|  |
| --- |
| 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**  **ĐỀ TÀI:**  **VIẾT CHƯƠNG TRÌNH MÃ HOÁ VÀ GIẢI MÃ BẰNG MẬT MÃ AES SỬ DỤNG NGÔN NGỮ JAVA** |
| Ngành: Công nghệ thông tin  Sinh viên thực hiện:  Trịnh Hữu Khải – CT030328  Nguyễn Duy Quyết – CT030342  Lớp: L01  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 GIẢNG VIÊN**

…….

LỜI NÓI ĐẦU

Từ xưa, con người đã phải quan tâm tới việc làm thế nào để đảm bảo an toán bí mật cho các tài liệu, văn bản quan trọng, đặc biệt là trong lĩnh vực quân sự, ngoại giao. Ngày nay 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 môi trường mạng - một môi trường mặc định là có nhiều rủi ro, kém 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 cần thiết. Các phương pháp mã hóa dữ liệu lần lượt được ra đời và AES là một trong những phương pháp mã hóa tiêu chuẩn.

AES (viết tắt của từ tiếng Anh: Advanced Encryption Standard, hay Tiêu chuẩn mã hóa tiên tiến) là một thuật toán mã hóa khối được chính phủ Mỹ áp dụng làm tiêu chuẩn mã hóa. Thuật toán AES làm việc với khối dữ liệu 128 bit và khóa độ dài là 128 bit, 192 bit và 256 bit. Mã hóa dùng AES là mã hóa khối lặp gồm nhiều chu trình, các khóa con sử dụng trong các chu trình được tạo ra bởi quá trình tạo khóa con Rijndael.

Chính vì lý do đó, nhóm em đã chọn đề tài “**Viết chương mình mã hoá và giải mã bằng mật mã AES sử dụng ngôn ngữ Java**” cho bài tập lần này, với mong muốn ôn tập và mở rộng kiến thức đã học, rèn luyện kỹ năng làm việc nhóm, tư duy thuật toán. Bài báo cáo nhóm gồm các phần tổng quan về AES, cài đặt chương trình bằng Java và thực nghiệm rút ra kết luận.

Để hoàn thành bài báo cáo này, nhóm em xin gửi lời cám ơn chân thành đến giảng viên ***TS. Nguyễn Đào Trường*** đã tận tình giúp đỡ và truyền đạt những kinh nghiệm quý báu trong suốt thời gian thực hiện đề tài.

Do hạn chế về kỹ năng và kiến thức chuyên môn nên sẽ không tránh khỏi những thiếu sót, kính mong được sự góp ý từ Thầy và mọi người để hoàn thiện bài báo cáo tốt hơn!

MỤC LỤC

[LỜI NÓI ĐẦU 1](#_Toc90538312)

[MỤC LỤC 2](#_Toc90538313)

[DANH MỤC HÌNH VẼ 4](#_Toc90538314)

[Chương 1. TỔNG QUAN AES 5](#_Toc90538315)

[1.1. Tìm hiểu về AES 5](#_Toc90538316)

[1.2. Lịch sử AES 5](#_Toc90538317)

[1.3. Hoạt động của AES 6](#_Toc90538318)

[1.4. AES và mạng Internet 6](#_Toc90538319)

[1.5. Thuật toán mã hoá 6](#_Toc90538320)

[1.5.1. Bảng S-box thuận và S-box nghịch 7](#_Toc90538321)

[1.5.2. Giải thuật sinh khoá phụ 8](#_Toc90538322)

[1.6. Các hàm chính 9](#_Toc90538323)

[1.6.1. Hàm ShiftRow 9](#_Toc90538324)

[1.6.2. Hàm InvShiftRow 10](#_Toc90538325)

[1.6.3. Hàm SubBytes 10](#_Toc90538326)

[1.6.4. Hàm InvSubBytes 11](#_Toc90538327)

[1.6.5. Hàm MixColumns 12](#_Toc90538328)

[1.6.6. Hàm InvMixColumns 12](#_Toc90538329)

[1.6.7. Hàm AddRoundKey 15](#_Toc90538330)

[1.6.8. Các dạng tấn công vào AES và phương pháp phòng chống 16](#_Toc90538331)

[1.7. Giới thiệu về ngôn ngữ lập trình 17](#_Toc90538332)

[1.7.1. Tổng quan về Java 17](#_Toc90538333)

[1.7.2. Tính năng chính 17](#_Toc90538334)

[1.7.3. Thư viện Socket 18](#_Toc90538335)

[Chương 2. CÀI ĐẶT CHƯƠNG TRÌNH BẰNG JAVA 21](#_Toc90538336)

[2.1. Cài đặt chương trình 21](#_Toc90538337)

[2.2. Cài đặt hàm expandKey 22](#_Toc90538338)

[2.3. Cài đặt hàm addRoundKey 23](#_Toc90538339)

[2.4. Cài đặt hàm subBytes và invSubBytes 23](#_Toc90538340)

[2.5. Cài đặt hàm ShiftRow và InvShifRow 23](#_Toc90538341)

[2.6. Cài đặt hàm mixColumnsvà invMixColumns 26](#_Toc90538342)

[2.7. Cài đặt hàm mã hoá và giải mã (cipher và decipher) 26](#_Toc90538343)

[Chương 3. THỰC NGHIỆM VÀ KẾT LUẬN 29](#_Toc90538344)

[3.1. Thực nghiệm 29](#_Toc90538345)

[3.1.1. Thực nghiệm với key 128bit 29](#_Toc90538346)

[3.1.2. Thực nghiệm với key 192bit 30](#_Toc90538347)

[3.1.3. Thực nghiệm với key 256bit 31](#_Toc90538348)

[3.2. Kết luận và đánh giá 32](#_Toc90538349)

[TÀI LIỆU THAM KHẢO 33](#_Toc90538350)

DANH MỤC HÌNH VẼ

# TỔNG QUAN AES

## Tìm hiểu về AES

AES là viết tắt của Advanced Encryption Standard, chuẩn mã hoá dữ liệu rất phổ biến, dùng cho nhiều mục đích và được cả chính phủ Mỹ sử dụng để bảo vệ các dữ liệu tuyệt mật. AES là kiểu mã hoá đối xứng dạng khối, nghĩa là mỗi khối văn bản có một kích thước nhất định (128bits) được mã hoá, khác với mã hoá dạng chuỗi khi từng ký tự được mã hoá. Đối xứng nghĩa là khoá để mã hoá và khoá giải mã đều là một.

## Lịch sử AES

AES được phát triển từ cuối những năm 90s để thay thế chuẩn mã hóa trước đó là Data Encryption Standard (DES) do IBM tạo ra đầu những năm 70s. Nó được chính phủ Mỹ dùng trong năm 1977 nhưng sau đó có nhiều lỗ hổng dễ bị tấn công (brute force, phân tích mật mã khác biệt/tuyến tính) do dựa trên thuật toán 56 bit, nên không còn hữu ích nữa khi vi xử lý máy tính ngày càng mạnh hơn. Vào năm 1998, DES trở thành 3DES hay còn gọi là Triple DES, dùng thuật toán DES để truyền thông điệp 3 lần liên tiếp với 3 khóa mã hóa khác nhau. 3DES khiến dữ liệu an toàn hơn trước kiểu tấn công brute force thời đó.15 thuật toán được đề xuất thay thế DES, bắt đầu quy trình 5 năm của chính phủ Mỹ. AES được hai nhà mật mã học là Vincent Rijmen và Joan Daemen đề xuất, sau được gọi là “đơn Rijindael”.

AES là chuẩn mở vì khi đó chuẩn thực sự cũng chưa được xác định. Trong quá trình thiết kế, nó cũng nhận bình luận, góp ý. Nó được Viện tiêu chuẩn và kỹ thuật quốc gia Hoa Kỳ phát triển với mục tiêu dễ dùng cho cả phần cứng và phần mềm. Một số thay đổi về khóa và khối được thực hiện để tăng tính an toàn.

NSA cũng tham gia xem xét 15 bản đề xuất. Tới tháng 8/1999 chỉ còn 5 thuật toán (Rijndael, Serpent, RC6, Twofish và MARS). Các “ứng viên” được phân tích thêm về độ bảo mật, tính dễ sử dụng, bản quyền, tốc độ, độ chính xác khi mã hóa và giải mã. Người chiến thắng sau cùng là Rijndael, sau đó được đưa lên cho chính phủ Mỹ vào năm 2002 và cả NSA cùng các tổ chức khác. Đến giờ, AES vẫn được dùng cho các tài liệu tuyệt mật, được cho là FIPS (Federal Information Processing Standard - tiêu chuẩn xử lý thông tin liên bang). Sau đó nó được dùng trong khối tư nhân, là chuẩn mã hóa phổ biến nhất với mã hóa khóa đối xứng.

## Hoạt động của AES

Ngày nay ai cũng có thể dùng AES, đúng như nguồn gốc mã nguồn mở của nó. Ví dụ như qua trang [AES Encryption](https://quantrimang.com/url?q=aHR0cHM6Ly9hZXNlbmNyeXB0aW9uLm5ldC8%3D) dữ liệu được đặt vào hộp văn bản, khóa được nhập bên dưới và chọn độ dài khóa, cả 3 độ dài khóa đều được hỗ trợ. Sau đó chọn mã hóa hay giải mã là được. Dù ai cũng có thể dùng AES qua web nhưng giao diện web thiếu an toàn nên khó mà chính phủ có thể sử dụng.

## AES và mạng Internet

Dù ban đầu AES được tạo ra để chính phủ Mỹ bảo mật dữ liệu nhưng nó cũng có thể dùng để chuyển dữ liệu qua mạng an toàn. AES được dùng để chia sẻ tập tin qua [kết nối HTTPS](https://quantrimang.com/https-la-gi-va-tai-sao-no-can-cho-trang-web-cua-ban-142187), an toàn hơn HTTP, vốn được thiết kế để ngăn tấn công trung gian.

AES cũng được dùng để mã hóa WiFi trên router, khi kết hợp với giao thức phổ biến WPA2 thì có tên AES/WPA2. Giao thức bảo mật khác được dùng cùng với WPA2 là TKIP cũ hơn AES và được cho là không còn an toàn. AES cũng được dùng để hỗ trợ mã hóa SSL.

## Thuật toán mã hoá

AES làm việc với các khối dữ liệu 128bit và độ dài khoá 128bit, 192bit hoặc 256bit. Các khoá mở rộng sử dụng trong chu trình được tạo ra bởi thủ tục sinh khoá Rijndael.

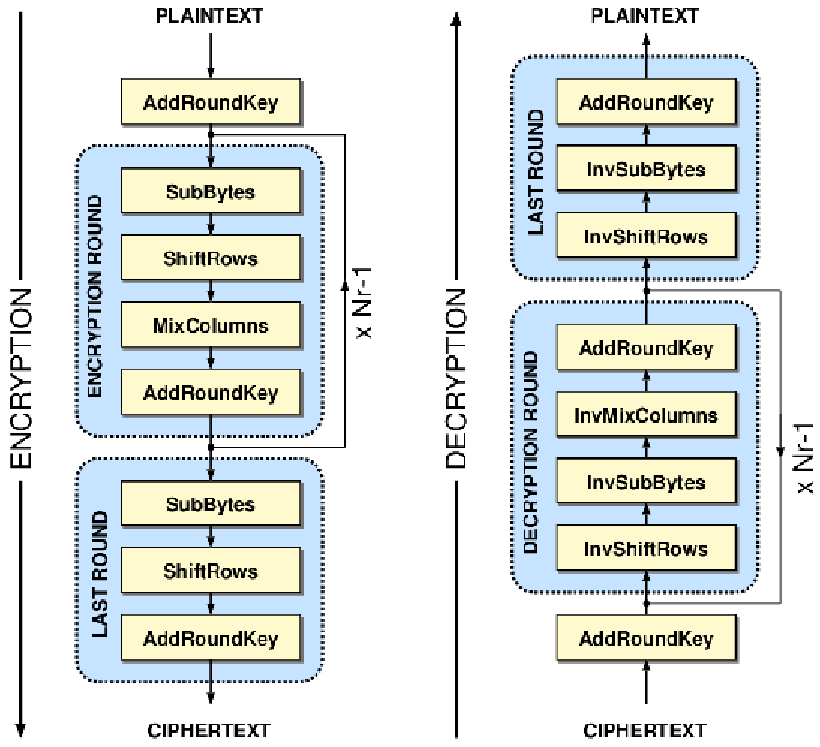
Hầu hết các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn của các byte. Mỗi khối dữ liệu đầu vào 128bit được chia thành 16byte, có thể xếp thành 4 cột, mỗi cột 4 phần tử hay một ma trận 4x4 của các byte, nó gọi là ma trận trạng thái.

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

Quá trình mã hoá trải qua ba bước sau :

* Vòng đầu tiên khởi tạo chỉ gồm AddRoundKey
* Nr-1 vòng lặp tiếp theo lần lượt qua 4 phép biến đổi: SubByte, ShiftRows, MixColumns, AddRoundKey.
* Vòng cuối cùng gồm các phép biến đổi giống các vòng lặp trước nhưng không có phép MixColumns.

Quá trình giải mã cũng được thực hiện với cấu trúc tương tự nhưng với các hàm ngược của các hàm so với quá trình mã hoá. Trừ hàm AddRoudKey cũng chính là làm ngược của chính nó,

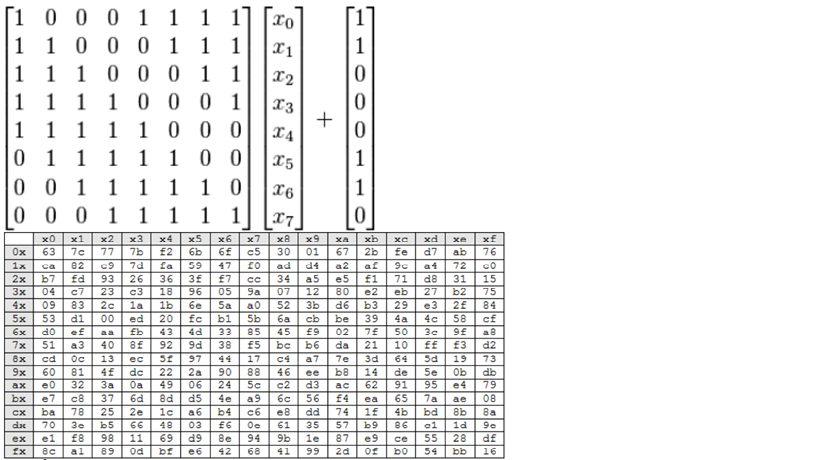


Hình 1. 1 Quá trình mã hoá và giải mã

### Bảng S-box thuận và S-box nghịch

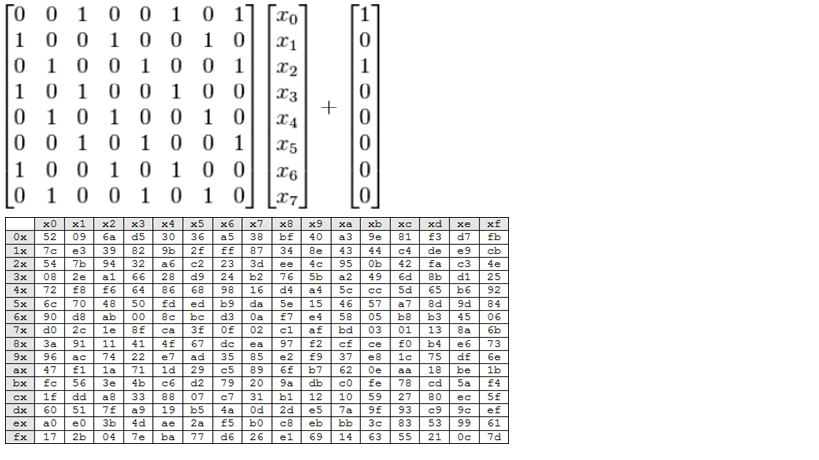
Bảng S-box thuận được sinh ra bằng việc xác định nghịch đảo cho một giá trị nhất định trên GF(28) = GF(2)[x] / (x8+x4+x3+x+1) (trường hữu hạn Rijindael). Giá trị 0 không có nghịch đảo thì được ánh xạ với 0. Những nghịch đảo được chuyển đổi thông qua phép biến đổi affine.

Công thức tính các giá trị bảng S-box và bảng S-box tương ứng:



Hình 1. 2 Bảng S-box thuận

Bảng S-box nghịch chỉ đơn giản là S-box chạy ngược. Nó được tính bằng phép biến đổi affine nghịch đảo các giá trị đầu vào. Phép biến đổi affine nghịch đảo được biểu diễn như sau:



Hình 1. 3 Bảng S-box nghịch

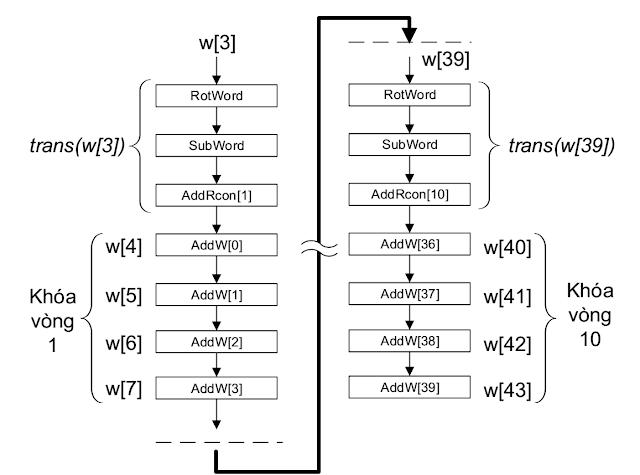
### Giải thuật sinh khoá phụ

Giải thuật sinh khoá phụ gồm 4 bước:

* Rotword: quay trái 8bit
* SubBytes
* Rcon: tính giá tị Rcon(i) trong đó:

Rcon(i) = x(i-1) mod (x8 + x4 + x3 + x + 1)

* ShiftRow



Hình 1. 4 Chức năng keyExpansion

## Các hàm chính

### Hàm ShiftRow

Trong biến đổi ShiftRow(), các byte trong ba hàng cuối cùng của trạng thái được dịch vòng đi các số byte khác nhau, cụ thể:

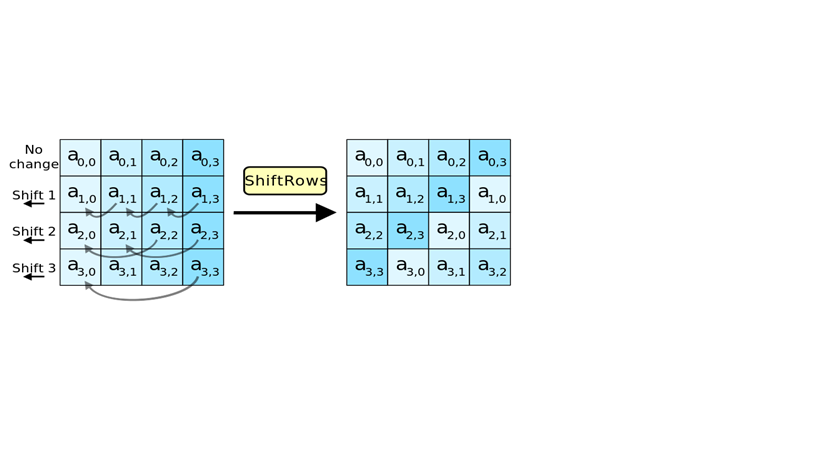
S’r,c = Sr,(c + shift ( r, Nb)) mod Nb (Nb = 4)

Trong đó giá trị dịch shift(r,Nb) phụ thuộc vào số hàng r như sau:

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

Hàng đầu tiên không bị dịch, ba hàng còn lại bị dịch tương ứng:

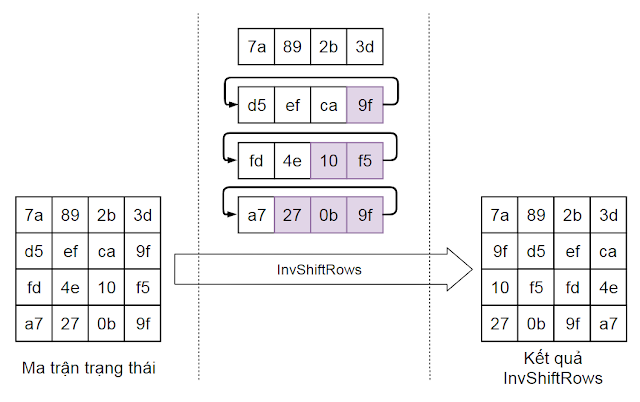
* Hàng thứ nhất giữ nguyên
* Hàng thứ hai dịch vòng trái 1 lần
* Hàng thứ ba dịch vòng trái 2 lần
* Hàng thứ tư dịch vòng trái 3 lần



Hình 1. 5 Hàm ShiftRow

### Hàm InvShiftRow

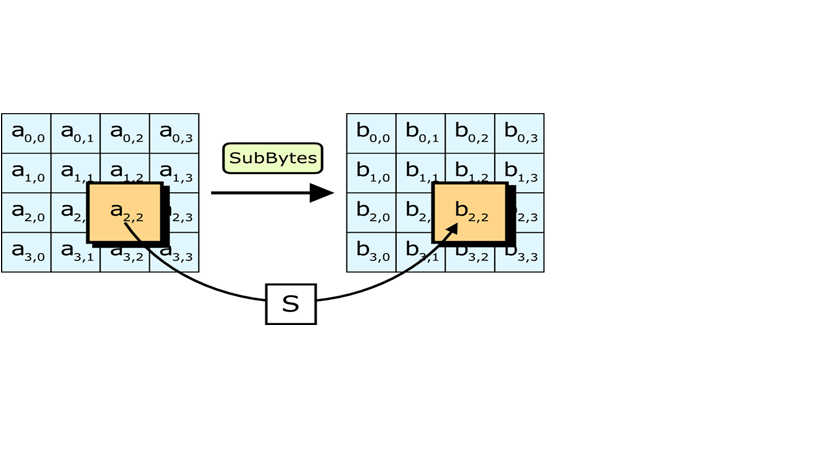
InvShiftRows là đảo của chức năng ShiftRows. InvShiftRows thực hiện quay phải từng hàng của ma trận trạng thái, sinh ra từ bước trước đó, theo byte với hệ số quay tăng dần từ 0 đến 3. Hàng đầu tiên có hệ số quay là 0 thì các byte được giữ nguyên vị trí. Hàng thứ hai có hệ số quay là 1 thì các được quay một byte. Hàng thứ ba quay hai byte và hàng thứ tư quay ba byte.



Hình 1. 6 Hàm InvShiftRow

### Hàm SubBytes

Biến đổi SubBytes() thay thế mỗi byte riêng rẽ của state Sr,c bằng một giá trị mới S’r,c sử dụng bảng thay thế S-box được xây dựng ở trên

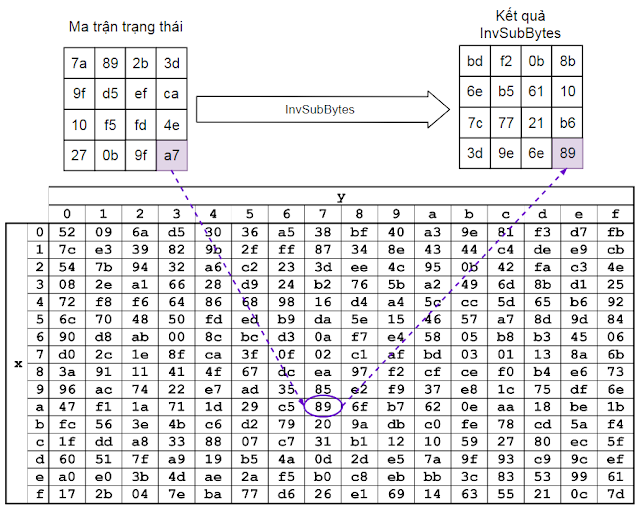


Hình 1. 7 Hàm SubBytes

### Hàm InvSubBytes

Chức năng InvSubBytes là thực hiện thay thế từng byte của ma trận trạng thái, bằng một giá trị đã quy định trong chuẩn AES. Bảng quy định giá trị thay thế cho InvSubBytes gọi là S-box đảo (Inverse S-box).

Ví dụ, byte cần thay thế là Ha7 thì dò ở hàng “a” và cột số 7 trong bảng S-box đảo sẽ được kết quả là H89.



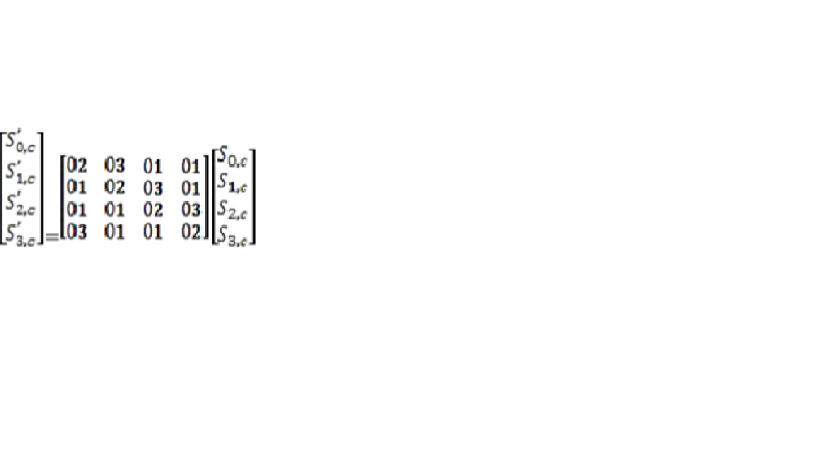
Hình 1. 8 Hàm InvSubBytes

### Hàm MixColumns

Biến đổi MixColumns() tính toán trên từng cột state. Các cột được coi như là đa thức trong trường GF(2^8) và nhân với một đa thức a(x) với:

a(x) = (03)x^3 +(01)x^2 +(01)x + (02)

Biến đổi này có thể được trình bày như phép nhân một ma trận, mà mỗi byte được hiểu như là một phần tử trong trường GF(2^8): s’(x) = a(x) ⊗ s(x):



Hình 1. 9 Hàm MixColumns

### Hàm InvMixColumns

InvMixColumns của quá trình giả mã là đảo của MixColumns trong quá trình mã hóa. Từng cột của ma trận trạng thái sẽ được nhân với ma trận chuyển đổi sau đây.

Việc biến đổi một cột của ma trận trạng thái được thực hiện bởi hai phép toán là nhân (.) và XOR (+).

Nguyên tắc tính toán InvMixColumns là biến đổi phép nhân với một số thành phép nhân với *H02* và *H01*. Phép nhân với *H01* thì giữ nguyên giá trị. Phép nhân với *H02* tương đương với việc dịch trái một bit và XOR có điều kiện như sau:

Nếu bit MSB của giá trị được dịch bằng 1 thì giá trị sau khi dịch được XOR với H1b.

Nếu bit MSB của giá trị được dịch bằng 0 thì giữ giá trị sau khi dịch.

Như vậy, việc biến đổi về phép nhân với *H01* và *H02* sẽ giúp dễ dàng thiết kế mạch logic thực hiện chức năng InvMixColumns với tài nguyên được tối ưu hóa. Thiết kế và RTL code sẽ được phân tích trong những bài sau. Tuy nhiên, cách tính InvMixColumns sẽ được trình bày sau đây:

**Phép nhân một byte A với H0e=B00001110** sẽ tương đương như sau:

**A.H0e = A.H08 + A.H04 + A.H02**

Trong đó, phép nhân với H04 và H08 hoàn toàn có thể chuyển về phép nhân với H02.

**A.H04 = A.H02.H02**

**A.H08 = A.H02.H02.H02**

Như vậy:

**A.H0e = A.H02.H02.H02 + A.H02.H02 + A.H02**

Phép nhân một byte A với H0b=B00001011 sẽ tương đương như sau:

**A.H0e = A.H08 + A.H02 + A.H01 = A.H02.H02.H02 + A.H02 + A.H01**

Phép nhân một byte A với H0d=B00001101 sẽ tương đương như sau:

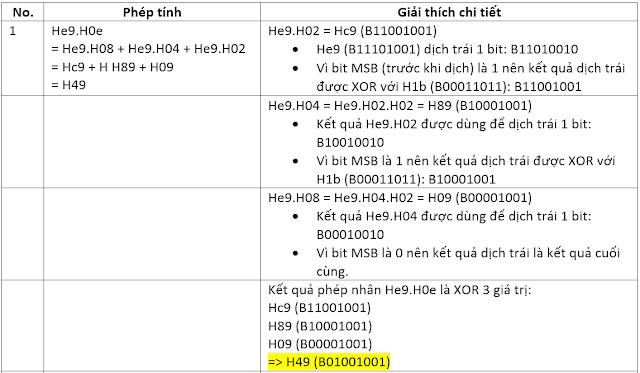
**A.H0e = A.H08 + A.H04 + A.H01 = A.H02.H02.H02 + A.H02.H02 + A.H01**

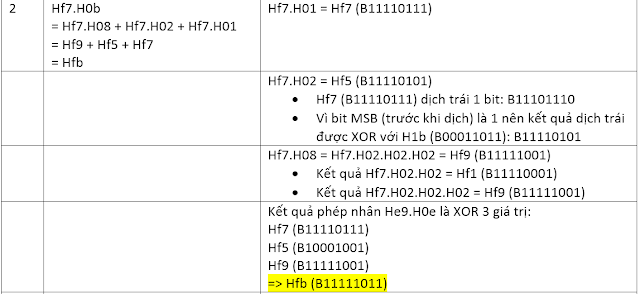
Phép nhân một byte A với H09=B00001001 sẽ tương đương như sau:

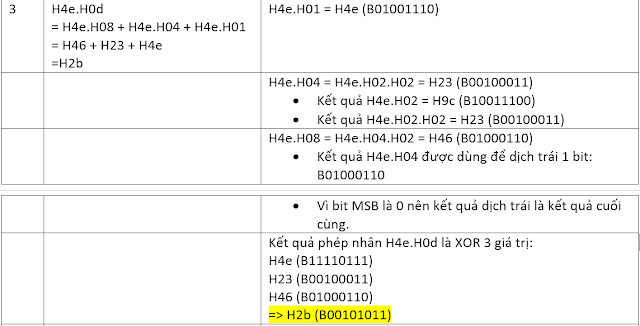
**A.H0e= A.H08 + A.H01= A.H02.H02.H02 + A.H01**

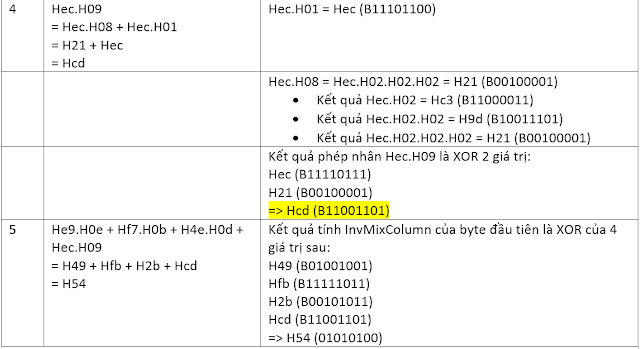
Biểu thức sau là biến đổi của phần tử H54, H là ký hiệu của số Hex, ở cột 1 trong hình minh họa "chức năng InvMixColumns".

**H54 = He9.H0e + Hf7.H0b + H4e.H0d + Hec.H09**



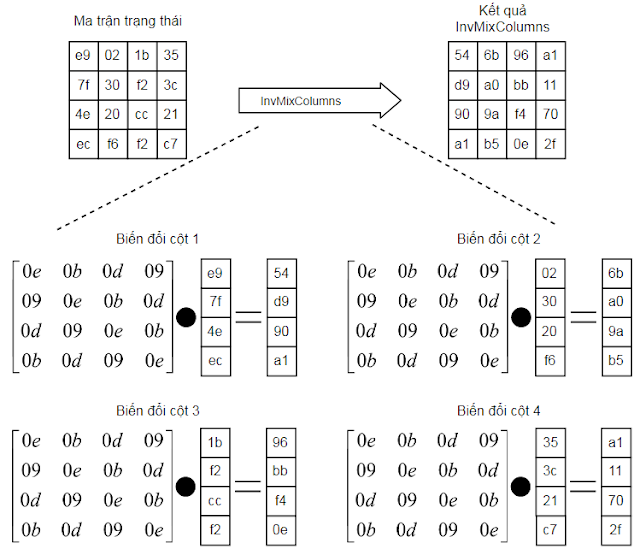






Hình 1. 10 Chi tiết cách tính InvMixColumns tạo ra phần tử H54 từ cột 1 của ma trận trạng thái

Việc tính InvMixColumns cho toàn bộ một ma trận trại thái như sau:



Hình 1. 11 Chức năng InvMixColumns

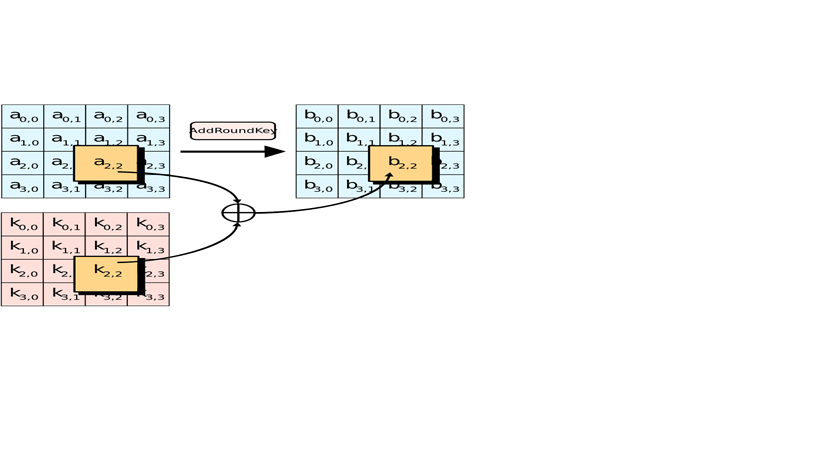
### Hàm AddRoundKey

Được áp dụng từ vòng lặp thứ nhất tới vòng lặp Nr.

Trong biến đổi AddRoundeKey(), một khoá vòng được cộng với state bằng một phép XOR theo từng bit đơn giản.

Mỗi khoá vòng gồm có 4 từ (128bit) được lấy từ lịch trình khoá. 4 từ đó được cộng vào mỗi cột của state sao cho:

[S’0,c, S’1,c, S’2,c, S’3,c ] = [S0,c, S1,c, S2,c, S3,c ] ⊕ [W(4\*i + c)] với 0 <= c < 4



Hình 1. 12 Hàm AddRoundKey

### Các dạng tấn công vào 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:

o Tấn công thời gian.

o Tấn công dựa vào lỗi.

o Tấn công phân tích năng lượng.

o Tấn công phân tích điện từ.

#### Known attack

* 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.

#### 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.

## Giới thiệu về ngôn ngữ lập trình

### Tổng quan về Java

Java là một ngôn ngữ lập lập trình, được phát triển bởi Sun Microsystem vào năm 1995, là ngôn ngữ kế thừa trực tiếp từ C/C++ và là một ngôn ngữ lập trình hướng đối tượng.

Vì sao ngôn ngữ này lại được đặt tên là Java? Java là tên một hòn đảo ở Indonesia - hòn đảo nổi tiếng với loại coffee Peet và cũng là loại nước uống phổ biến của các kỹ sư Sun. Ban đầu Ngôn ngữ này được đặt tên là "Oak" (có nghĩa là "Cây sồi" - 1991), nhưng các luật sư của Sun xác định rằng tên đó đã được đăng ký nhãn hiệu nên các nhà phát triển đã phải thay thế bằng một tên mới - và cũng vì lý do trên mà cái tên Java đã ra đời và trở thành tên gọi chính thức của Ngôn ngữ này - Ngôn ngữ Lập trình Java.

### Tính năng chính

* **Đơn giản**
* Cú pháp dựa trên C++
* Gỡ bỏ đặc điểm gây bối rối như : các con trỏ tường minh, nạp chồng toán tử,…
* **Hướng đối tượng**

Hướng đối tượng nghĩa là chúng ta tổ chức phần mềm dưới dạng kết hợp của nhiều loại đối tượng khác nhau, trong đó có sự kết hợp chặt chẽ cả về dữ liệu và hành vi của chúng.

Lập trình hướng đối tượng (OOP) là một phương pháp làm đơn giản hóa việc phát triển và bảo trì phần mềm bằng việc cung cấp một số qui tắc.

* **Độc lập nền tảng**

Java code có thể chạy trên nhiều nền tảng như Windows, Linux, Sun Solaris, Mac/OS, … Java code được biên dịch bởi Trình biên dịch (Compiler) và được chuyển đổi thành Bytecode. Bytecode này là một code độc lập nền tảng bởi vì nó có thể chạy trên nhiều nền tảng khác nhau. Đó là lý do vì sao java có khẩu hiệu “Viết một lần, Chạy khắp nơi (Write Once and Run Anywhere)”.

* **Bảo mật**
* Không có con trỏ tường minh
* Chương trình chạy bên trong máy ảo
* Classloader: Thêm sự bảo mật bằng việc phân chia package cho các class của hệ thống file trên local mà từ đó chúng được import với các file từ nguồn mạng.
* Bytecode Vertifier: Kiểm tra các đoạn code để tìm ra các phần code không hợp lệ mà có thể truy cập trái phép tới các đối tượng.
* Security Manager: Quyết định xem nguồn source nào mà một lớp có thể truy cập trái phép tới các đối tượng.
* **Đa luồng**

Chúng ta có thể tạo các ứng dụng phân tán trong Java. RMI và EJB được sử dụng để tạo các ứng dụng này. Chúng ta có thể truy cập các file bằng việc gọi các phương thức từ bất cứ thiết bị nào trên internet.

### Thư viện Socket

* **Khái niệm:**

Socket là một điểm cuối của liên kết giao tiếp hai chiều giữa hai chương trình đang chạy trên mạng. Các lớp socket được sử dụng để thể hiện kết nối giữa chương trình máy khách và chương trình máy chủ.

Điểm cuối này bao gồm một địa chỉ IP và số cổng kết nối. Gói java.net cung cấp hai lớp – Socket và ServerSocket – tương ứng triển khai phía máy khách của kết nối và phía máy chủ của kết nối.

* **Hoạt động**

Thông thường, một máy chủ hoạt động có một socket liên kết với một số cổng cụ thể. Máy chủ khi hoạt động sẽ chờ và lắng nghe socket của máy khách thực hiện tạo request.

Trong khi đó, bên phía máy khách khi hoạt động đã biết hostname – tên của máy chủ và port number – số cổng mà máy chủ đang nghe. Để thực hiện một yêu cầu kết nối, máy khách sử dụng hostname và port number này để cố gắng kết nối với máy chủ. Bên cạnh đó, máy khách cũng phải tự khai báo cổng mà nó sử dụng trong suốt quá trình kết nối.

Nếu máy chủ chấp nhận kết nối này, nó sẽ tạo một socket được liên kết với port mà nó sử dụng, đồng thời điểm cuối của kết nối này là địa chỉ và port của máy khách.

Về phía máy khách sau khi đã có kết nối với máy chủ, môt socket cũng được tạo ra và máy khách có thể dùng socket này để liên lạc với máy chủ.

Vậy giữa máy chủ và máy khách hiện thời đã có thể liên lạc được với nhau thông qua việc ghi hoặc đọc dữ liệu từ socket vừa được tạo.

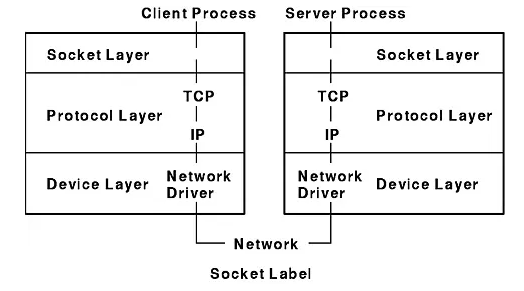


Hình 1. 13 Hoạt động của socket

* **Làm việc socket trong java**

Gói java.net cung cấp class Socket, nhằm thực hiện các kết nối hai chiều bên phía máy khách để các ứng dụng Java trong mạng có thể giao tiếp với nhau. Lớp này nằm trên một triển khai phụ thuộc nền tảng, nó ẩn đi mọi chi tiết về hệ thống tách biệt với chương trình Java, bằng cách này thay vì dựa vào mã gốc, các chương trình Java của bạn có thể giao tiếp qua mạng một cách độc lập.

Thêm vào đó java.net cung cấp class SocketServer, phía máy chủ có thể sử dụng socket này để nghe và nhận các yêu cầu kết nối từ phía máy khách.



Hình 1. 14 Quy trình làm việc của socket trong java

# CÀI ĐẶT CHƯƠNG TRÌNH BẰNG JAVA

## Cài đặt chương trình

Chương trình sử dụng 1 thuật toán AES và giao thức TCP/IP để giao tiếp qua lại giữa máy chủ và máy khách. Các thông điệp được mã hóa bằng AES, và khóa được sử dụng trong thuật toán AES được trao đổi qua lại giữa Client và Server bằng giao thức Socket .Cách làm này cho phép mô-đun Máy khách tải tệp lên mô-đun Máy chủ một cách an toàn.

* **Phía Server**

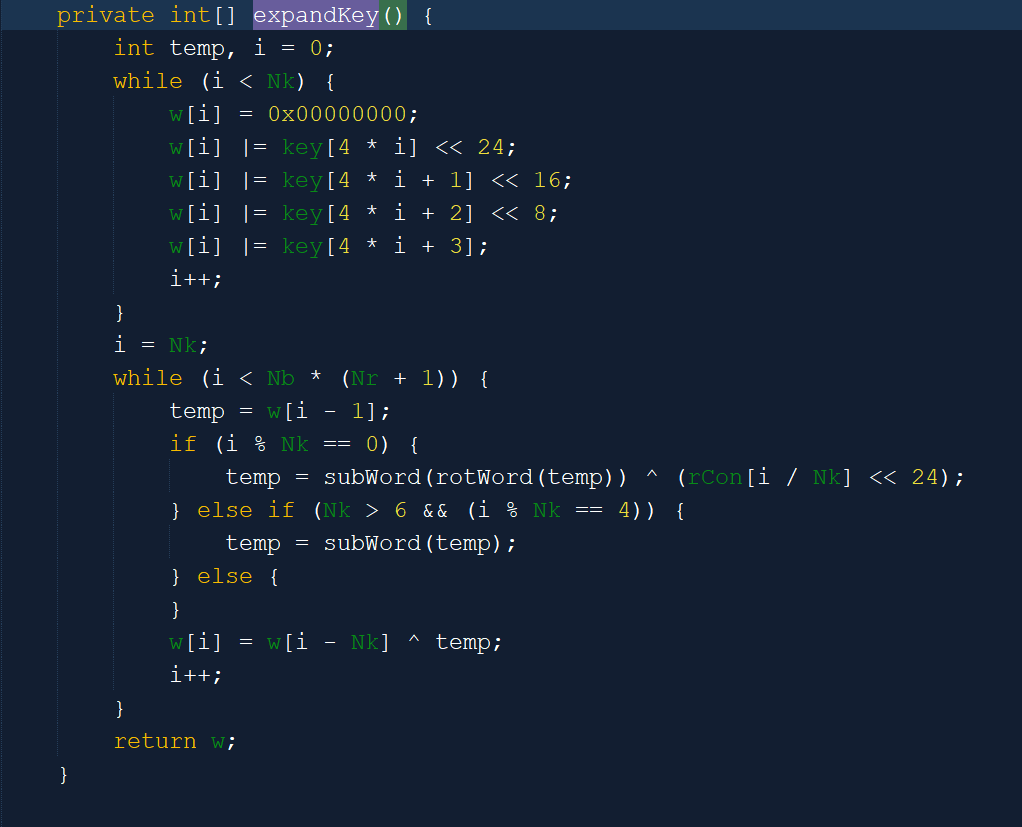
Máy chủ là đa luồng, do đó có thể chấp nhận và xử lý các yêu cầu từ nhiều máy khách đồng thời. Các máy khách muốn giao tiếp với máy chủ có thể thực hiện một cách an toàn bằng cách mã hóa thông tin bằng khóa công khai của máy chủ. Việc này được thực hiện bằng cách máy khách gửi yêu cầu “GET PUBLIC KEY'” đến máy chủ, khi nhận được yêu cầu máy chủ sẽ gửi lại máy khách khóa công khai. Sau khi máy chủ giải mã thành công nó sẽ lưu tệp đã nhận và sẽ gửi thông tin sang máy khách nếu được yêu cầu.

* **Phía Client**

Trước tiên vì mục đích là để bảo vệ dữ liệu của tệp khi nó được gửi đến máy chủ, máy khách sẽ mã hóa nội dung tệp bằng cách sử dụng mã hóa AES 128 bit hoặc AES 192 bit hoặc AES 256 bit. Vì đây là kỹ thuật mã hóa khóa đối xứng, máy chủ cũng cần khóa AES này để giải mã tệp. Do đó, khi gửi yêu cầu “TRANSFER FILE”, máy khách sẽ gửi chính khóa AES được mã hóa bằng khóa công khai của máy chủ, cũng như tên tệp và nội dung được mã hóa bằng khóa AES.

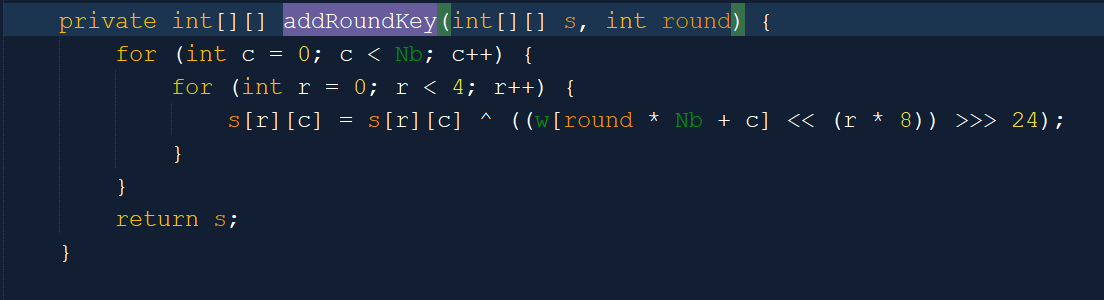
## Cài đặt hàm expandKey

* Hàm keyExpansion được sử dụng để sinh ra các khóa sử dụng tại các vòng lặp của thuật toán mã hóa, giải mã từ khóa chính ban đầu
* Đầu tiên, hàm toWord chuyển đổi mảng key (1 byte) thành mảng word (4 byte). Với mỗi loại khóa 128bit, 192bit, 256bit, ta có Nr tương ứng là 10, 12, 14 🡪 word có 4\*(Nr + 1) phần tử
* Sau đó, w[i] với Nk < i < \*(Nr + 1) được biến đổi qua các hàm rotWord, subWord
* Hàm rotWord nhận w[i-1] và thực hiện 1 hoán vị vòng
* Hàm subWord nhận word sau khi biến đổi bởi hàm rotWord và lấy giá trị từ Sbox dựa trên mỗi giá trị 4 byte để thu 1 word mới
* Sau đó, XOR kết quả subWord với giá trị Rcon[i/Nk], rồi XOR tiếp với w[i-Nk] để thu 1 word hoàn chỉnh



Hình 2. 1 Cài đặt hàm expandKey

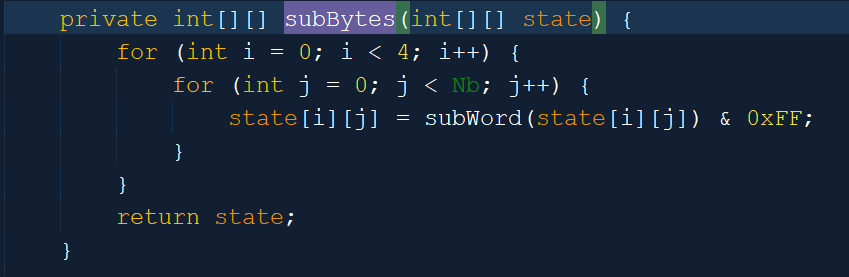
## Cài đặt hàm addRoundKey



Hình 2. 2 Cài đặt hàm addRoundKey

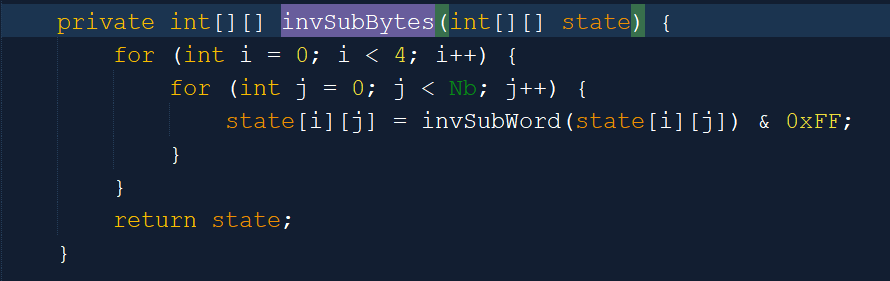
## Cài đặt hàm subBytes và invSubBytes

Hàm subBytes nhận mảng state và lấy giá trị từ Sbox dựa trên mỗi giá trị state.



Hình 2. 3 Cài đặt hàm subBytes

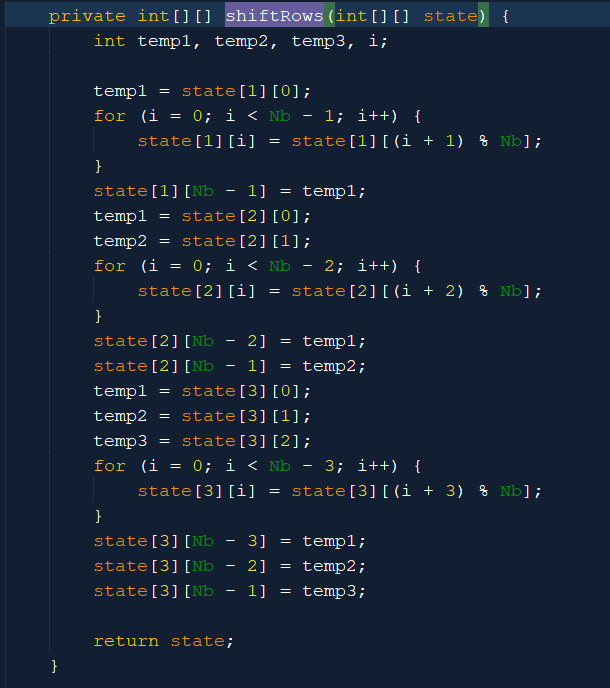
Hàm invSubBytes là hàm ngược của subBytes, nó nhận mảng state và lấy giá trị từ sboxInv dựa trên mỗi giá trị state.



Hình 2. 4 Cài đặt hàm invSubBytes

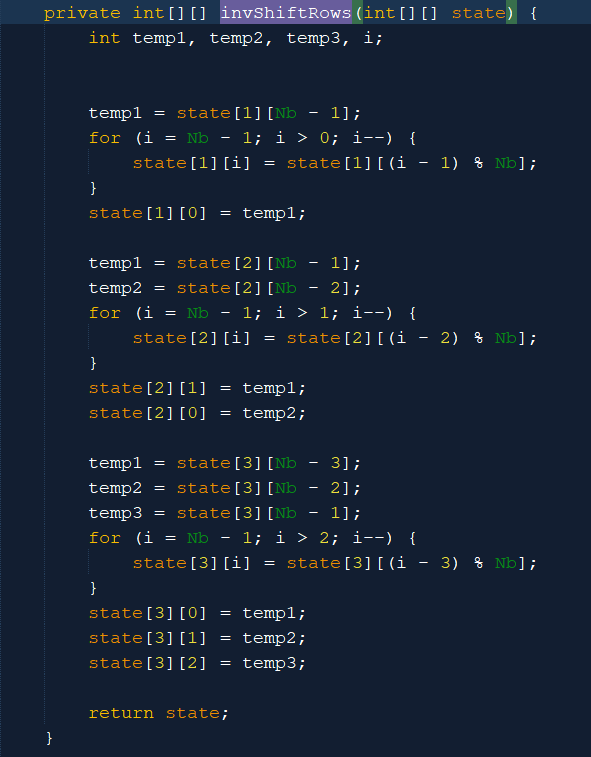
## Cài đặt hàm ShiftRow và InvShifRow

* Hàm shiftRows dịch vòng 3 hàng cuối của state với số lần dịch khác nhau:
* Hàng đầu không dịch
* Hàng 2 dịch vòng trái 1 bit
* Hàng 3 dịch vòng trái 2 bit
* Hàng 4 dịch vòng trái 3 bit



Hình 2. 5 Cài đặt hàm ShiftRow

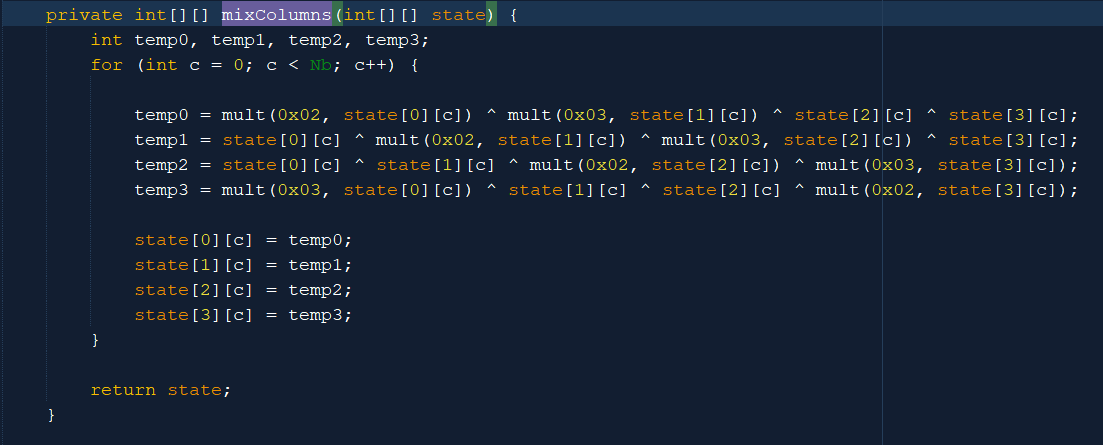
* Hàm invShiftRows là hàm ngược của shiftRows



Hình 2. 6 Cài đặt hàm invShiftRows

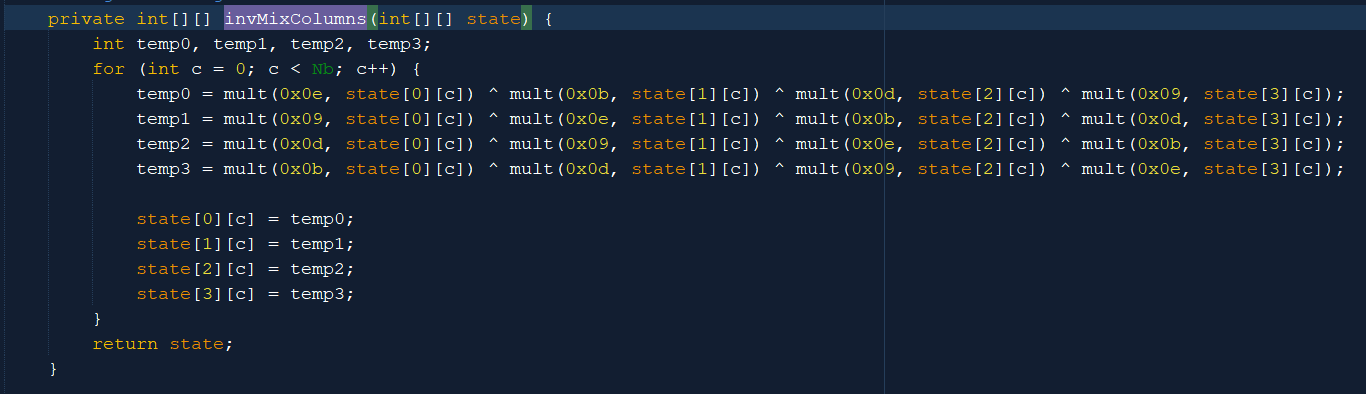
## Cài đặt hàm mixColumnsvà invMixColumns

* Hàm mixColumns nhận tất cả các cột của 1 state và trộn với dữ liệu của nó để nhận 1 cột mới



Hình 2. 7 Cài đặt hàm mixColumns

* Hàm invMixColumns là hàm ngược của hàm mixColumns



Hình 2. 8 Cài dặt hàm invMixColumns

## Cài đặt hàm mã hoá và giải mã (cipher và decipher)

Hàm cipher là hàm mã hóa

* Vòng 0 chỉ thực hiện addRoundKey
* Từ vòng 1 đến vòng Nr -1 thực hiện lần lượt subBytes, shiftRows, mixColumns và addRoundKey
* Vòng cuối thực hiện lần lượt subBytes, shiftRows và addRoundKey



Hình 2. 9 Cài đặt hàm mã hoá (cipher)

Hàm decipher là hàm giải mã

* Vòng 0 chỉ thực hiện addRoundKey
* Từ vòng 1 đến vòng Nr -1 thực hiện lần lượt invShiftRows, invSubBytes, addRoundKey và invMixColumns
* Vòng cuối thực hiện lần lượt invShiftRows, invSubBytes và addRoundKey

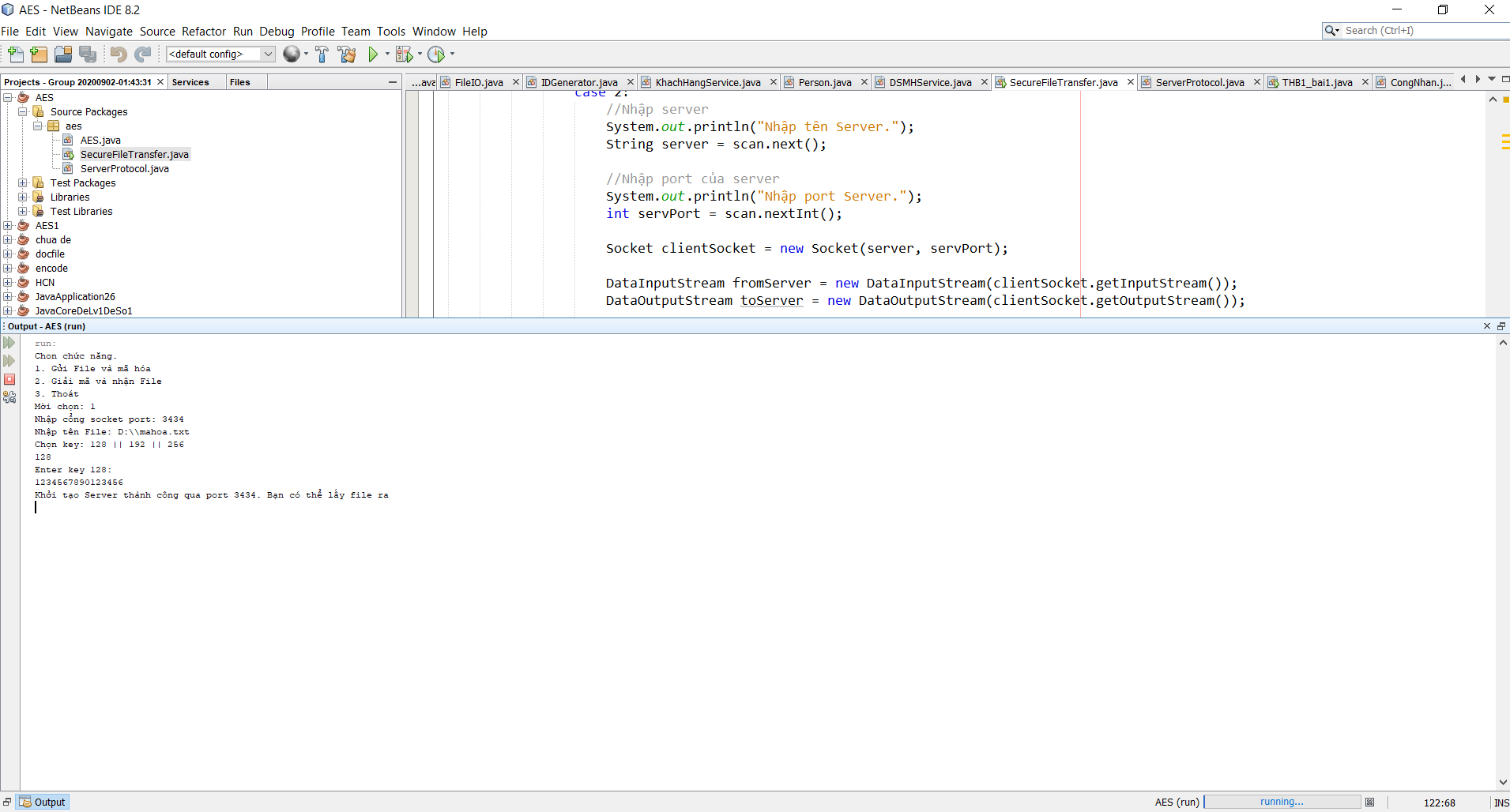


Hình 2. 10 Cài đặt hàm giải mã (decipher)

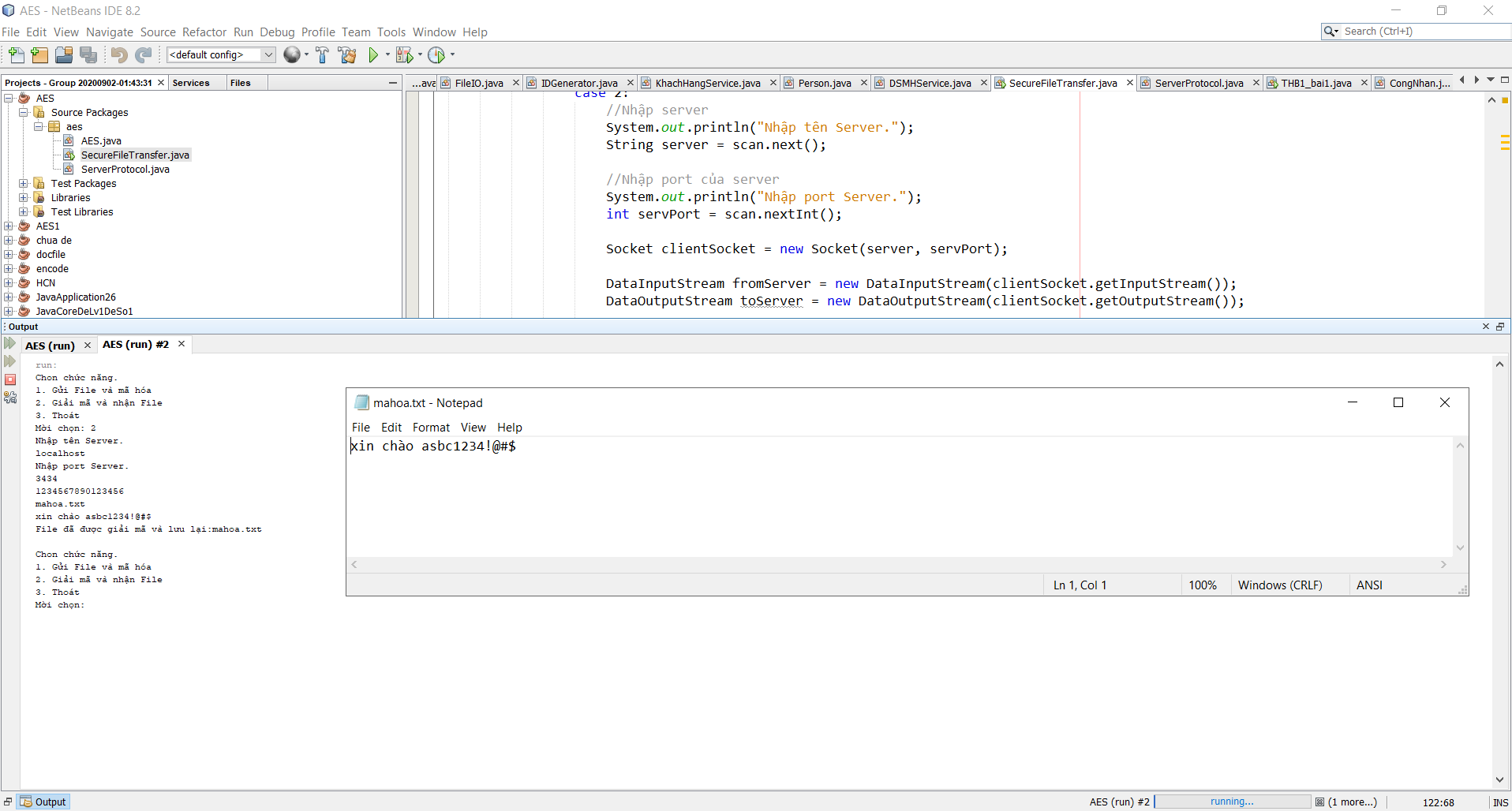
# THỰC NGHIỆM VÀ KẾT LUẬN

## Thực nghiệm

### Thực nghiệm với key 128bit

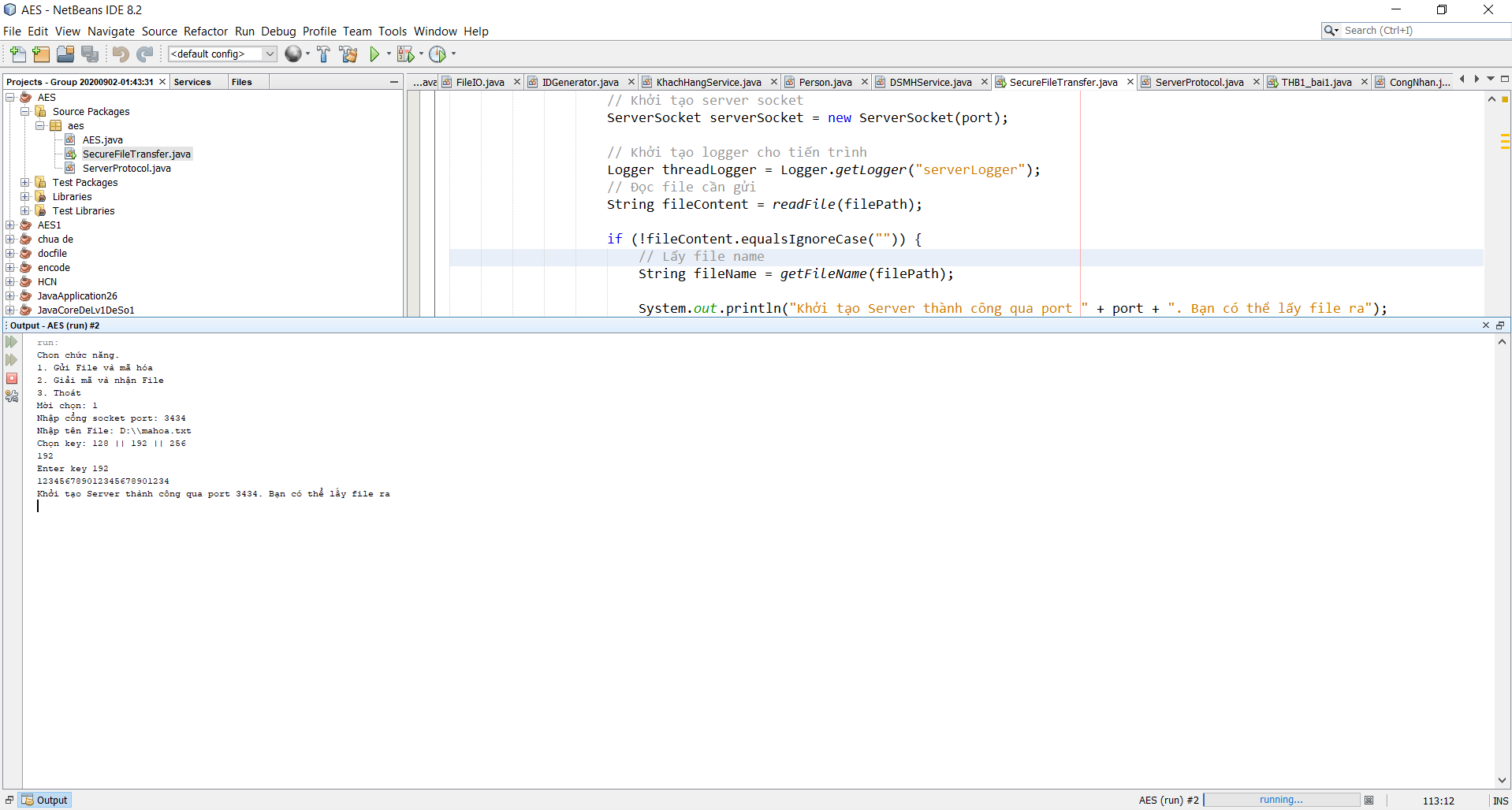


Hình 3. 1 Mã hóa File với key 128bit

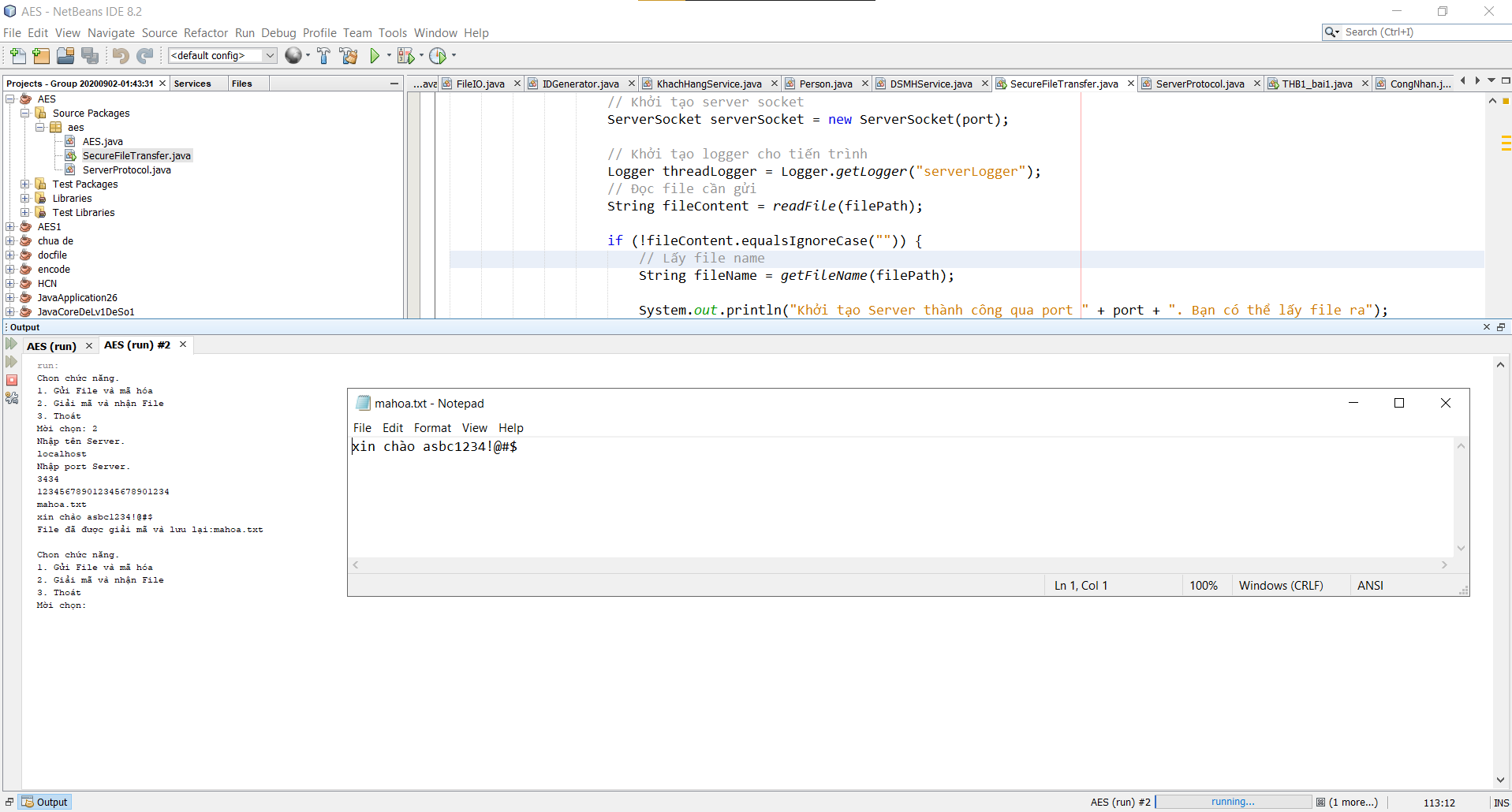


Hình 3. 2 Giải mã ra File với key 128bit

### Thực nghiệm với key 192bit

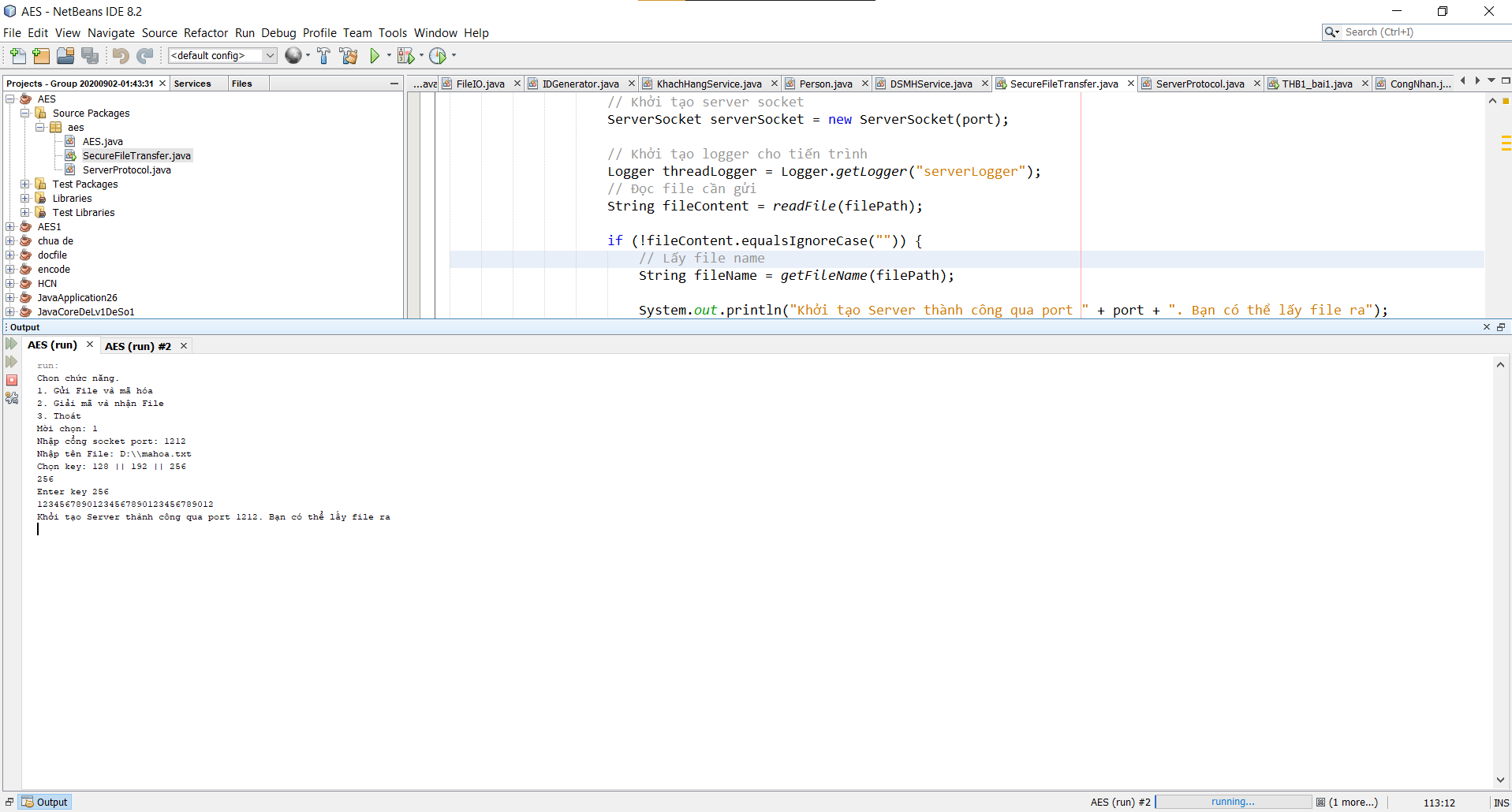


Hình 3. 3 Mã hoá File với key 192bit

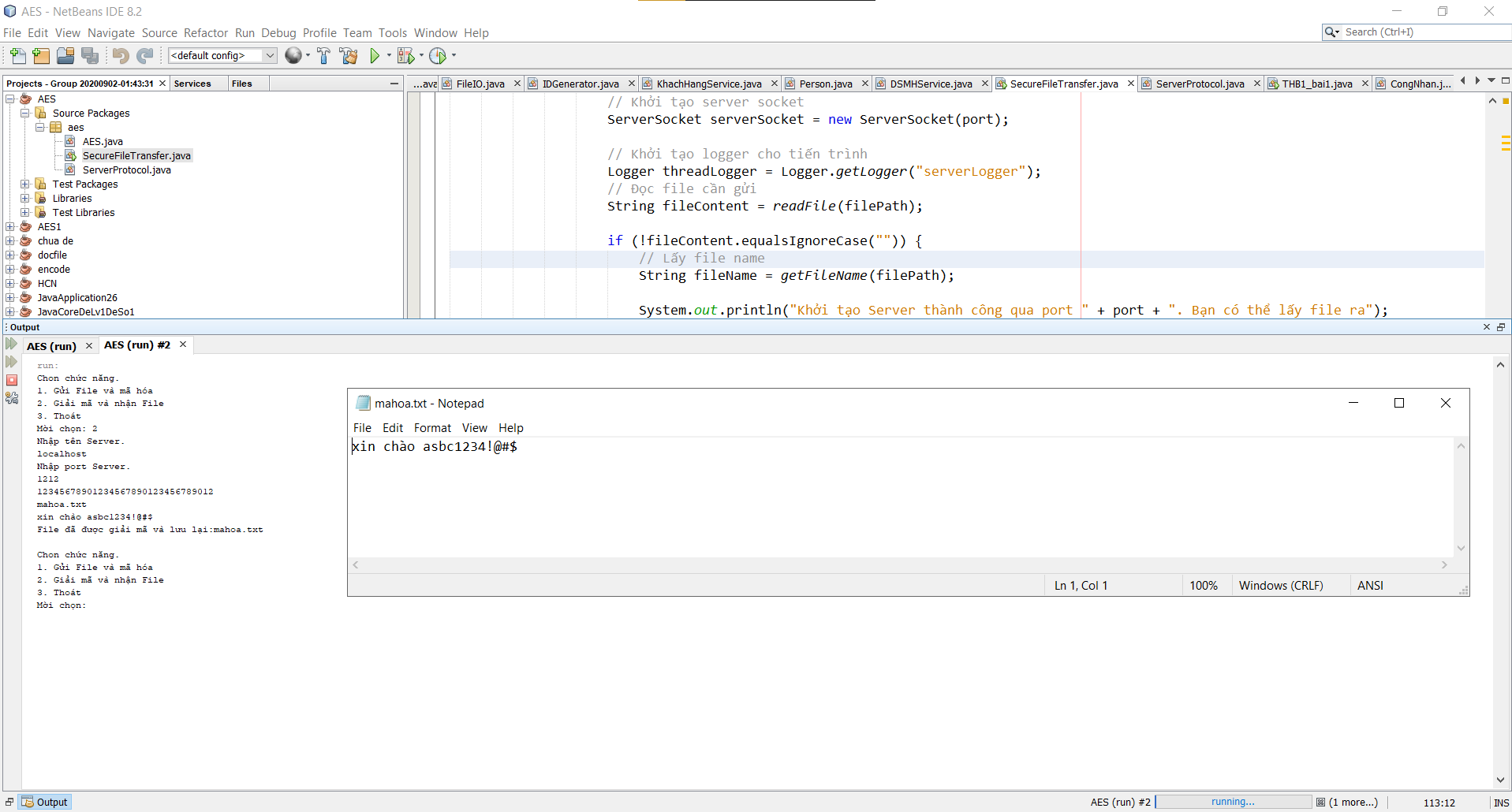


Hình 3. 4 Giải mã File với key 192bit

### Thực nghiệm với key 256bit



Hình 3. 5 Mã hoá File với key 265bit



Hình 3. 6 Giải mã File với key 256bit

## Kết luận và đánh giá

Bài tập môn học đã đạt được những kết quả:

* Tìm hiểu về mật mã AES
* Tìm hiểu về an toàn dữ liệu
* Hoạt động đúng với yêu cầu đề ra

Phương pháp mã ASE

* Phương pháp mã hóa AES đơn giản, có thể thực hiện hiệu quả trên các vi xử lý 8 bit cũng như trên các vi xử lý 32 bit, chỉ dùng phép XOR là chủ yếu
* Thiết kế và độ dài khóa của thuật toán AES (128bit, 192bit, 256bit) là đủ an toàn để bảo vệ các thông tin, được xếp vào loại tối mật nhưng về an ninh của AES thì được đánh giá là chưa cao. Nếu các kỹ thuật tấn công được cải thiện thì AES có thể bị phá vỡ.
* Thuật toán AES cho phép thực hiện hiệu quả bằng cả phần mềm và phần cứng.
* Thông thường với những ứng dụng không yêu cầu cao về hiệu năng và tốc độ thì AES được thực hiện ở dạng phần mềm. Với việc thực hiện trên phần mềm, thuật toán AES có thể được viết bằng nhiều ngôn ngữ lập trình như Assembler, C/C++, Visual Basic, Java, C#... và có thể vận hành trên nhiều hệ điều hành như Windows, Linux/Unix, Solaris....
* Khi thực hiện trên phần cứng, thuật toán AES hỗ trợ thực hiện hai dòng: dòng thiết bị thứ nhất dựa vào một hệ vi xử lý phụ kết hợp với hệ vi xử lý chính của máy tính, dòng thiết bị thứ hai thường được thiết kế ở dạng thẻ thông minh (smart card) hoặc các thiết bị cắm qua cổng USB (Universal Serial Bus)
* Một vấn đề khác nữa là cấu trúc toán học của AES khá đơn giản.

TÀI LIỆU THAM KHẢO

[1] Nghiên cứu về thuật toán mật mã AES và cài đặt mô phỏng AES128

<https://123docz.net/document/4573405-nghien-cuu-thuat-toan-mat-ma-aes-va-cai-dat-mo-phong-aes128.html>

[2] AES Advanced Encryption Standard

<https://www.movable-type.co.uk/scripts/aes.html>

[3] Thuật toán mã hoá AES

<https://youtu.be/S6zlQXj35rI>