|  |
| --- |
| HỌC VIỆN KỸ THUẬT MẬT MÃ  **KHOA CÔNG NGHỆ THÔNG TIN**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| BÀI TẬP MÔN HỌC CƠ SỞ AN TOÀN VÀ BẢO MẬT THÔNG TIN  **VIẾT CHƯƠNG TRÌNH MÃ HÓA VÀ GIẢI MÃ BẰNG MẬT MÃ AES** |
| Ngành: Công nghệ thông tin  Chuyên ngành: Kỹ thuật phần mềm nhúng và di động  Sinh viên thực hiện: Đoàn Thị Thanh Lam - CT030133  Trần Anh Tuấn – CT030355  *Nhóm 9*  Người hướng dẫn:  **TS. Nguyễn Đào Trường**  Khoa Công nghệ thông tin – Học viện Kỹ thuật mật mã  Hà Nội, 2021 |

**NHẬN XÉT VÀ CHO ĐIỂM CỦA GIÁO VIÊN**

# MỤC LỤC

[MỤC LỤC 2](#_Toc89629370)

[DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT 4](#_Toc89629371)

[DANH MỤC HÌNH ẢNH 5](#_Toc89629372)

[LỜI NÓI ĐẦU 6](#_Toc89629373)

[Chương 1. CƠ SỞ LÝ THUYẾT 1](#_Toc89629374)

[1.1. Tổng quan. 1](#_Toc89629375)

[1.1.1. An toàn bảo mâṭ thông tin và mâṭ mã học 1](#_Toc89629376)

[1.1.2. Giới thiệu về mật mã khối 2](#_Toc89629377)

[1.2. Xác định yêu cầu bài toán 2](#_Toc89629378)

[1.2.1. Xác định yêu cầu 2](#_Toc89629379)

[1.2.2. Ngôn ngữ lập trình 3](#_Toc89629380)

[Chương 2. MẬT MÃ AES (Advanced Encryption Standard) 4](#_Toc89629381)

[2.1. Giới thiệu thuật toán AES. 4](#_Toc89629382)

[2.1.1. Mô tả thuật toán AES 4](#_Toc89629383)

[2.1.2. Trạng thái (State) 6](#_Toc89629384)

[2.1.3. Bảng liệt kê khóa 6](#_Toc89629385)

[2.2. Mã hóa 7](#_Toc89629386)

[2.2.1. Hàm AddRoundKey () 7](#_Toc89629387)

[2.2.2. Hàm SubBytes () 8](#_Toc89629388)

[2.2.3. Hàm ShiftRow () 9](#_Toc89629389)

[2.2.4. Hàm MixColumns () 10](#_Toc89629390)

[2.2.5. Hàm Encrypt () 11](#_Toc89629391)

[2.2.6. Thời gian mã hóa 12](#_Toc89629392)

[2.3. Giải mã 12](#_Toc89629393)

[2.3.1. Hàm Decrypt () 12](#_Toc89629394)

[2.3.2. Hàm InvCipher () 13](#_Toc89629395)

[2.3.3. Hàm InvShiftRows () 14](#_Toc89629396)

[2.3.4. Hàm InvSubBytes () 14](#_Toc89629397)

[2.3.5. Hàm InvMixColumns () 15](#_Toc89629398)

[2.3.6. Thời gian giải mã 16](#_Toc89629399)

[2.4. Các dạng tấn công AES và phương pháp phòng chống 16](#_Toc89629400)

[2.4.1. Side-channel attack 16](#_Toc89629401)

[2.4.2. Known attacks 17](#_Toc89629402)

[2.4.3. Các phương pháp phòng chống 17](#_Toc89629403)

[Chương 3. KẾT QUẢ THỰC NGHIỆM 18](#_Toc89629404)

[3.1. Mã hóa AES EBC khóa 128 bit 18](#_Toc89629405)

[3.2. Mã hóa AES EBC khóa 192 bit 19](#_Toc89629406)

[3.3. Mã hóa AES EBC khóa 256 bit 20](#_Toc89629407)

[3.4. Một số phần mở rộng 21](#_Toc89629408)

[3.4.1. Tích hợp thêm bộ mã hoá 21](#_Toc89629409)

[3.4.2. Tích hợp thêm các chế độ mã hoá khác của AES 22](#_Toc89629410)

[3.5. Nhận xét và đánh giá 26](#_Toc89629411)

[PHỤ LỤC 27](#_Toc89629412)

[KẾT LUẬN 28](#_Toc89629413)

[TÀI LIỆU THAM KHẢO 29](#_Toc89629414)

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| AES | Chuẩn mã hóa cao cấp |
| Bit | Một số nhi ̣phân nhân giá tri ̣0 hoặc 1 |
| Byte | Một nhóm 8 bit |
| Block | Một dãy các bit nhi ̣phân tạo thành input, output, trạng thái (state) và các khóa sử dụng tại các vòng lặp (Round Key) của hệ mã. Độ dài của dãy (khối) là số lượng các bit mà nó chứa. Các khối cũng có thể được xem là môṭ dãy các byte |
| Cipher | Thuâṭ toán mã hóa |
| Cipher Key | Khóa của hệ mã, có thể được biểu diễn dưới dạng một mảng 2 chiều gồm 4 hàng và Nk cột |
| Ciphertext | Bản mã |
| Round Key | Là các giá trị sinh ra từ khóa chính bằng cách sử dụng thủ tục sinh khóa. Các khóa này được sử dụng tại các vòng lặp của thuật toán |
| Trạng thái (State) | Các giá trị mã hóa trung gian có thể biểu diễn dưới dạng một mảng 2 chiều gồm 4 hàng và Nb cột |
| S-box | Môṭ bảng thế phi tuyến được sử dụng trong thủ tục sinh khóa và trong các biến đổi thay thế các byte để thực hiêṇ các thay thế 1-1 đối với môṭ giá tri ̣1 byte |
| Word | Một nhóm 32bit có thể được xem như 1 đơn vi ̣tính toán độc lâp̣ hoặc là môṭ mảng 4 byte |

DANH MỤC HÌNH ẢNH

[Hình 2‑1 Thuật toán mã hóa và giải mã của AES 5](#_Toc89629417)

[Hình 2‑2 Các trạng thái của AES 6](#_Toc89629418)

[Hình 2‑3 Hàm AddRoundKey () của AES 7](#_Toc89629419)

[Hình 2‑4 Code hàm AddRoundKey () 8](#_Toc89629420)

[Hình 2‑5 Bảng thế S-box của AES 9](#_Toc89629421)

[Hình 2‑6 Code hàm SubBytes () 9](#_Toc89629422)

[Hình 2‑7 Hàm ShifRows () của AES 10](#_Toc89629423)

[Hình 2‑8 Code hàm ShiftRow () 10](#_Toc89629424)

[Hình 2‑9 Hàm MixColumns () của AES 11](#_Toc89629425)

[Hình 2‑10 Code hàm MixColumns () 11](#_Toc89629426)

[Hình 2‑11 Code hàm Encrypt 12](#_Toc89629427)

[Hình 2‑12 Code Hàm Decrypt () 13](#_Toc89629428)

[Hình 2‑13 Code hàm InvCipher () 13](#_Toc89629429)

[Hình 2‑14 Hàm InvShiftRows () của AES 14](#_Toc89629430)

[Hình 2‑15 Code *h*àm InvShiftRows () 14](#_Toc89629431)

[Hình 2‑16 Bảng thế cho hàm InvSubBytes () 15](#_Toc89629432)

[Hình 2‑17 Code hàm InvSubBytes () 15](#_Toc89629433)

[Hình 2‑18 Code hàm InvMixColumns () 16](#_Toc89629434)

[Hình 3‑1 Kết quả mã hóa khóa 128bit 18](#_Toc89629435)

[Hình 3‑2 Kết quả giải mã khóa 128bit 19](#_Toc89629436)

[Hình 3‑3 Kết quả mã hóa khóa 192bit 19](#_Toc89629437)

[Hình 3‑4 Kết quả giải mã khóa 192bit 20](#_Toc89629438)

[Hình 3‑5 Kết quả mã hóa khóa 256bit 20](#_Toc89629439)

[Hình 3‑6 Kết quả giải mã khóa 256bit 21](#_Toc89629440)

[Hình 3‑7 Mã hoá AES EBC 128bit tích hợp bộ mã UTF8 22](#_Toc89629441)

[Hình 3‑8 Cách thức hoạt động mã hoá AES CBC 22](#_Toc89629442)

[Hình 3‑9 Cách giải mã AES CBC 23](#_Toc89629443)

[Hình 3‑10 Mã hoá AES PCBC 23](#_Toc89629444)

[Hình 3‑11Giải mã AES PCBC 23](#_Toc89629445)

[Hình 3‑12 Mã hoá AES CFB 24](#_Toc89629446)

[Hình 3‑13 Giải mã AES CFB 24](#_Toc89629447)

[Hình 3‑14 Mã hoá AES OFB 24](#_Toc89629448)

[Hình 3‑15 Giải mã AES OFB 25](#_Toc89629449)

[Hình 3‑16 Mã hoá AES CTR 25](#_Toc89629450)

[Hình 3‑17 Giải mã AES CTR 25](#_Toc89629451)

[Hình 3‑18 Mã hoá AES CBC 128bit với bộ mã utf-8 26](#_Toc89629452)

LỜI NÓI ĐẦU

Ngày nay, thông tin là một bộ phần quan trọng và là tài sản có thuộc quyền sở hữu của các tổ chức. Chúng ta cần phải quan tâm tới việc làm thế nào để đảm bảo an toàn bí mật thông tin cho các tài liệu, văn bản quan trọng, đặc biệt là trong lĩnh vực quân sự, ngoại giao. Với sự xuất hiện của máy tính, các tài liệu văn bản giấy tờ và các thông tin quan trọng đều được số hóa và xử lý trên máy tính, được truyền đi trong một môi trường mà mặc định là không an toàn. Do đó yêu cầu về việc có một cơ chế, giải pháp để bảo vệ sự an toàn và bí mật của các thông tin nhạy cảm, quan trọng ngày càng trở nên cấp thiết. Sự thiệt hại và lạm dụng thông tin không chỉ ảnh hưởng đến người sử dụng hoặc các ứng dụng mà nó còn gây ra các hậu quả tai hại cho toàn bộ tổ chức đó. Thêm vào đó sự ra đời của Internet đã giúp cho việc truy cập thông tin ngày càng trở nên dễ dàng hơn.

Mật mã học chính là ngành khoa học đảm bảo cho sự an toàn và bí mật của thông tin. Tài liệu này dựa trên quá trình học tập và nghiên cứu môn học Cơ sở an toàn và bảo mật thông tin, nhóm em xây dựng đề tài Viết chương trình mã hóa và giải mã bằng mật mã AES “Mã hóa một đoạn dữ liệu có sẵn trong một biến của chương trình, cho biết thời gian mã hóa và thời gian giải mã”. Đề tài được chia thành ba chương chính:

Chương 1: Cơ sở lý thuyết. Chương này giới thiệu về An toàn bảo mật thông tin và mật mã học, vấn đề đặt ra và yêu cầu của đề tài.

Chương 2: Hệ mã AES. Chương này trình bày chi tiết toàn bộ thuật toán mã hóa và giải mã chuẩn mã hóa cao cấp AES.

Chương 3: Kết quả thực nghiệm. Chương này mô tả về kết quả mã hóa và giải mã AES, đưa ra nhận xét và đánh giá về thực nghiệm mà nhóm thực hiện.

# CƠ SỞ LÝ THUYẾT

Chương này trình bày toàn bộ những kiến thức phần lý thuyết liên quan tới nội dung thực hành, bài tập của chủ đề.

## Tổng quan.

### An toàn bảo mâṭ thông tin và mâṭ mã học

Trải qua nhiều thế kỷ hàng loạt các giao thức (protocol) và các cơ chế (mechanism) đã được taọ ra để đáp ứng nhu cầu an toàn bảo mâṭ thông tin khi mà nó được truyền tải trên các phương tiệṇ vâṭ lý (giấy, sách, báo …). Thường thì các muc̣ tiêu của an toàn bảo mâṭ thông tin không thể đaṭ được nếu chỉ đơn thuần dựa vào các thuậṭ toán toán hoc̣ và các giao thức, mà để đạt được điều này đòi hỏi cần có các kỹ thuật mang tính thủ tục và sự tôn troṇg các điều luâṭ. Chẳng haṇ sự bí mâṭ của các bức thư tay là do sự phân phát các lá thư đã có đóng dấu bởi một dịch vụ thư tín đã được chấp nhận. Tính an toàn về măṭ vâṭ lý của các lá thư là haṇ chế (nó có thể bị xem trộm) nên để đảm bảo sự bí mật của bức thư pháp luật đã đưa ra qui định: viêc̣ xem thư mà không được sự đồng ý của chủ nhân hoặc những người có thẩm quyền là phạm pháp và sẽ bị trừng phạt.

Về măṭ ý tưởng viêc̣ lưu giữ thông tin là không có nhiều thay đổi đáng kể qua thời gian. Ngày xưa thông tin thường được lưu và vận chuyển trên giấy tờ, trong khi giờ đây chúng được lưu dưới dạng số hóa và được vận chuyển bằng các hệ thống viễn thông hoăc̣ các hệ thống không dây. Tuy nhiên sự thay đổi đáng kể ở đây chính là khả năng sao chép và thay đổi thông tin. Người ta có thể tạo ra hàng ngàn mẩu tin giống nhau và không thể phân biệt được nó với bản gốc. Với các tài liêụ lưu trữ và vâṇ chuyển trên giấy điều này khó khăn hơn nhiều. Và điều cần thiết đối với một xã hội mà thông tin hầu hết được lưu trữ và vâṇ chuyển trên các phương tiêṇ điêṇ tử chính là các phương tiêṇ đảm bảo an toàn bảo mâṭ thông tin đôc̣ lâp̣ với các phương tiêṇ lưu trữ và vận chuyển vật lý của nó. Phương tiêṇ đó chính là mâṭ mã hoc̣, môṭ ngành khoa hoc̣ có lic̣h sử lâu đời dựa trên nền tảng các thuâṭ toán toán hoc̣, số hoc̣, xác suất và các môn khoa học khác.

### Giới thiệu về mật mã khối

Các hệ mã cổ điển đều có đặc điểm chung là từng ký tự của bản rõ được mã hoá tách biệt. Điều này làm cho việc phá mã trở nên 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 hoá cùng một lúc như là một đơn vị mã hoá đồng nhất. Trong kiểu mã hoá này, các tham số quan trọng là kích thước (độ dài) của mỗi khối và kích thước khóa.

Điều kiện để mã hoá khối an toàn:

* Kích thước khối phải đủ lớn để chống lại phương án tấn công bằng phương pháp thống kê. Tuy nhiên điều này sẽ dẫn đến thời gian mã hoá sẽ tăng lên.
* Không gian khoá, tức chiều dài khoá phải đủ lớn để chống lại phương án tấn công bằng vét cạn. Tuy nhiên khoá phải đủ ngắn để việc tạo khoá, phân phối và lưu trữ khoá được dễ dàng.

Khi thiết kế một hệ mã khối, phải đảm bảo hai yêu cầu sau:

* Sự hỗn loạn (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 phá 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.

Đặc điểm chung của các hệ mã khối là quá trình mã hóa làm việc với các khối dữ liệu (thường ở dạng xâu bit) có kích thước khác nhau (tối thiếu là 64 bit), khóa của hệ mã cũng là một xâu bit có độ dài cố định (56 bit với DES, các hệ mã khác là 128, 256, hoặc thậm chí 512 bit). Tất cả các hệ mã này đều dựa trên lý thuyết của Shannon đưa ra năm 1949 và nếu mang mã hóa hai bản rõ giống nhau sẽ thu được cùng một bản mã. Hoạt động của các hệ mã khối thường được thực hiện qua một số lần lặp, mỗi lần sẽ sử dụng một khóa con được sinh ra từ khóa chính.

## Xác định yêu cầu bài toán

### Xác định yêu cầu

Viết chương trình mã hóa và giải mã bằng mật mã AES:

Mã hóa một đoạn dữ liệu có sẵn trong một biến của chương trình, cho biết thời gian mã hóa và thời gian giải mã.

Input: một đoạn dữ liệu có sẵn trong một biến của chương trình.

Output: đoạn dữ liệu đã được mã hóa, thời gian mã hóa và thời gian giải mã.

### Ngôn ngữ lập trình

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình và là ngôn ngữ lập trình dễ học; được dùng rộng rãi trong phát triển trí tuệ nhân tạo. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

Python không có nhiều trong thư viện tiêu chuẩn liên quan đến mã hóa. Thay vào đó, chúng có một số thư viện phổ biến như hàm băm, ... Chúng ta sẽ xem xét ngắn gọn về chương này, nhưng trọng tâm chính sẽ là mã hóa và giải mã chuẩn mã hóa cao cấp AES. Chúng ta sẽ học cách mã hóa và giải mã chuỗi với mật mã này.

# MẬT MÃ AES (Advanced Encryption Standard)

AES (viết tắt của từ tiếng anh: Advanced Encryption Standard, hay Tiêu chuẩn mã hóa nâng cao) là một thuật toán mã hóa khối được chính phủ Hoa Kỳ áp dụng làm tiêu chuẩn mã hóa.

Thuật toán được xây dựng dựa trên Rijndael Cipher phát triển bởi 2 nhà mật mã học người Bỉ: Joan Daemen và Vincent Rijmen. AES làm việc với các khối dữ liệu 128bit và độ dài khóa 128bit, 192bit hoặc 256bit. Các khóa mở rộng sử dụng trong chu trình được tạo ra bởi thủ tục sinh khóa Rijndael. Hầu hết các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn của các byte. Mỗi khối dữ liệu đầu vào 128bit được chia thành 16byte, có thể xếp thành 4 cột, mỗi cột 4 phần tử hay một ma trận 4x4 của các byte, nó gọi là ma trận trạng thái.

Tùy thuộc vào độ dài của khóa khi sử dụng 128bit, 192bit hay 256bit mà thuật toán được thực hiện với số lần lặp khác nhau.

Tiêu chuẩn mã hóa nâng cao (AES), còn được gọi là Rijndael, là một thuật toán mật mã khối đã được phân tích và hiện đang được sử dụng rộng rãi. AES là một mật mã khối đối xứng nhằm thay thế DES làm tiêu chuẩn đã được phê duyệt cho một loạt các ứng dụng. Rijndael rất an toàn và không có điểm yếu nào được biết đến.

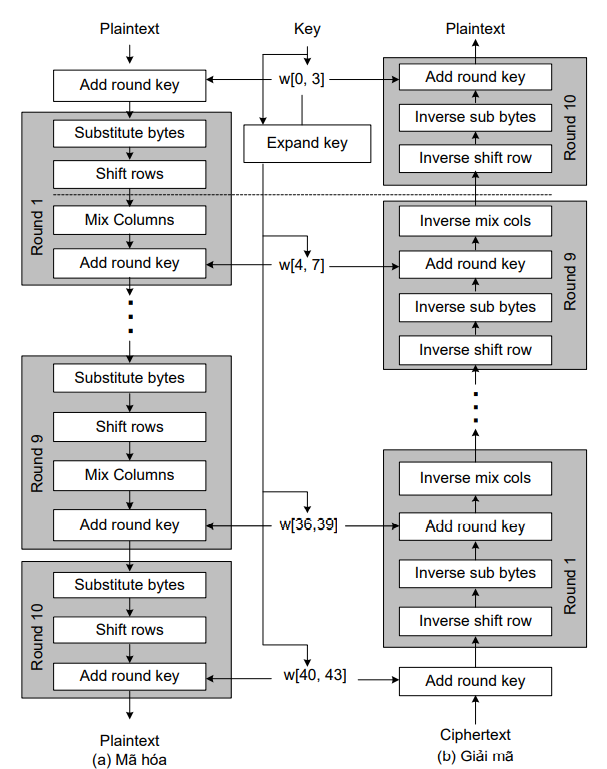
## Giới thiệu thuật toán AES.

### Mô tả thuật toán AES

Bắt đầu thuâṭ toán bản rõ (input) được copy vào mảng trạng thái. Sau khi cộng với khóa Round Key khởi tạo mảng trạng thái được biến đổi bằng các thực hiện một hàm vòng (round function) Nr lần (10, 12, hoăc̣ 14 phụ thuộc vào độ dài khóa) trong đó lần cuối cùng thực hiện khác các lần trước đó. Trạng thái sau lần lặp cuối cùng sẽ được chuyển thành output của thuật toán.

Hàm vòng được tham số hóa sử dụng một (key schedule) dãy các khóa được biểu diễn như là môṭ mảng 1 chiều của các word 4-byte được sinh ra từ thủ tuc̣ sinh khóa (Key Expansion).

Tất cả các vòng đều thực hiện các công việc giống nhau dựa trên 4 hàm (theo thứ tự ) SubBytes(), ShiftRows(), MixColumns() và AddRoundKey() trừ vòng cuối cùng bỏ qua việc thực hiện hàm MixColumns().

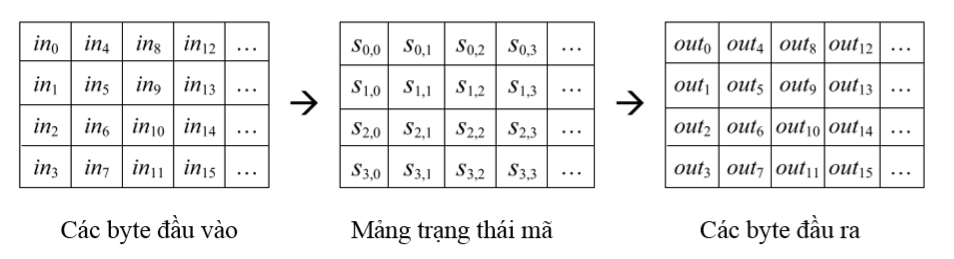


Hình 2‑1 Thuật toán mã hóa và giải mã của AES

### Trạng thái (State)

Các thao tác bên trong của AES được thực hiện trên một mảng 2 chiều các byte được gọi là trang thái. Một trạng thái gồm bốn hàng các byte, mỗi hàng có Nb byte trong đó Nb là kích thước của khối chia cho 32. Mảng trạng thái ký hiệu là s trong đó mỗi byte của mảng có 2 chỉ số hàng r và cột c (0 ≤ r, c < 4).

Tại thời điểm bắt đầu input của thuật toán – mảng các byte in 0, in1, …, in15 được copy vào mảng trạng thái theo qui tắc được minh hoạ bằng hình vẽ:



Hình 2‑2 Các trạng thái của AES

trong đó các giá tri ̣của mảng s và mảng output được tính như sau:

s[r, c] = in[r + 4c] ∀ 0 ≤ r, c < 4

out[r + 4c] = s[r, c] ∀ 0 ≤ r, c < 4

### Bảng liệt kê khóa

Các khóa vòng được phát sinh từ khóa mã bằng bảng liệt kê khóa. Thao tác này bao gồm 2 bước chính: mở rộng khóa và chọn khóa vòng. Nguyên tắc cơ bản được thể hiện như sau:

* Tổng số bit khóa vòng bằng độ dài khối nhân với số vòng cộng thêm 1 (ví dụ: với độ dài khối là 128 và có 10 vòng thì cần 1408bit khóa vòng).
* Khóa mã được mở rộng thành một khóa mở rộng.

Các khóa vòng được lấy từ khóa mở rộng trên theo cách sau: khóa vòng đầu tiên gồm có Nb từ đầu tiên, khóa vòng thứ hai gồm Nb từ tiếp theo và cứ tiếp tục cho đến hết.

Mở rộng khóa

* Khóa mở rộng là một mảng tuyến tính từ các từ 4byte và được kí hiệu là:

W [ Nb \* ( Nr +1)]

* Nk từ đầu tiên chứa khóa mã. Tất cả các từ khác được định nghĩa đệ quy theo các từ có chỉ số nhỏ hơn. Hàm mở rộng khóa phụ thuộc vào giá trị Nk.

Các hằng số vòng độc lập với Nk và được định nghĩa là:

Rcon[i] = (RC[i], ‘00’, ‘00’, ‘00’)

Trong đó:

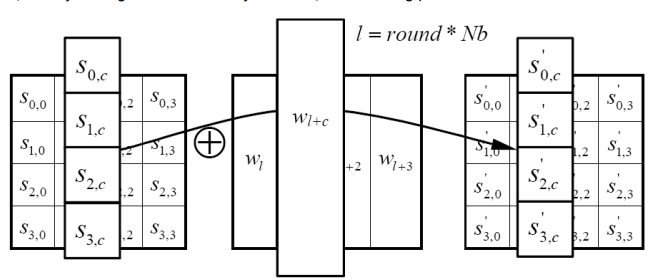
* RC[i] đại diện cho một phần tử trong GF (28) với giá trị x(i-1) sao cho:
* RC [1] = 1
* RC[i] = x (RC [i 1]) = x (i 1)

Sau đây sẽ đi vào chi tiết việc định nghĩa “trạng thái” cũng như nội dung các phép toán được sử dụng trong thuật toán AES.

## Mã hóa

### Hàm AddRoundKey ()

Trong hàm này một khóa vòng (Round Key) sẽ được cộng vào mảng trạng thái bằng một thao tác XOR bit. Mỗi khóa vòng gồm Nb word được sinh ra bởi thủ tục sinh khóa. Các word này sẽ được cộng vào mỗi cột của mảng trạng thái.



Hình 2‑3 Hàm AddRoundKey () của AES

Code hàm AddRoundKey () bằng Python:



Hình 2‑4 Code hàm AddRoundKey ()

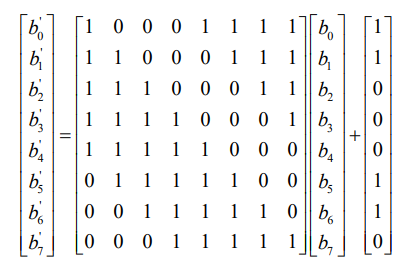
### Hàm SubBytes ()

Hàm SubBytes () thực hiện phép thay thế các byte của mảng trạng thái bằng cách sử dụng môṭ bảng thế S -box, bảng thế này là khả nghịch và được xây dựng bằng cách kết hợp hai biến đối sau:

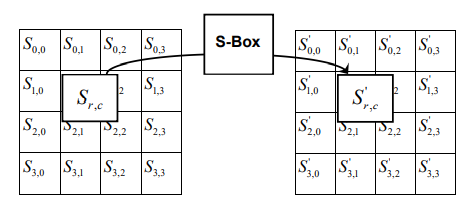
- Nhân nghịch đảo trên trường hữn haṇ GF(28), phần tử {00} được ánh xạ thành chính nó

- Áp dụng biến đổi Affine sau (trên GF(2))

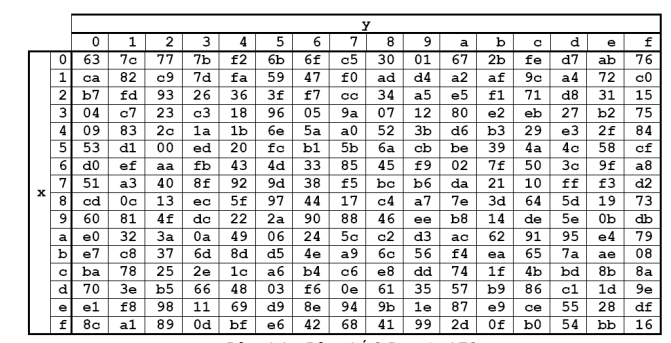
Các phần tử biến đổi affine của S-box có thể được biểu diến dưới dạng ma trận như sau:



Hình sau minh họa kết quả của việc áp dụng hàm biến đổi SubBytes () đối với mảng trạng thái:

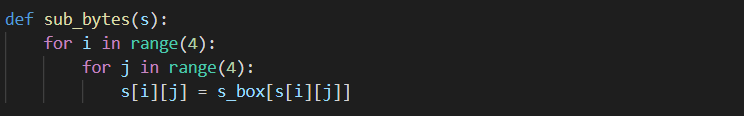


Bảng thế S -box được sử dụng trong hàm SubBytes () có thể được biểu diễn dưới dạng hexa như sau:



Hình 2‑5 Bảng thế S-box của AES

Code hàm SubBytes () bằng Python:



Hình 2‑6 Code hàm SubBytes ()

### Hàm ShiftRow ()

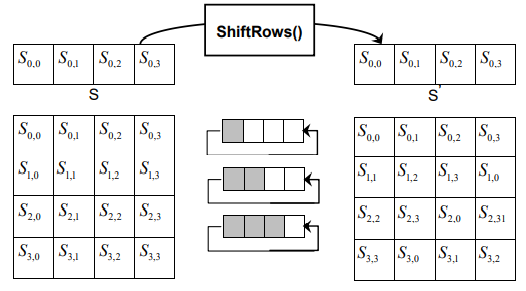
Trong hàm này các byte trong 3 hàng cuối của mảng trạng thái sẽ được dịch vòng với số lần dịch (hay số byte bi ̣dịch) khác nhau. Hàng đầu tiên r = 0 không bi ̣dịch. Cụ thể hàm này sẽ tiến hành biến đổi sau:

**S’r,c = Sr,(c + shift ( r, Nb)) mod Nb (Nb = 4)**

trong đó giá tri ̣dịch shift (r, Nb) phụ thuôc̣ vào số hàng r như sau:

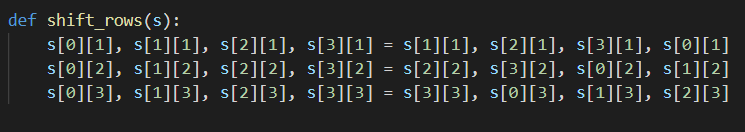
shift(1, 4) = 1, shift(2, 4) = 2, shift(3, 4) = 3.

Thao tác này sẽ chuyển các byte tới các vi ̣trí thấp hơn trong các hàng , trong khi các byte thấp nhất sẽ được chuyển lên đầu của hàng. Tất các các mô tả trên có thể minh họa qua hình vẽ sau:



Hình 2‑7 Hàm ShifRows () của AES

Code hàm ShiftRow () bằng Python:



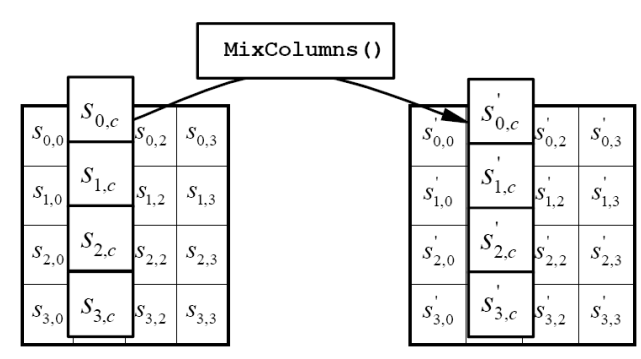
Hình 2‑8 Code hàm ShiftRow ()

### Hàm MixColumns ()

Hàm này làm việc trên các cột của bảng trạng thái, nó coi mỗi cột của mảng trạng thái như là một đa thức gồm 4 hạng tử. Các cột sẽ được xem như là các đa thức trên GF(28 ) và được nhân theo modulo x4 + 1 với một đa thức cố định a(x):

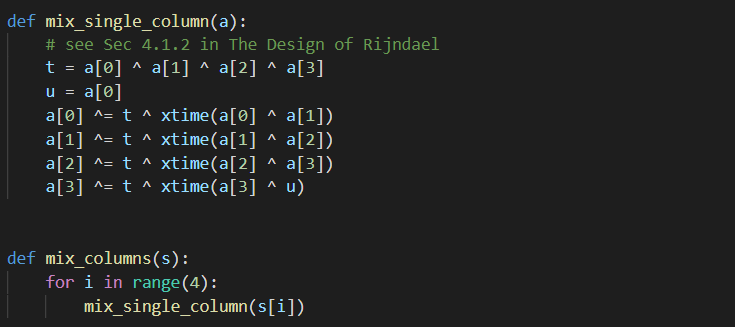
a(x) = {03}x3 + {01}x2 + {01}x + {02}

Có thể minh họa việc thực hiện của hàm này bằng hình vẽ sau:



Hình 2‑9 Hàm MixColumns () của AES

Code hàm MixColumns () bằng Python:

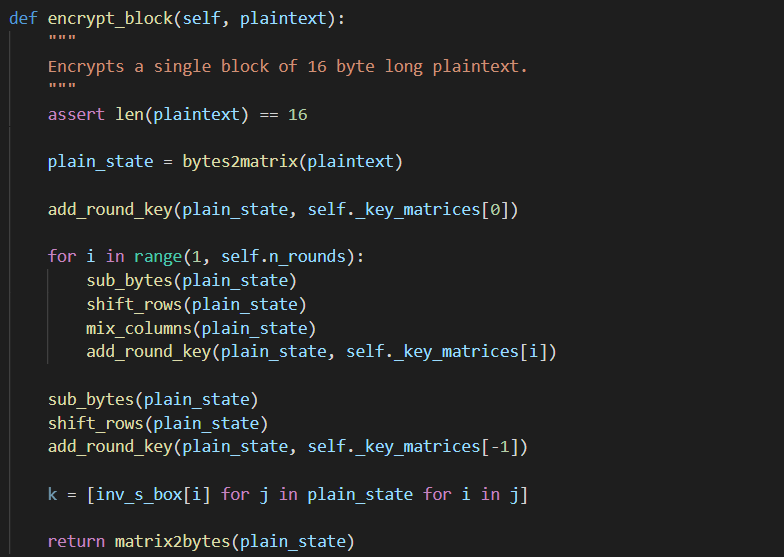


Hình 2‑10 Code hàm MixColumns ()

### Hàm Encrypt ()

Thuật toán mã hóa có cấu trúc sử dụng 4 hàm chính:

* Hàm SubBytes ()
* Hàm ShiftRow ()
* Hàm MixColumns ()
* Hàm AddRoundKey ()



Hình 2‑11 Code hàm Encrypt

### Thời gian mã hóa

Thời gian giải mã tính bằng thời gian bắt đầu giải mã cho đến thời gian kết thúc mã hóa.

Thời gian bắt đầu mã hóa:



Thời gian kết thúc mã hóa:



Tổng thời gian mã hóa (tính theo millisecond):



## Giải mã

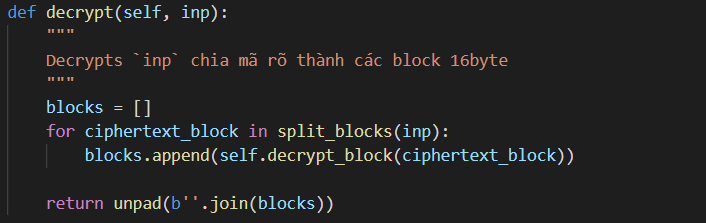
### Hàm Decrypt ()

Đọc file đã mã hóa dưới dạng 1 mảng byte

Tiến hành gọi hàm InvCipher (Phương thức đảo ngược lại quá trình).

Lưu file sau khi giải mã

Code Hàm Decrypt () bằng Python:

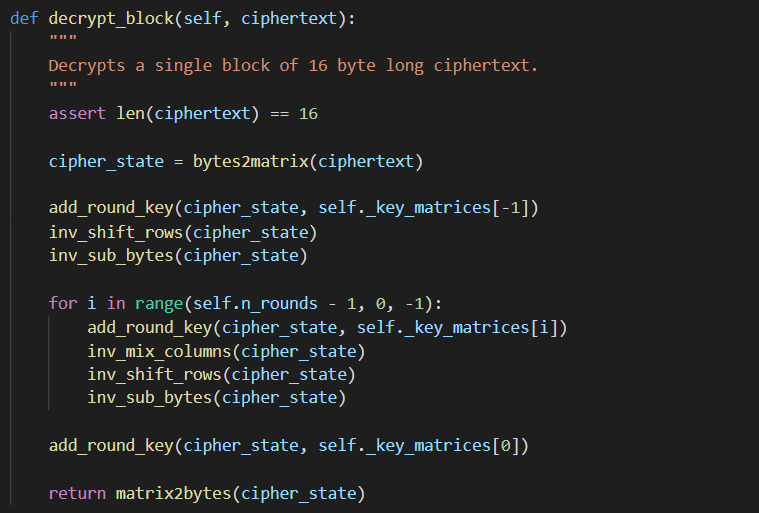


Hình 2‑12 Code Hàm Decrypt ()

### Hàm InvCipher ()

Lần lượt đảo ngược lại quá trình mã hóa.

Code hàm InvCipher () bằng Python:

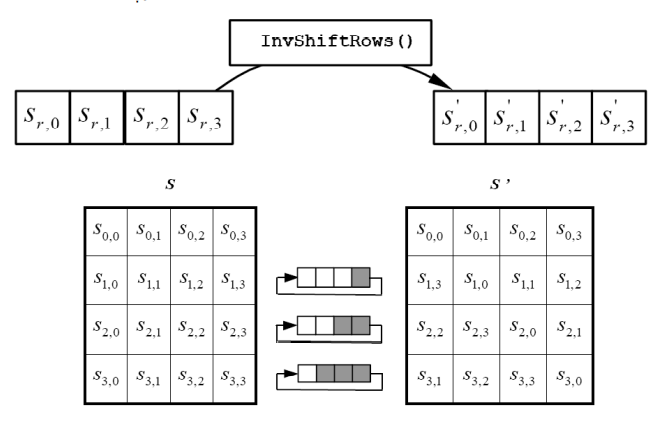


Hình 2‑13 Code hàm InvCipher ()

### Hàm InvShiftRows ()

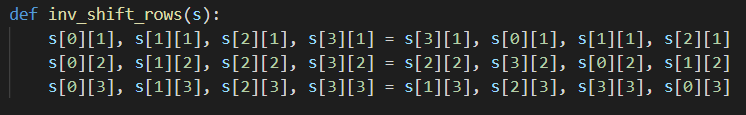
Hàm này là hàm ngược của hàm ShiftRows (). Các byte của ba hàng cuối của

mảng trạng thái sẽ được dịch vòng với các vị trí dịch khác nhau. Hàng đầu tiên không bị dịch, ba hàng cuối bi ̣dịch đi Nb – shift(r, Nb) byte trong đó các giá tri ̣shift (r, Nb) phụ thuôc̣ vào số hàng.



Hình 2‑14 Hàm InvShiftRows () của AES

Code hàm InvShiftRows () bằng Python:

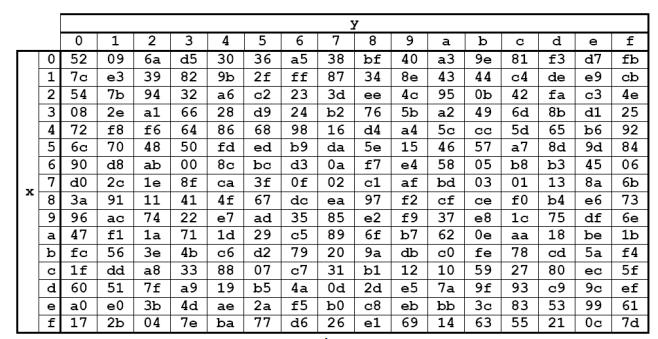


Hình 2‑15 Code *h*àm InvShiftRows ()

### Hàm InvSubBytes ()

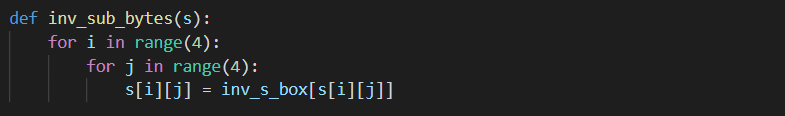
Hàm này là hàm ngược của hàm SubBytes (), hàm sử dụng nghịch đảo của biến đổi Affine bằng cách thực hiêṇ nhân nghịch đảo trên GF (28).

Bảng thế được sử dụng trong hàm là:



Hình 2‑16 Bảng thế cho hàm InvSubBytes ()

Code hàm InvSubBytes () bằng Python:

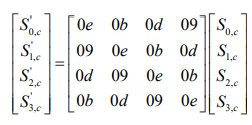


Hình 2‑17 Code hàm InvSubBytes ()

### Hàm InvMixColumns ()

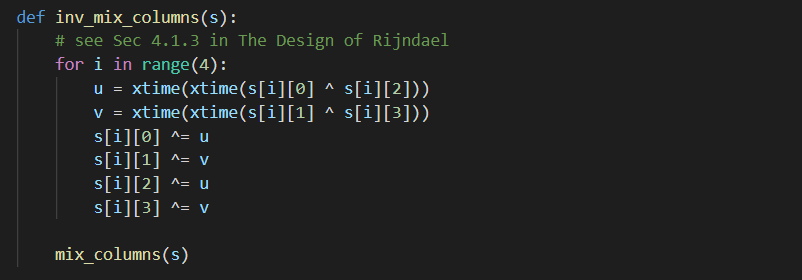
Hàm này là hàm ngược của hàm MixColumns (). Hàm làm việc trên các cột của mảng trạng thái, coi mỗi cột như là một ô đa thức 4 hạng tử. Các cột được xem là các đa thức trên GF(28 ) và được nhân theo modulo x 4 +1 với một đa thức cố định là a -1 (x).

Và có thể mô tả bằng phép nhân ma trận như sau:



trong đó 0 ≤ c < Nb.

Code hàm InvMixColumns () bằng Python:



Hình 2‑18 Code hàm InvMixColumns ()

### Thời gian giải mã

Thời gian giải mã tính bằng thời gian bắt đầu giải mã cho đến thời gian kết thúc giải mã.

Thời gian bắt đầu giải mã:



Thời gian kết thúc giải mã:



Tổng thời gian giải mã (tính theo millisecond):



## Các dạng tấn công AES và phương pháp phòng chống

### Side-channel attack

Side Channels (Kênh kề) được định nghĩa là các kênh đầu ra không mong muốn từ một hệ thống.

Tấn công kênh bên hay còn gọi là Tấn công kênh kề là loại tấn công dễ thực hiện trong các loại tấn công mạnh chống lại quá trình triển khai mã hóa, và mục tiêu của loại tấn công này là phân tích các nguyên tố, các giao thức, modul, và các thiết bị trong mỗi hệ thống.

Phân loại:

* Tấn công thời gian.
* Tấn công dựa vào lỗi.
* Tấn công phân tích năng lượng.
* Tấn công phân tích điện từ.

### Known attacks

Vào năm 2002, Nicolas Courtois và Josef Pieprzyk phát hiện một tấn công trên lý thuyết gọi là tấn công XSL và chỉ ra điểm yếu tiềm tàng của AES.

Tuy nhiên, một vài chuyên gia về mật mã học khác cũng chỉ ra một số vấn đề trong cơ sở toán học của tấn công này và cho rằng các tác giả đã có sai lầm trong tính toán. Việc tấn công dạng này có thực sự trở thành hiện thực hay không vẫn còn để ngỏ và cho tới nay thì tấn công XSL vẫn chỉ là suy đoán.

### Các phương pháp phòng chống

Các phương pháp phòng chống.

Phương pháp 1: Mã hóa cực mạnh

Sử dụng các biện pháp để tăng tính bảo mật của các thuật toán mã hóa.

Phương pháp 2: Bảo vệ dữ liệu theo phương pháp vật lý

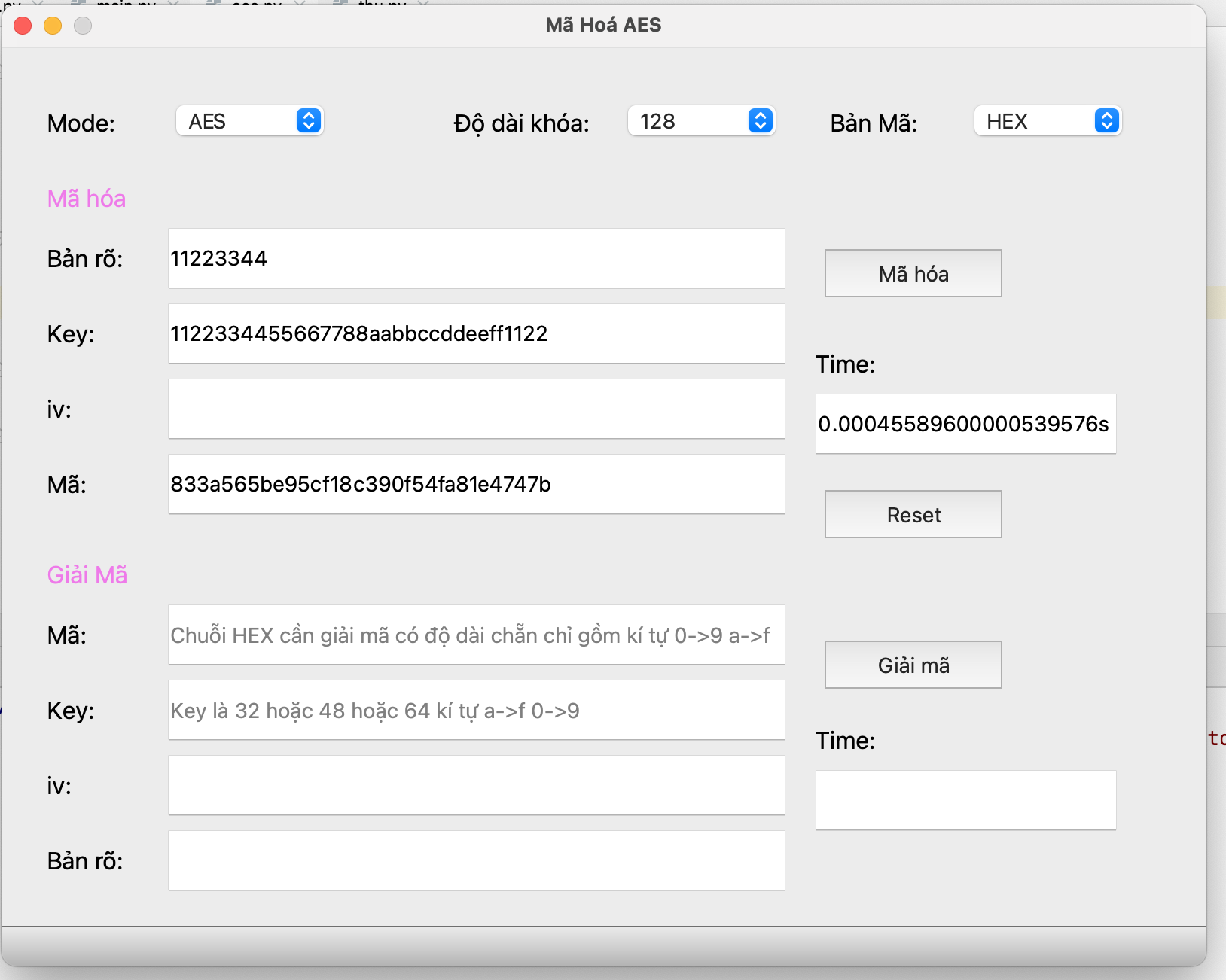
Nếu một kẻ tấn công không thể tiếp cận vật lý với dữ liệu, dĩ nhiên khả năng đánh cắp khóa mã hóa sẽ khó khăn hơn. Vì vậy, trước những cuộc tấn công qua âm thanh tiềm tàng, bạn có thể sử dụng các giải pháp bảo vệ vật lý như đặt laptop vào các hộp cách ly âm thanh, không để ai lại gần máy tính khi đang giải mã dữ liệu hoặc sử dụng các nguồn âm thanh băng rộng tần số đủ cao để gây nhiễu.

Phương pháp 3: Kết hợp cả 2 cách trên.

# KẾT QUẢ THỰC NGHIỆM

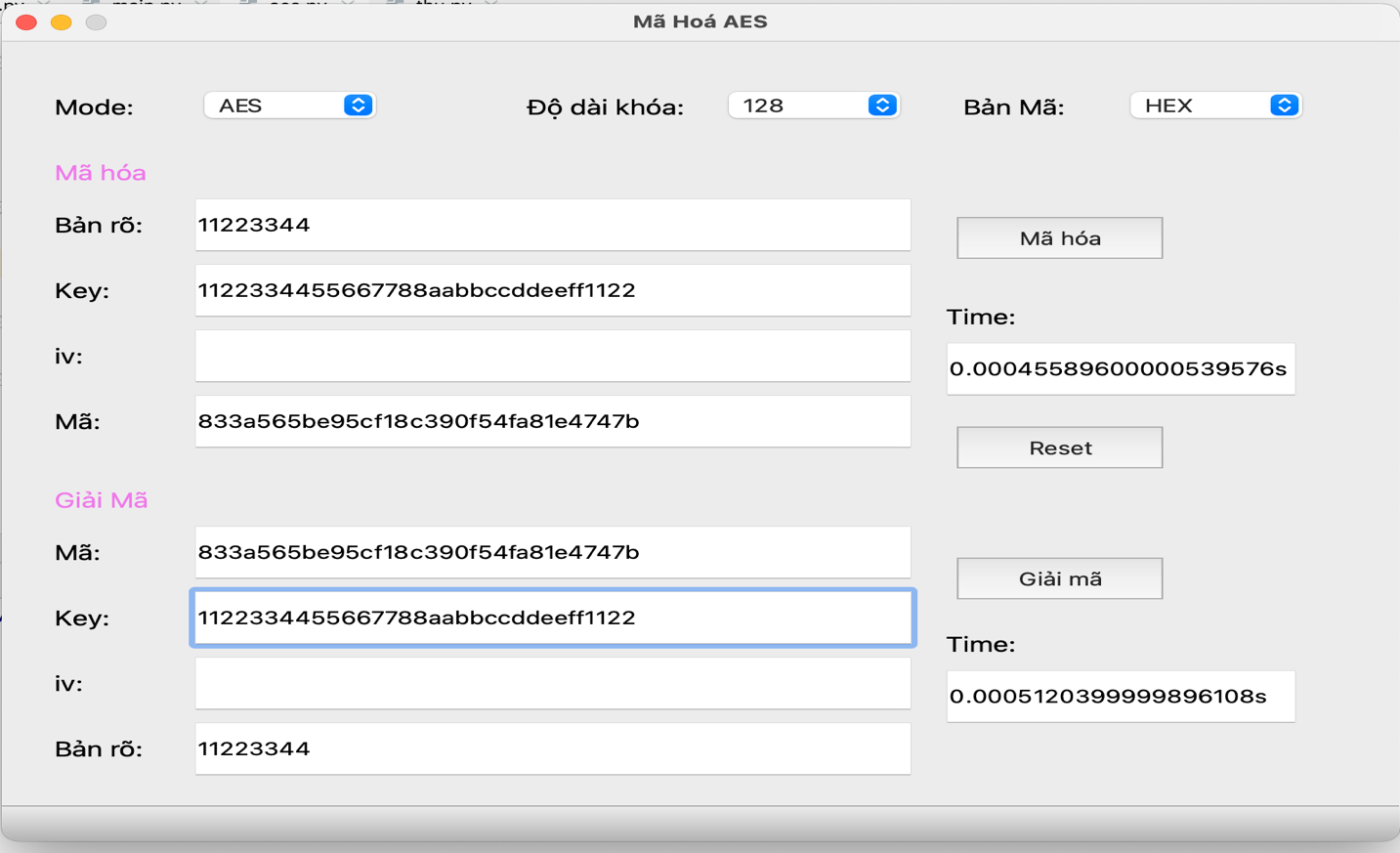
## Mã hóa AES EBC khóa 128 bit

Kết quả mã hóa khóa 128bit:



Hình 3‑1 Kết quả mã hóa khóa 128bit

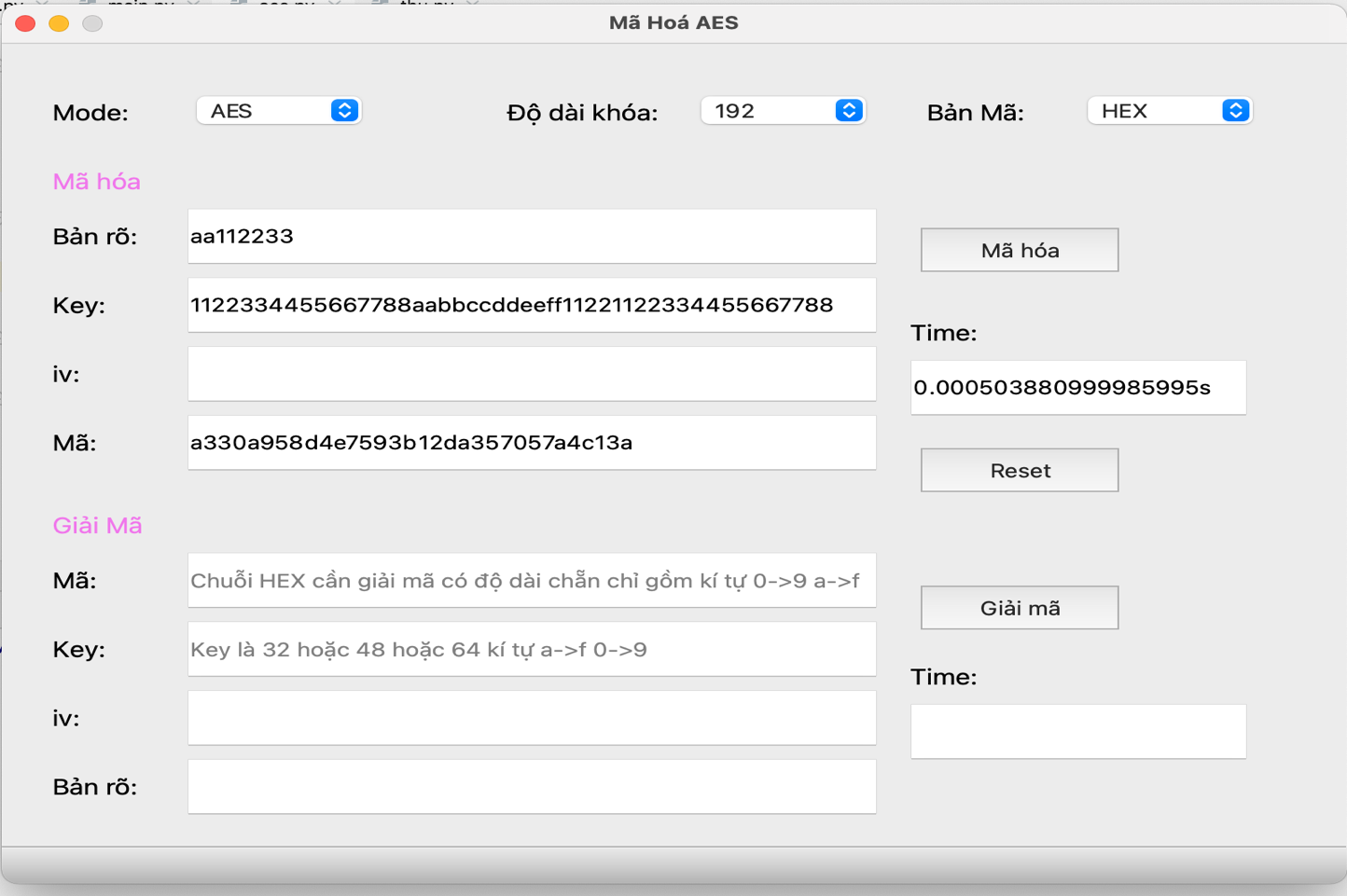
Kết quả giải mã khóa 128bit:



Hình 3‑2 Kết quả giải mã khóa 128bit

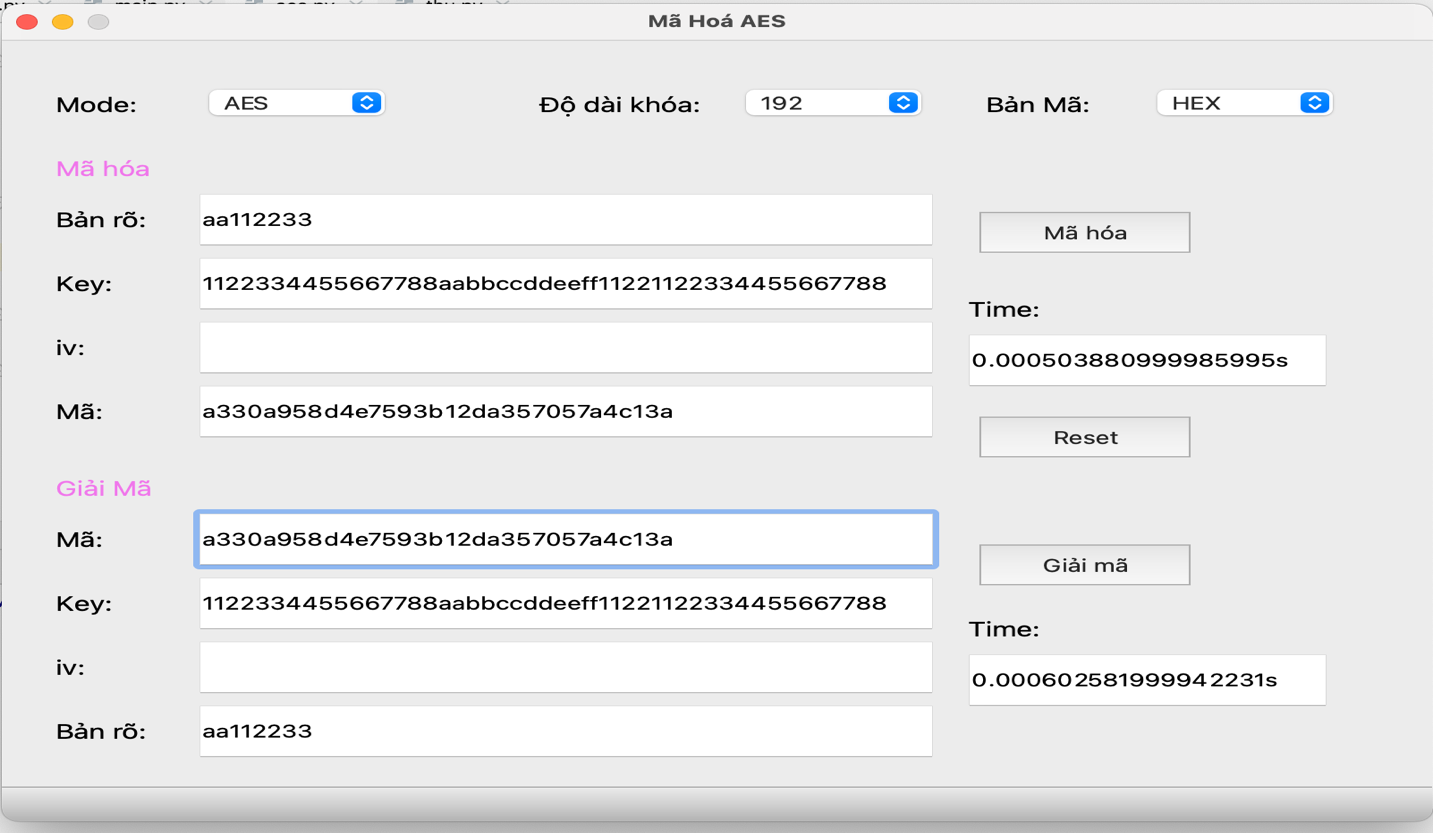
## Mã hóa AES EBC khóa 192 bit

Kết quả mã hóa khóa 192bit:



Hình 3‑3 Kết quả mã hóa khóa 192bit

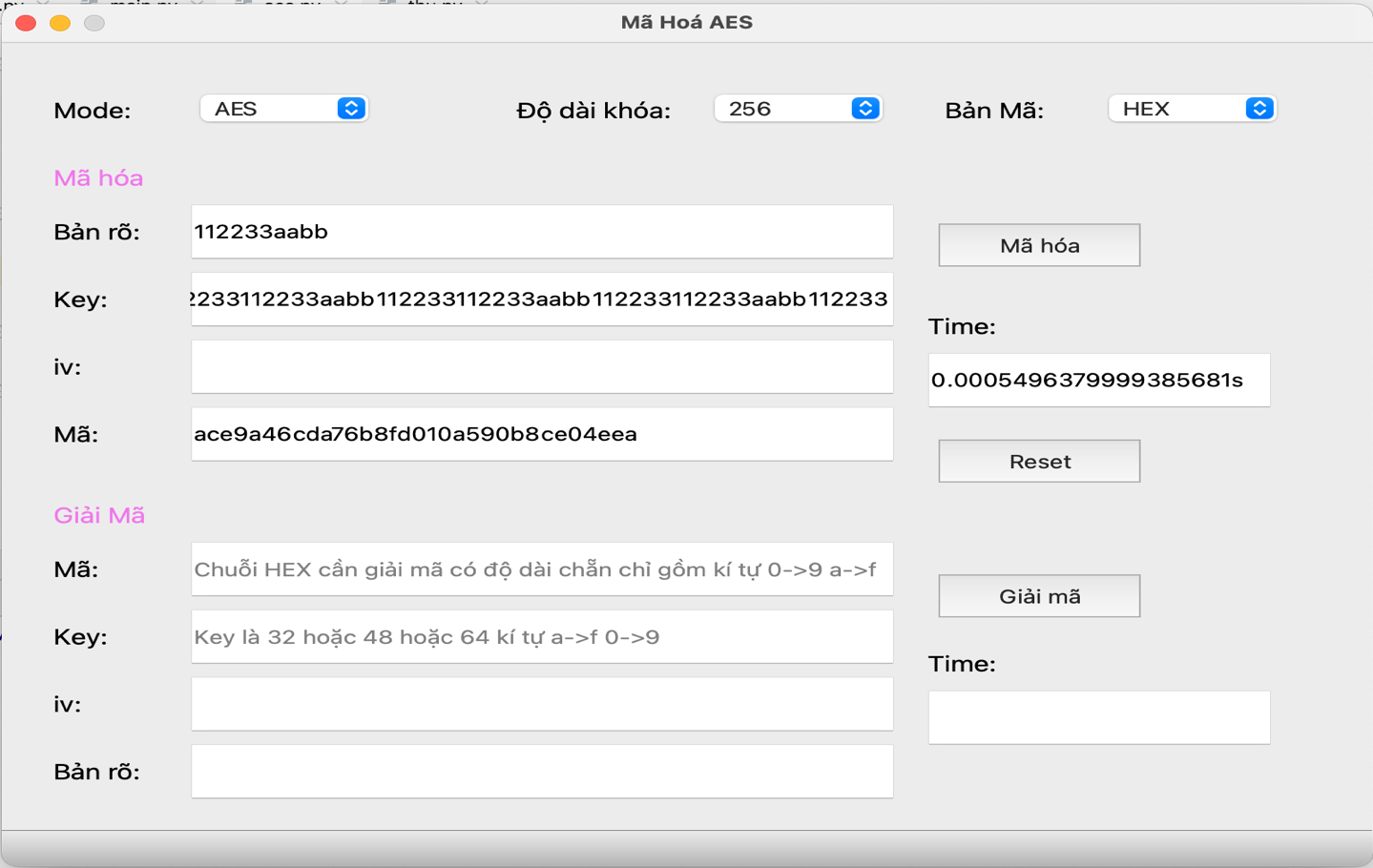
Kết quả giải mã khóa 192bit:



Hình 3‑4 Kết quả giải mã khóa 192bit

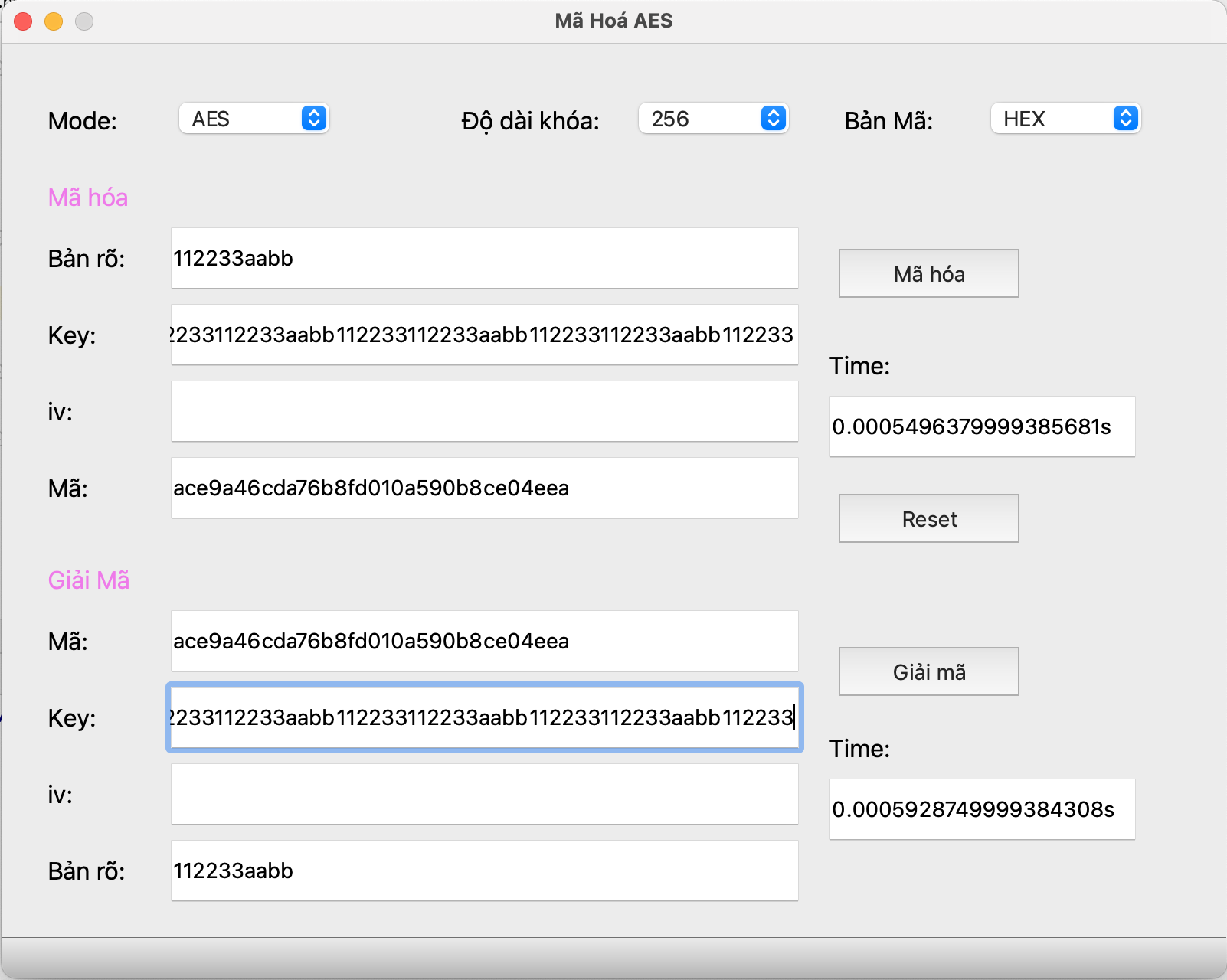
## Mã hóa AES EBC khóa 256 bit

Kết quả mã hóa khóa 256bit:



Hình 3‑5 Kết quả mã hóa khóa 256bit

Kết quả giải mã khóa 256bit:



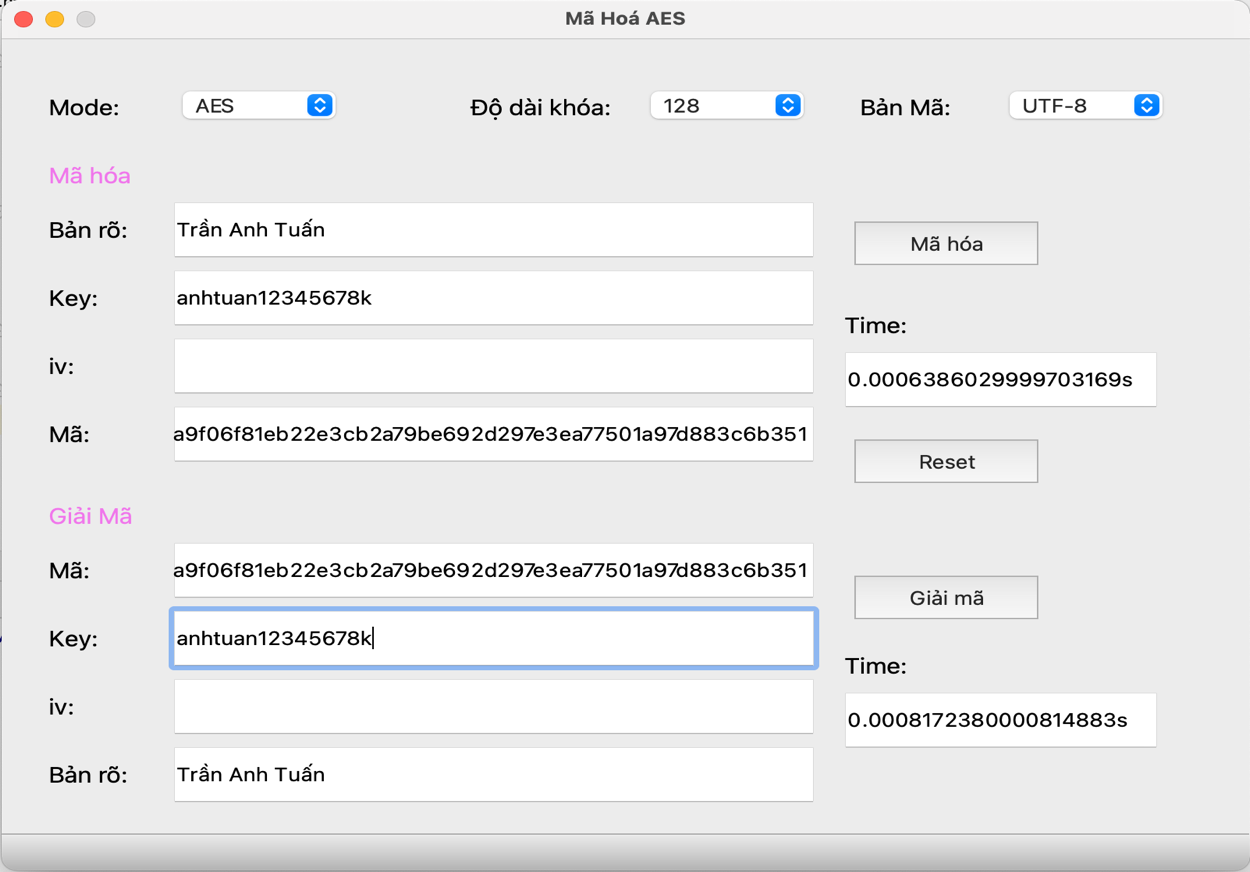
Hình 3‑6 Kết quả giải mã khóa 256bit

## Một số phần mở rộng

### Tích hợp thêm bộ mã hoá

Ngoài đầu vào bằng mã hex, tích hợp thêm bộ mã hoá utf-8 và utf-16 cho phần mềm. Thích hợp cho việc mã hoá với ngôn ngữ tự nhiên và đa ngôn ngữ.

Kết quả khi tích hợp bộ mã utf-8:



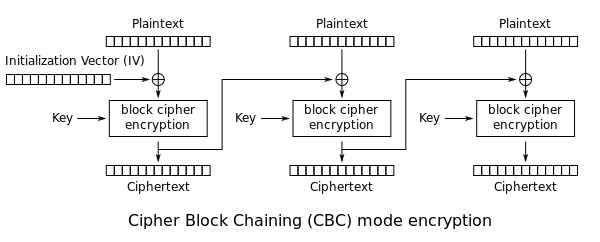
Hình 3‑7 Mã hoá AES EBC 128bit tích hợp bộ mã UTF8

### Tích hợp thêm các chế độ mã hoá khác của AES

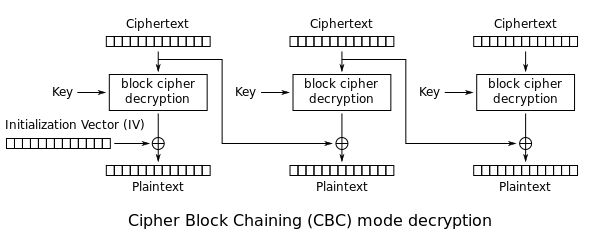
Tích hợp thêm các chế độ mã hoá khác của AES như:

* CBC
* PCBC
* CFB
* OFB
* CTR

Tăng thêm tính bảo mật cho mã hoá vì mã hoá AES EBC mã hoá các block 16 byte riêng lẻ, sự thay đổi ở một block không ảnh hưởng đến các block còn lại.

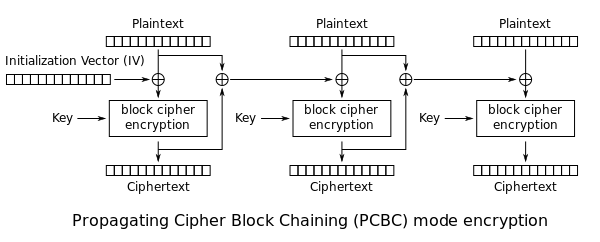


Hình 3‑8 Cách thức hoạt động mã hoá AES CBC

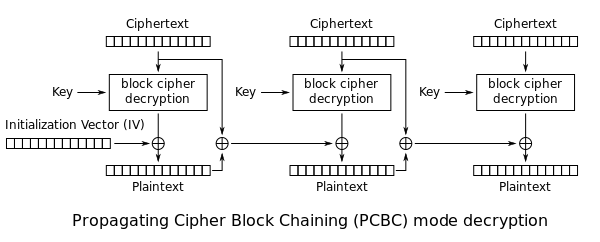


Hình 3‑9 Cách giải mã AES CBC

Trong chế độ CBC mỗi block bản mã rõ được XOR với bản mã trước đó trước khi được mã hoá. Bằng cách này, mỗi block bản mã phụ thuộc vào tất cả các block bản rõ được xử lý cho đến thời điểm đó. Một vector khởi tạo (IV) phải được sử dụng trong block đầu tiên.

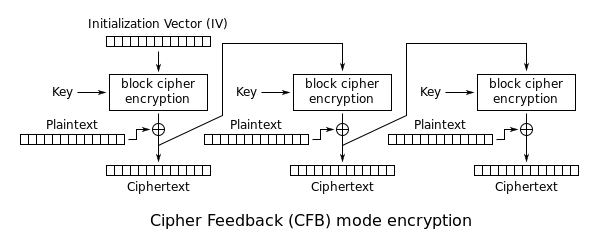


Hình 3‑10 Mã hoá AES PCBC

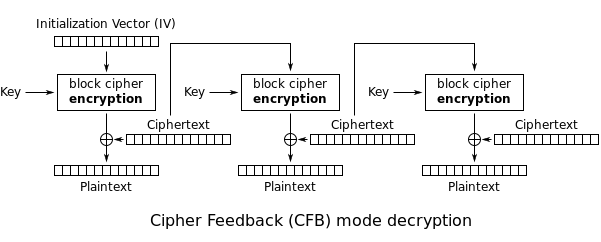


Hình 3‑11Giải mã AES PCBC

Trong chế độ PCBC, mỗi khối bản rõ được XOR với cả khối bản rõ trước và khối bản mã trước đó trước khi được mã hóa. Giống như với chế độ CBC, một vectơ khởi tạo được sử dụng trong khối đầu tiên.

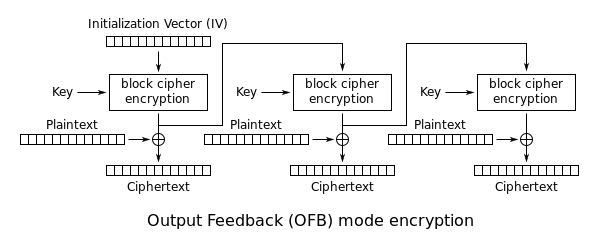


Hình 3‑12 Mã hoá AES CFB

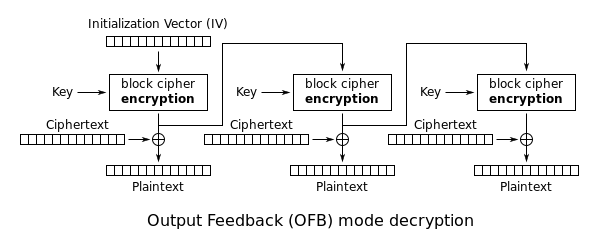


Hình 3‑13 Giải mã AES CFB

Giống như chế độ CBC, những thay đổi trong bản rõ truyền mãi mãi trong bản mã và mã hóa không thể được thực hiện song song. Cũng giống như CBC, việc giải mã có thể được thực hiện song song.

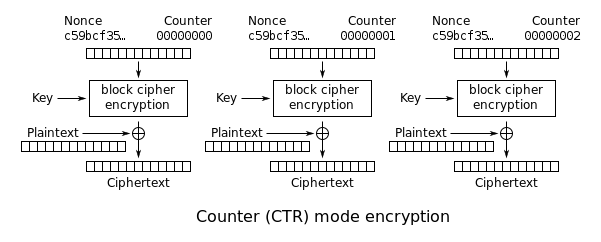


Hình 3‑14 Mã hoá AES OFB

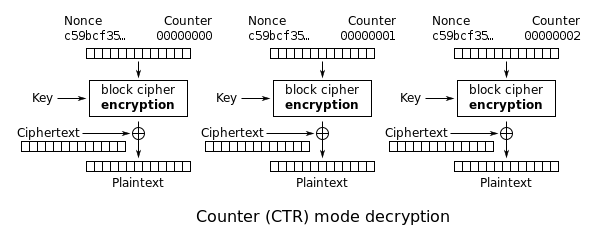


Hình 3‑15 Giải mã AES OFB

Chế độ làm cho một thuật toán mã hóa khối thành một đồng bộ dòng mật mã. Nó tạo ra keystream khối, mà sau đó được XORed với các khối bản rõ để có được bản mã. Cũng giống như với các mật mã dòng khác, việc lật một chút trong bản mã sẽ tạo ra một bit được lật trong bản rõ tại cùng một vị trí. Thuộc tính này cho phép nhiều mã sửa lỗi hoạt động bình thường ngay cả khi được áp dụng trước khi mã hóa.



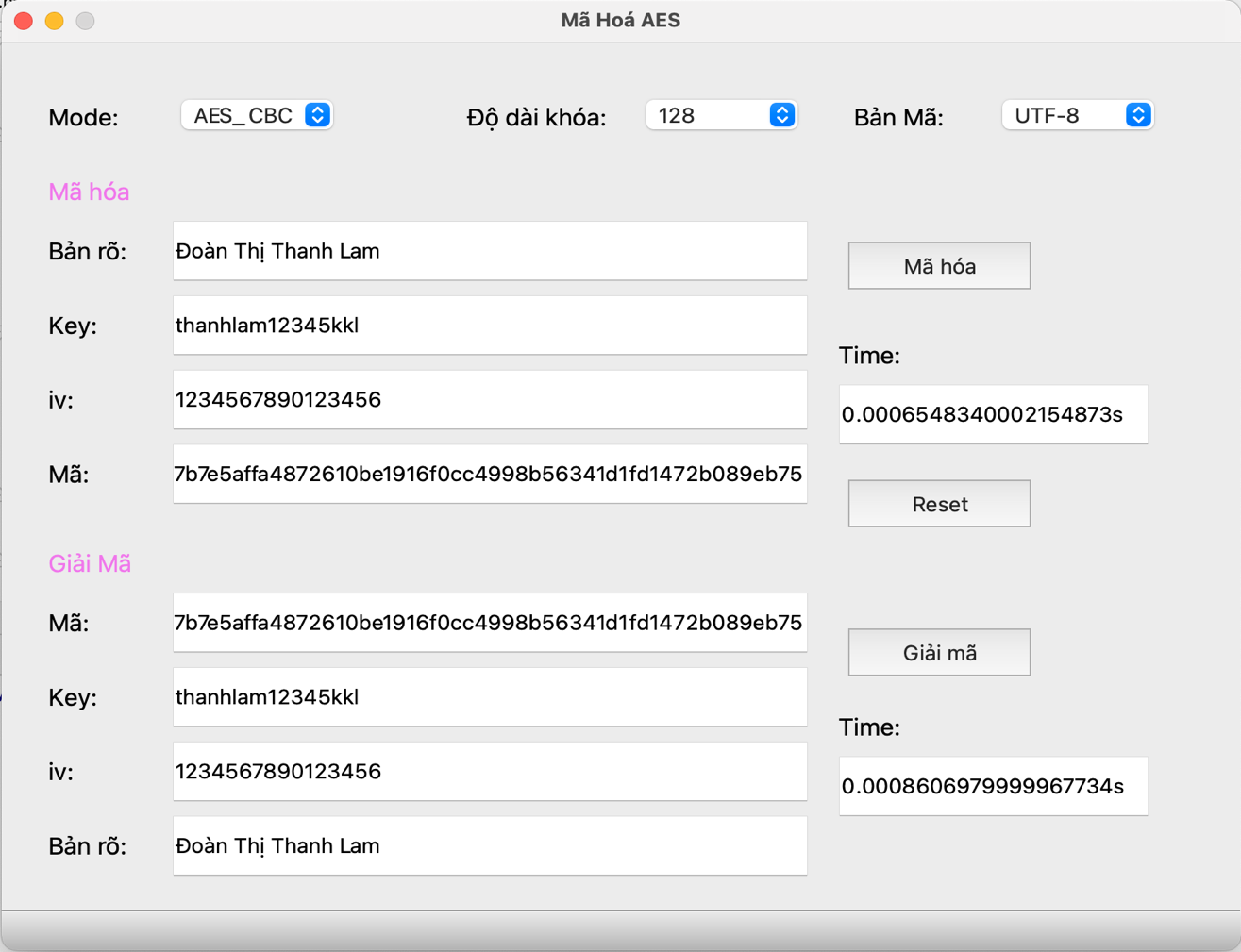
Hình 3‑16 Mã hoá AES CTR



Hình 3‑17 Giải mã AES CTR

Chế độ CTR có các đặc điểm tương tự như OFB, nhưng cũng cho phép một thuộc tính truy cập ngẫu nhiên trong quá trình giải mã. Chế độ CTR rất phù hợp để hoạt động trên một máy nhiều bộ xử lý nơi các khối có thể được mã hóa song song. Hơn nữa, nó không bị vấn đề chu kỳ ngắn có thể ảnh hưởng đến OFB. Ngày nay chế độ CTR được chấp nhận rộng rãi và bất kỳ vấn đề nào được coi là điểm yếu của mật mã khối cơ bản, được mong đợi sẽ được bảo mật bất kể độ lệch hệ thống trong đầu vào của nó.

Lưu ý rằng nonce trong sơ đồ trên tương đương với vector khởi tạo(IV) trong các sơ đồ khác. Tuy nhiên, nếu thông tin offset/location bị hỏng, sẽ không thể khôi phục một phần dữ liệu đó do phụ thuộc vào byte offset.



Hình 3‑18 Mã hoá AES CBC 128bit với bộ mã utf-8

## Nhận xét và đánh giá

Thiết kế và độ dài khóa của thuật toán AES ( 128, 192 và 256 bit ) là đủ an toàn để bảo vệ các thông tin được xếp vào loại tối mật nhưng về an ninh của AES thì các nhà khoa học đánh giá là chưa cao. Nếu các kỹ thuật tấn công được cải thiện thì AES có thể bị phá vỡ.

Một vấn đề khác nữa là cấu trúc toán học của AES khá đơn giản.

PHỤ LỤC

https://github.com/ktap23k/python-aes-128-192-256

KẾT LUẬN

A. Những kết quả đạt được của bài tập môn học

Nêu những kết quả đạt được của môn học, hiểu và nắm được những kiến thức gì, có những thực nghiệm gì cho sự chuẩn bị sau này làm đồ án tốt nghiệp hay ra trường…..

B. Hướng phát triển tiếp theo và đề xuất

Nêu những hướng phát triển tiếp theo của đề tài, trong ứng dụng vào thực tiễn hoặc trong đồ án tốt nghiệp sau này.

Đề xuất thêm gì đối với khoa và học viện.

TÀI LIỆU THAM KHẢO

[1]. TS. Nguyễn Đào Trường. *Slide bài giảng An toàn và bảo mật thông tin.*2021;

[2]. Nguyễn Hữu Tuân. *Giáo trình AT&BMTT*. Tháng 12 năm 2007;

[3]. https://en.wikipedia.org/wiki/Block\_cipher\_mode\_of\_operation

[4]. https://en.wikipedia.org/wiki/Initialization\_vector