156, 139, 108, 244, 48, 84, 176, 248])
sage: S.nonlinearity()
94
```

3. Tính bậc đại số cực đại

- Sử dụng hàm max_degree( )

- Trả về bậc đại số cực đại của tất cả các hàm thành phần của nó

```
sage: from sage.crypto.sbox import SBox
sage: S = SBox([195, 131, 90, 220, 102, 255, 242, 105, 45, 128, 57, 32, 134, 110, 180, 240, 121, 70, 96,
....: 13, 58, 253, 182, 171, 132, 178, 193, 91, 187, 41, 49, 72, 181, 94, 212, 79, 226, 67, 78, 17, 111,
....: 27, 243, 60, 197, 184, 224, 135, 153, 46, 40, 234, 192, 188, 106, 75, 169, 209, 76, 5, 63, 55, 163,
....:  252, 183, 194, 62, 103, 61, 250, 89, 37, 125, 83, 149, 160, 203, 23, 198, 31, 100, 74, 236, 119, 1
....: 66, 0, 8, 196, 245, 38, 214, 33, 109, 12, 30, 115, 175, 7, 173, 15, 50, 170, 237, 47, 164, 68, 3, 6
....: 5, 230, 36, 104, 161, 59, 145, 215, 4, 43, 19, 137, 127, 25, 223, 73, 11, 77, 225, 24, 168, 98, 172
....: , 217, 238, 213, 123, 207, 20, 222, 107, 232, 92, 16, 97, 2, 150, 69, 81, 52, 18, 165, 42, 130, 148
....: , 53, 251, 190, 56, 141, 86, 233, 28, 87, 231, 235, 99, 227, 129, 179, 101, 118, 204, 185, 122, 158
....: , 174, 44, 22, 82, 21, 254, 144, 206, 177, 10, 14, 210, 152, 120, 151, 147, 54, 9, 66, 93, 114, 29,
....:  143, 201, 241, 142, 246, 124, 205, 199, 157, 34, 64, 6, 202, 200, 1, 140, 186, 138, 117, 219, 167,
....:  154, 239, 112, 249, 95, 116, 216, 162, 26, 159, 71, 51, 126, 88, 189, 247, 218, 146, 155, 221, 208
....: , 211, 39, 133, 229, 228, 191, 80, 85, 113, 35, 136, 156, 139, 108, 244, 48, 84, 176, 248])
sage: S.max_degree()
7
```

4. Điểm bất động

- Sử dụng hàm fixed_points()

- Trả về danh sách tất cả các điểm cố định của S-box. Trả về mảng rỗng tức là S-box không có điểm bất động

```
sage: from sage.crypto.sbox import SBox
sage: S = SBox([195, 131, 90, 220, 102, 255, 242, 105, 45, 128, 57, 32, 134, 110, 180, 240, 121, 70, 96,
....: 13, 58, 253, 182, 171, 132, 178, 193, 91, 187, 41, 49, 72, 181, 94, 212, 79, 226, 67, 78, 17, 111,
....: 27, 243, 60, 197, 184, 224, 135, 153, 46, 40, 234, 192, 188, 106, 75, 169, 209, 76, 5, 63, 55, 163,
....:  252, 183, 194, 62, 103, 61, 250, 89, 37, 125, 83, 149, 160, 203, 23, 198, 31, 100, 74, 236, 119, 1
....: 66, 0, 8, 196, 245, 38, 214, 33, 109, 12, 30, 115, 175, 7, 173, 15, 50, 170, 237, 47, 164, 68, 3, 6
....: 5, 230, 36, 104, 161, 59, 145, 215, 4, 43, 19, 137, 127, 25, 223, 73, 11, 77, 225, 24, 168, 98, 172
....: , 217, 238, 213, 123, 207, 20, 222, 107, 232, 92, 16, 97, 2, 150, 69, 81, 52, 18, 165, 42, 130, 148
....: , 53, 251, 190, 56, 141, 86, 233, 28, 87, 231, 235, 99, 227, 129, 179, 101, 118, 204, 185, 122, 158
....: , 174, 44, 22, 82, 21, 254, 144, 206, 177, 10, 14, 210, 152, 120, 151, 147, 54, 9, 66, 93, 114, 29,
....:  143, 201, 241, 142, 246, 124, 205, 199, 157, 34, 64, 6, 202, 200, 1, 140, 186, 138, 117, 219, 167,
....:  154, 239, 112, 249, 95, 116, 216, 162, 26, 159, 71, 51, 126, 88, 189, 247, 218, 146, 155, 221, 208
....: , 211, 39, 133, 229, 228, 191, 80, 85, 113, 35, 136, 156, 139, 108, 244, 48, 84, 176, 248])
sage: S.fixed_points()
[]
```

5. Giá trị xấp xỉ vi sai cực đại

- Sử dụng hàm maximal_difference_probability_absolute()

- Trả về xác suất chênh lệch của chênh lệch có xác suất cao nhất theo giá trị tuyệt đối, tức là tổng tần suất nó xảy ra.

- Tương tự, điều này tương đương với độ đồng nhất vi sai của S-Box này.

```
sage: from sage.crypto.sbox import SBox
sage: S = SBox([195, 131, 90, 220, 102, 255, 242, 105, 45, 128, 57, 32, 134, 110, 180, 240, 121, 70, 96,
....: 13, 58, 253, 182, 171, 132, 178, 193, 91, 187, 41, 49, 72, 181, 94, 212, 79, 226, 67, 78, 17, 111,
....: 27, 243, 60, 197, 184, 224, 135, 153, 46, 40, 234, 192, 188, 106, 75, 169, 209, 76, 5, 63, 55, 163,
....:  252, 183, 194, 62, 103, 61, 250, 89, 37, 125, 83, 149, 160, 203, 23, 198, 31, 100, 74, 236, 119, 1
....: 66, 0, 8, 196, 245, 38, 214, 33, 109, 12, 30, 115, 175, 7, 173, 15, 50, 170, 237, 47, 164, 68, 3, 6
....: 5, 230, 36, 104, 161, 59, 145, 215, 4, 43, 19, 137, 127, 25, 223, 73, 11, 77, 225, 24, 168, 98, 172
....: , 217, 238, 213, 123, 207, 20, 222, 107, 232, 92, 16, 97, 2, 150, 69, 81, 52, 18, 165, 42, 130, 148
....: , 53, 251, 190, 56, 141, 86, 233, 28, 87, 231, 235, 99, 227, 129, 179, 101, 118, 204, 185, 122, 158
....: , 174, 44, 22, 82, 21, 254, 144, 206, 177, 10, 14, 210, 152, 120, 151, 147, 54, 9, 66, 93, 114, 29,
....:  143, 201, 241, 142, 246, 124, 205, 199, 157, 34, 64, 6, 202, 200, 1, 140, 186, 138, 117, 219, 167,
....:  154, 239, 112, 249, 95, 116, 216, 162, 26, 159, 71, 51, 126, 88, 189, 247, 218, 146, 155, 221, 208
....: , 211, 39, 133, 229, 228, 191, 80, 85, 113, 35, 136, 156, 139, 108, 244, 48, 84, 176, 248])
sage: S.maximal_difference_probability_absolute()
12
```

6. Giá trị xấp xỉ tuyến tính cực đại

- Sử dụng hàm maximal_linear_bias_absolute()

- Trả về độ lệch tuyến tính tối đa, tức là tần suất xấp xỉ tuyến tính có độ lệch cao nhất

```
sage: from sage.crypto.sbox import SBox
sage: S = SBox([195, 131, 90, 220, 102, 255, 242, 105, 45, 128, 57, 32, 134, 110, 180, 240, 121, 70, 96,
....: 13, 58, 253, 182, 171, 132, 178, 193, 91, 187, 41, 49, 72, 181, 94, 212, 79, 226, 67, 78, 17, 111,
....: 27, 243, 60, 197, 184, 224, 135, 153, 46, 40, 234, 192, 188, 106, 75, 169, 209, 76, 5, 63, 55, 163,
....:  252, 183, 194, 62, 103, 61, 250, 89, 37, 125, 83, 149, 160, 203, 23, 198, 31, 100, 74, 236, 119, 1
....: 66, 0, 8, 196, 245, 38, 214, 33, 109, 12, 30, 115, 175, 7, 173, 15, 50, 170, 237, 47, 164, 68, 3, 6
....: 5, 230, 36, 104, 161, 59, 145, 215, 4, 43, 19, 137, 127, 25, 223, 73, 11, 77, 225, 24, 168, 98, 172
....: , 217, 238, 213, 123, 207, 20, 222, 107, 232, 92, 16, 97, 2, 150, 69, 81, 52, 18, 165, 42, 130, 148
....: , 53, 251, 190, 56, 141, 86, 233, 28, 87, 231, 235, 99, 227, 129, 179, 101, 118, 204, 185, 122, 158
....: , 174, 44, 22, 82, 21, 254, 144, 206, 177, 10, 14, 210, 152, 120, 151, 147, 54, 9, 66, 93, 114, 29,
....:  143, 201, 241, 142, 246, 124, 205, 199, 157, 34, 64, 6, 202, 200, 1, 140, 186, 138, 117, 219, 167,
....:  154, 239, 112, 249, 95, 116, 216, 162, 26, 159, 71, 51, 126, 88, 189, 247, 218, 146, 155, 221, 208
....: , 211, 39, 133, 229, 228, 191, 80, 85, 113, 35, 136, 156, 139, 108, 244, 48, 84, 176, 248])
sage: S.maximal_linear_bias_absolute()
34
```

PHỤ LỤC 4

Một số chương trình phân tích tính chất mật mã của hình ảnh mã hóa

1. Entropy analysis

```
clear
plaintext=imread('beach.png');
ciphertext1=imread('encrypted_image_default.png');
ciphertext2=imread('encrypted_image.png');

%Tách kênh màu R, G, B
[R,G,B] = imsplit(plaintext);
[R1,G1,B1] = imsplit(ciphertext1);
[R2,G2,B2] = imsplit(ciphertext2);

%Entropy toàn cục
ent_plaintext_R = entropy(R)
ent_plaintext_G = entropy(G)
ent_plaintext_B = entropy(B)

ent_encrypted1_R = entropy(R1)
ent_encrypted1_G = entropy(G1)
ent_encrypted1_B = entropy(B1)

ent_encrypted2_R = entropy(R2)
ent_encrypted2_G = entropy(G2)
ent_encrypted2_B = entropy(B2)

% Khởi tạo các biến lưu entropy cục bộ
entP_R = []; entP_G = []; entP_B = [];
entE1_R = []; entE1_G = []; entE1_B = [];
entE2_R = []; entE2_G = []; entE2_B = [];

[rows, cols] = size(R);  % Lấy kích thước ảnh
Mat = zeros(rows, cols); % Ma trận đánh dấu các ô đã chọn
counter = 0;

% Tính entropy cục bộ với 10 ô vuông không chồng lấn kích thước 44x44
while counter < 10
    % Xác định tọa độ góc dưới bên trái của ô vuông ngẫu nhiên
    localx = randi(rows - 44);
    localy = randi(cols - 44);

    % Kiểm tra xem ô vuông có chồng lấn không
    if sum(sum(Mat(localx:localx+44, localy:localy+44))) == 0
        counter = counter + 1;
        Mat(localx:localx+44, localy:localy+44) = 1;

        % Tính entropy cục bộ cho ảnh gốc (plaintext)
        entP_R = [entP_R, entropy(R(localx:localx+44, localy:localy+44))];
        entP_G = [entP_G, entropy(G(localx:localx+44, localy:localy+44))];
        entP_B = [entP_B, entropy(B(localx:localx+44, localy:localy+44))];

        % Tính entropy cục bộ cho ảnh mã hóa thứ nhất (ciphertext1)
```

```
        entE1_R = [entE1_R, entropy(R1(localx:localx+44, localy:localy+44))];
        entE1_G = [entE1_G, entropy(G1(localx:localx+44, localy:localy+44))];
        entE1_B = [entE1_B, entropy(B1(localx:localx+44, localy:localy+44))];

        % Tính entropy cục bộ cho ảnh mã hóa thứ hai (ciphertext2)
        entE2_R = [entE2_R, entropy(R2(localx:localx+44, localy:localy+44))];
        entE2_G = [entE2_G, entropy(G2(localx:localx+44, localy:localy+44))];
        entE2_B = [entE2_B, entropy(B2(localx:localx+44, localy:localy+44))];
    end
end

figure
imshow(uint8(255*Mat))

% Tính giá trị trung bình entropy cục bộ cho từng kênh
local_ent_P_R = mean(entP_R);
local_ent_P_G = mean(entP_G);
local_ent_P_B = mean(entP_B);

local_ent_E1_R = mean(entE1_R);
local_ent_E1_G = mean(entE1_G);
local_ent_E1_B = mean(entE1_B);

local_ent_E2_R = mean(entE2_R);
local_ent_E2_G = mean(entE2_G);
local_ent_E2_B = mean(entE2_B);

% Hiển thị kết quả
fprintf('Entropy cục bộ trung bình của ảnh gốc:\n');
fprintf('R: %.4f, G: %.4f, B: %.4f\n', local_ent_P_R, local_ent_P_G, local_ent_P_B);

fprintf('Entropy cục bộ trung bình của ảnh mã hóa thứ nhất:\n');
fprintf('R: %.4f, G: %.4f, B: %.4f\n', local_ent_E1_R, local_ent_E1_G,
local_ent_E1_B);

fprintf('Entropy cục bộ trung bình của ảnh mã hóa thứ hai:\n');
fprintf('R: %.4f, G: %.4f, B: %.4f\n', local_ent_E2_R, local_ent_E2_G,
local_ent_E2_B);
```

2. Histogram analysis

```
clear
plaintext = imread('beach.png');
ciphertext1 = imread('encrypted_image_default.png');
ciphertext2 = imread('encrypted_image_other.png');


%Tách kênh màu RGB
[R, G, B] = imsplit(plaintext);
[R1, G1, B1] = imsplit(ciphertext1);
[R2, G2, B2] = imsplit(ciphertext2);

figure
```

```
subplot(3,3,1)
hold all
histogram(R, 256, 'FaceColor', "r", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
set(gca, 'fontweight', 'bold')
title('Plaintext Image')
box on
subplot(3,3,2)
hold all
histogram(G, 256, 'FaceColor', "g", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
set(gca, 'fontweight', 'bold')
title('Plaintext Image')
box on
subplot(3,3,3)
hold all
histogram(B, 256, 'FaceColor', "b", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
set(gca, 'fontweight', 'bold')
title('Plaintext Image')
box on

subplot(3,3,4)
hold all
histogram(R1, 256, 'FaceColor', "r", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
set(gca, 'fontweight', 'bold')
title('Encrypted Image1')
box on
subplot(3,3,5)
hold all
histogram(G1, 256, 'FaceColor', "g", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
set(gca, 'fontweight', 'bold')
title('Encrypted Image1')
box on
subplot(3,3,6)
hold all
histogram(B1, 256, 'FaceColor', "b", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
set(gca, 'fontweight', 'bold')
title('Encrypted Image1')
box on

subplot(3,3,7)
hold all
histogram(R2, 256, 'FaceColor', "r", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
set(gca, 'fontweight', 'bold')
title('Encrypted Image2')
box on
subplot(3,3,8)
hold all
histogram(G2, 256, 'FaceColor', "g", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
```

```
set(gca, 'fontweight', 'bold')
title('Encrypted Image2')
box on
subplot(3,3,9)
hold all
histogram(B2, 256, 'FaceColor', "b", 'EdgeColor', "none")
set(gca, 'fontsize', 10)
set(gca, 'fontweight', 'bold')
title('Encrypted Image2')
box on

% Tính phương sai của histogram
counts = imhist(R);
disp('Variance: Plaintext R')
var(counts)
counts = imhist(G);
disp('Variance: Plaintext G')
var(counts)
counts = imhist(B);
disp('Variance: Plaintext B')
var(counts)

counts = imhist(R1);
disp('Variance: Encrypted1 R')
var(counts)
counts = imhist(G1);
disp('Variance: Encrypted1 G')
var(counts)
counts = imhist(B1);
disp('Variance: Encrypted1 B')
var(counts)

counts = imhist(R2);
disp('Variance: Encrypted1 R')
var(counts)
counts = imhist(G2);
disp('Variance: Encrypted1 G')
var(counts)
counts = imhist(B2);
disp('Variance: Encrypted1 B')
var(counts)
```

3. Correlation Analysis

```
clear
plaintext = imread('beach.png');
ciphertext1 = imread('encrypted_image_default.png');
ciphertext2 = imread('encrypted_image_other.png');

% Hiển thị histogram của các kênh RGB
[R, G, B] = imsplit(plaintext);
[R1, G1, B1] = imsplit(ciphertext1);
[R2, G2, B2] = imsplit(ciphertext2);
% Tính toán cho mỗi kênh
```

```
%RED
%horizontal
c_horz_o = corrcoef(double(R(:, 1:end-1)), double(R(:, 2:end)))
c_horz_e1 = corrcoef(double(R1(:, 1:end-1)), double(R1(:, 2:end)))
c_horz_e2 = corrcoef(double(R2(:, 1:end-1)), double(R2(:, 2:end)))
%vertical
c_vert_o = corrcoef(double(R(1:end-1, :)), double(R(2:end, :)))
c_vert_e1 = corrcoef(double(R1(1:end-1, :)), double(R1(2:end, :)))
c_vert_e2 = corrcoef(double(R2(1:end-1, :)), double(R2(2:end, :)))
%diagonal
c_diag_o = corrcoef(double(R(1:end-1, 1:end-1)), double(R(2:end, 2:end)))
c_diag_e1 = corrcoef(double(R1(1:end-1, 1:end-1)), double(R1(2:end, 2:end)))
c_diag_e2 = corrcoef(double(R2(1:end-1, 1:end-1)), double(R2(2:end, 2:end)))
%Green
%horizontal
c_horz_o = corrcoef(double(G(:, 1:end-1)), double(G(:, 2:end)))
c_horz_e1 = corrcoef(double(G1(:, 1:end-1)), double(G1(:, 2:end)))
c_horz_e2 = corrcoef(double(G2(:, 1:end-1)), double(G2(:, 2:end)))
%vertical
c_vert_o = corrcoef(double(G(1:end-1, :)), double(G(2:end, :)))
c_vert_e1 = corrcoef(double(G1(1:end-1, :)), double(G1(2:end, :)))
c_vert_e2 = corrcoef(double(G2(1:end-1, :)), double(G2(2:end, :)))
%diagonal
c_diag_o = corrcoef(double(G(1:end-1, 1:end-1)), double(G(2:end, 2:end)))
c_diag_e1 = corrcoef(double(G1(1:end-1, 1:end-1)), double(G1(2:end, 2:end)))
c_diag_e2 = corrcoef(double(G2(1:end-1, 1:end-1)), double(G2(2:end, 2:end)))
%Blue
%horizontal
c_horz_o = corrcoef(double(B(:, 1:end-1)), double(B(:, 2:end)))
c_horz_e1 = corrcoef(double(B1(:, 1:end-1)), double(B1(:, 2:end)))
c_horz_e2 = corrcoef(double(B2(:, 1:end-1)), double(B2(:, 2:end)))
%vertical
c_vert_o = corrcoef(double(B(1:end-1, :)), double(B(2:end, :)))
c_vert_e1 = corrcoef(double(B1(1:end-1, :)), double(B1(2:end, :)))
c_vert_e2 = corrcoef(double(B2(1:end-1, :)), double(B2(2:end, :)))
%diagonal
c_diag_o = corrcoef(double(B(1:end-1, 1:end-1)), double(B(2:end, 2:end)))
c_diag_e1 = corrcoef(double(B1(1:end-1, 1:end-1)), double(B1(2:end, 2:end)))
c_diag_e2 = corrcoef(double(B2(1:end-1, 1:end-1)), double(B2(2:end, 2:end)))
% Phân tích độ tương quan của từng kênh màu
subplot(3,3,1)
scatter(double(R(1:end-1, 1:end-1)), double(R(2:end, 2:end)),'.r')
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Plaintext Image (R)')
subplot(3,3,2)
scatter(double(R1(1:end-1, 1:end-1)), double(R1(2:end, 2:end)),'.r');
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Encypted1 (R)')
subplot(3,3,3)
```

```
scatter(double(R1(1:end-1, 1:end-1)), double(R1(2:end, 2:end)),'.r');
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Encrypted2 (R)')
subplot(3,3,4)
scatter(double(G(1:end-1, 1:end-1)), double(G(2:end, 2:end)),'.g')
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Plaintext Image (G)')
subplot(3,3,5)
scatter(double(G1(1:end-1, 1:end-1)), double(G1(2:end, 2:end)),'.g');
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Encrypted1 (G)')
subplot(3,3,6)
scatter(double(G2(1:end-1, 1:end-1)), double(G2(2:end, 2:end)),'.g');
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Encrypted2 (G)')
subplot(3,3,7)
scatter(double(B(1:end-1, 1:end-1)), double(B(2:end, 2:end)),'.b')
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Plaintext Image (B)')
subplot(3,3,8)
scatter(double(B1(1:end-1, 1:end-1)), double(B1(2:end, 2:end)),'.b');
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Encrypted1 (B)')
subplot(3,3,9)
scatter(double(B2(1:end-1, 1:end-1)), double(B2(2:end, 2:end)),'.b');
set(gca,'fontsize',10)
set(gca,'fontweight','bold')
axis([0,256,0,256])
box
title('Encrypted2 (B)')
```