**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙧🟉🟉🟉🟉🙥**

****

**TIỂU LUẬN HỌC PHẦN: BẢO MẬT MÁY TÍNH**

**NHÓM 6**

**TÊN ĐỀ TÀI: TÌM HIỂU VỀ GIẢI THUẬT MÃ HÓA AES (CƠ CHẾ HOẠT ĐỘNG CỦA GIẢI THUẬT MÃ HÓA – GIẢI MÃ, CƠ SỞ TOÁN HỌC ĐƯỢC SỬ DỤNG TRONG AES,**

**TP. HỒ CHÍ MINH, THÁNG 4 NĂM 2024**

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙧🟉🟉🟉🟉🙥**

****

**TIỂU LUẬN HỌC PHẦN: BẢO MẬT MÁY TÍNH**

**NHÓM 6**

**TÊN ĐỀ TÀI: TÌM HIỂU VỀ GIẢI THUẬT MÃ HÓA AES (CƠ CHẾ HOẠT ĐỘNG CỦA GIẢI THUẬT MÃ HÓA – GIẢI MÃ, CƠ SỞ TOÁN HỌC ĐƯỢC SỬ DỤNG TRONG AES,**

**TP. HỒ CHÍ MINH, THÁNG 4 NĂM 2024**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DANH SÁCH THÀNH VIÊN** | | | | |
| STT | MSSV | Họ và Tên | Đóng góp | Ghi chú |
| 1 | 2001223740 | Lưu Hoàng Phúc |  |  |
| 2 | 2033220309 | Nguyễn Kim Chí Bảo |  |  |
| 3 | 2001222101 | Đặng Phạm Đăng Khoa |  |  |

//MỤC LỤC

1. **MỞ ĐẦU**
   1. **TẦM QUAN TRỌNG CỦA VIỆC BẢO MẬT THÔNG TIN TRONG THỜI ĐẠI SỐ HÓA**

Trong thời đại công nghệ số, bảo mật thông tin rất quan trọng. Với lượng lớn dữ liệu được tạo ra và chia sẻ trực tuyến, việc bảo vệ thông tin cá nhân và doanh nghiệp là một nhu cầu và trách nhiệm. Không bảo mật thông tin có thể dẫn đến việc tiết lộ thông tin cá nhân, tài khoản ngân hàng, các vấn đề pháp lý, mất lòng tin từ khách hàng và ảnh hưởng xấu đến danh tiếng và hoạt động kinh doanh. Do đó, việc tăng cường nhận thức và sử dụng các biện pháp hiệu quả để bảo vệ thông tin là yếu tố quan trọng để thành công trong thời đại công nghệ số.

* 1. **VAI TRÒ CỦA THUẬT TOÁN MÃ HÓA TRONG VIỆC BẢO MẬT THÔNG TIN**

Thuật toán mã hóa đóng một vai trò cực kỳ quan trọng trong việc bảo vệ thông tin, là một phương pháp hiệu quả để tăng tính riêng tư cho người dùng. Chúng ngăn chặn các hành vi không mong muốn như nghe lén, đánh cắp, thay đổi hay từ chối quyền sở hữu với thông tin/dữ liệu. Bằng cách mã hóa thông tin, chúng ta có thể giữ cho dữ liệu của mình an toàn khỏi những người có ý định xấu, đảm bảo rằng chỉ những người được phép mới có thể truy cập vào dữ liệu.

* 1. **KHÁI QUÁT VỀ MÃ HÓA AES**

AES (viết tắt của Advanced Encryption Standard) là một thuật toán mã hóa khối được chính phủ Hoa Kỳ áp dụng làm tiêu chuẩn mã hóa và được sử dụng rộng rãi trên toàn cầu để bảo vệ dữ liệu nhạy cảm.

Đặc điểm chính:

Mã hóa đối xứng: sử dụng cùng một khóa bí mật để mã hóa và giải mã dữ liệu.

Kích thước khối: 128 bit

Độ dài khóa: 128, 192 hoặc 256 bit

An toàn: được đánh giá cao bởi cộng đồng mật mã học và chưa có phương pháp tấn công hiệu quả nào được công bố.

Hiệu quả: tốc độ mã hóa cao, phù hợp cho nhiều ứng dụng khác nhau.

1. **NỘI DUNG**
   1. **LỊCH SỬ**
   2. **NGUỒN GỐC**

AES (Advanced Encryption Standard) là một thuật toán mã hóa đối xứng, được sử dụng rộng rãi để bảo vệ dữ liệu trên nhiều hệ thống máy tính và mạng khác nhau. AES là một chuẩn mã hóa mà Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) đã công bố vào năm 2001.

* 1. **LÝ DO AES THAY THẾ DES**
* Độ an toàn: DES sử dụng khóa 56 bit, được xem là không còn đủ an toàn do sự phát triển của công nghệ tính toán.

AES sử dụng khóa 128, 192 hoặc 256 bit, mang lại khả năng bảo mật cao hơn nhiều so với DES.

* Hiệu quả:

AES có tốc độ mã hóa nhanh hơn DES, đặc biệt là trên các phần cứng hiện đại.

AES cũng tiết kiệm năng lượng hơn so với DES.

* Linh hoạt:

AES có thể được sử dụng cho nhiều ứng dụng khác nhau, từ mã hóa dữ liệu trên ổ cứng đến bảo mật thông tin liên lạc.

DES có một số hạn chế về khả năng sử dụng, ví dụ như không phù hợp để mã hóa các tập dữ liệu lớn.

* Tiêu chuẩn quốc tế:

AES được Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) công bố là tiêu chuẩn mã hóa vào năm 2001.

DES được NIST khuyến nghị không nên sử dụng cho các ứng dụng mới từ năm 2007.

* Hỗ trợ rộng rãi:

AES được hỗ trợ bởi hầu hết các phần mềm và phần cứng hiện đại.

DES ngày càng ít được hỗ trợ, dẫn đến khó khăn trong việc sử dụng và bảo trì.

* 1. **CÁC KHÁI NIỆM CƠ BẢN**
* ***Mã hóa khối:***

Mã hóa khối là một phương pháp mã hóa dữ liệu trong đó dữ liệu được chia thành các khối có kích thước cố định, sau đó mỗi khối được mã hóa riêng lẻ bằng cách sử dụng cùng một thuật toán và khóa bí mật.

*Ưu điểm:*

Hiệu quả: có thể mã hóa lượng dữ liệu lớn một cách nhanh chóng.

An toàn: có thể sử dụng các thuật toán mã hóa mạnh mẽ để bảo vệ dữ liệu.

Linh hoạt: có thể sử dụng cho nhiều ứng dụng khác nhau.

*Nhược điểm:*

Khóa bí mật cần được bảo mật cẩn thận, nếu bị lộ sẽ dẫn đến nguy cơ mất an toàn dữ liệu.

Không phù hợp để mã hóa các tập dữ liệu nhỏ.

* **Thuật toán Rijndael:**

Rijndael là một thuật toán mã hóa khối được phát triển bởi hai nhà mật mã học người Bỉ Joan Daemen và Vincent Rijmen. Rijndael là thuật toán mã hóa được lựa chọn để trở thành Tiêu chuẩn mã hóa nâng cao (AES) bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) vào năm 2001.

Đặc điểm chính của Rijndael:

* Sử dụng các phép toán đơn giản nhưng hiệu quả để mã hóa dữ liệu.
* Có thể sử dụng với các kích thước khóa và khối khác nhau.
* An toàn cao và chưa có phương pháp tấn công hiệu quả nào được công bố.

Cách thức hoạt động của Rijndael:

* Dữ liệu được chia thành các khối 128 bit.
* Khóa bí mật được sử dụng để biến đổi từng khối dữ liệu qua nhiều vòng lặp.
* Mỗi vòng lặp sử dụng các phép toán logic và toán học phức tạp để thay đổi dữ liệu.
* Kết quả là dữ liệu được mã hóa thành dạng không thể đọc được.
* **Các phép biến đổi trong AES**
* **SubBytes**

Phép biến đổi dùng trong phép mã hóa áp dụng lên trạng thái (kết quả mã hóa trung gian, được mô tả dưới dạng một mảng chữ nhật của các byte) sử dụng một bảng thay thế byte phi tuyến (Hộp S – bảng thay thế phi tuyến, được sử dụng trong một số phép thay thế byte và trong quy trình mở rộng khóa, nhằm thực hiện một phép thay thế 1-1 đối với giá trị mỗi byte) trên mỗi byte trạng thái một cách độc lập.

* **ShiftRows**

Phép biến đổi dùng trong phép mã hóa áp dụng lên trạng thái bằng cách chuyển dịch vòng ba hàng cuối của trạng thái theo số lượng byte các offset khác nhau.

* **MixColumns**

Phép biến đổi trong phép mã hóa thực hiện bằng cách lấy tất cả các cột trạng thái trộn với dữ liệu của chúng (một cách độc lập nhau) để tạo ra các cột mới.

* **AddRoundKey**

Phép biến đổi trong phép mã hóa và phép giải mã. Trong đó, một khóa vòng (các giá trị sinh ra từ khóa mã bằng quy trình mở rộng khóa) được cộng thêm vào trạng thái bằng phép toán XOR (phép toán hoặc và loại trừ). Độ dài của khóa vòng bằng độ dài của trạng thái.

* **Mở rộng khóa**

Thuật toán AES nhận vào một khóa mã K và thực hiện phép mở rộng khóa để tạo ra một lược đồ khóa. Phép mở rộng khóa tạo ra tổng số Nb(Nr+1) từ. Thuật toán yêu cầu một tập khởi tạo gồm Nb từ và mỗi trong số Nr vòng đòi hỏi Nb từ làm dữ liệu khóa đầu vào. Lược đồ khóa kết quả là một mảng tuyến tính các từ 4 byte.

* **Phép giải mã**

Các phép biến đổi trong phép mã hóa có thể được đảo ngược và sau đó thực hiện theo chiều ngược lại nhằm tạo ra phép giải mã trực tiếp của thuật toán AES. Các phép biến đổi sử dụng trong phép giải mã gồm: InvShiftRows(), InvSubBytes(), InvMixColumns() và AddRoundKey().

* **InvSubBytes**

Phép biến đổi InvSubBytes là nghịch đảo của phép thay thế theo byte SubBytes(), trong đó sử dụng một bảng chọn S nghịch đảo áp dụng cho mỗi byte của trạng thái.

* **InvShiftRows**

Phép biến đổi InvShiftRows là phép biến đổi ngược của ShiftRows(). Các byte trong ba từ cuối của trạng thái được dịch vòng theo số byte khác nhau. Ở hàng đầu tiên (r=0) không thực hiện phép chuyển dịch, ba hàng dưới cùng được dịch vòng Nb-shift(r,Nb) byte.

* **InvMixColumns**

Phép biến đổi InvMixColumns là phép biến đổi ngược của MixColumns(). Nó thao tác theo từng cột của trạng thái, xem mỗi cột như một đa thức bốn hạng tử.

* **Biến đổi nghịch AddRoundKey**

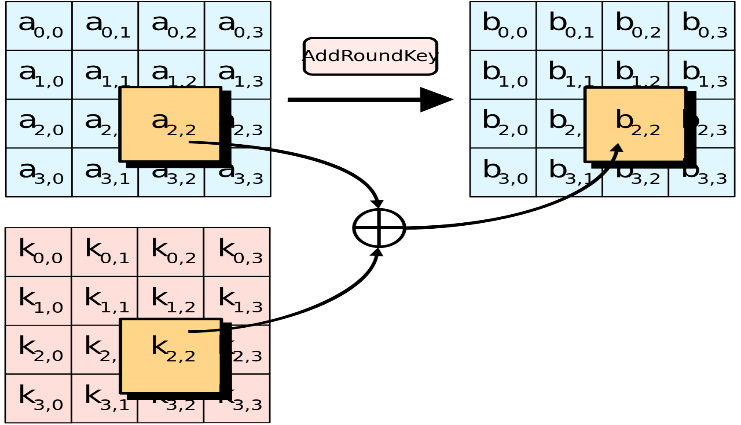
Phép biến đổi AddRoundKey là phép biến đổi thuận nghịch vì nó chỉ áp dụng một phép toán XOR nên nó được thực hiện như nhau ở cả phép mã hóa và phép giải mã.

* 1. **NGUYÊN LÝ HOẠT ĐỘNG**
     1. **Mã hóa**
* **Hàm AddRoundKey().**

Hàm Addroundkey() là một phần thiết yếu của thuật toán mã hóa AES. Chức năng chính của hàm này là thêm vòng khóa vào khối dữ liệu đang được mã hóa.

Quá trình hoạt động của hàm Addroundkey():

1. Chia khối dữ liệu thành ma trận: Dữ liệu đầu vào được chia thành một ma trận theo kích thước được xác định bởi độ dài của nó (ví dụ: 4x4 cho khối 128 bit).
2. Chia key thành ma trận: Tương tự, key cũng được chia thành một ma trận có kích thước tương ứng với ma trận dữ liệu.
3. Thực hiện phép toán XOR: Mỗi phần tử trong ma trận dữ liệu được thực hiện phép toán XOR bitwise với phần tử tương ứng trong ma trận của key. Phép toán này kết hợp giá trị dữ liệu và giá trị key.
4. Lặp lại cho các vòng tiếp theo: Quá trình thêm vòng khóa và thay đổi trạng thái được lặp lại cho số vòng được xác định bởi độ dài khóa (10 vòng cho AES-128, 12 vòng cho AES-192 và 14 vòng cho AES-256).



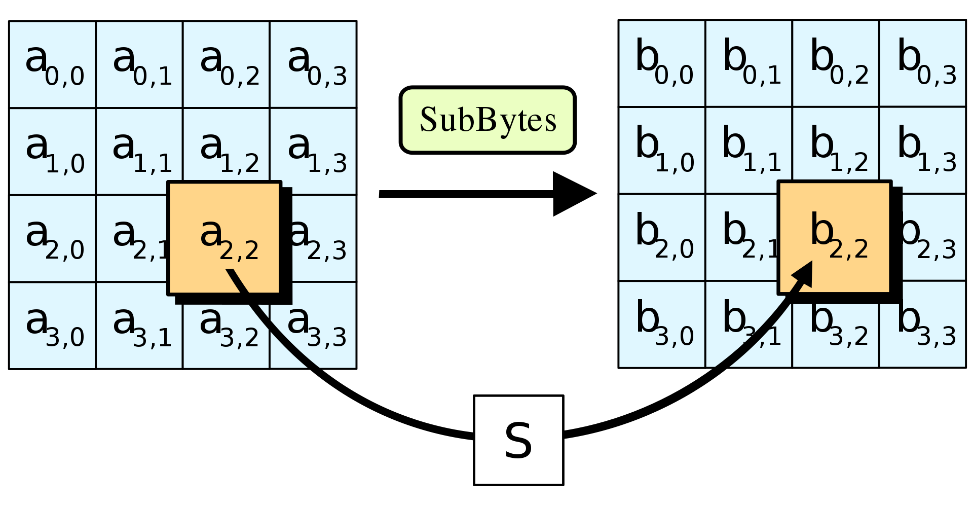
*Hình 1: AddRoundKey()*

* **Hàm SubBytes()**

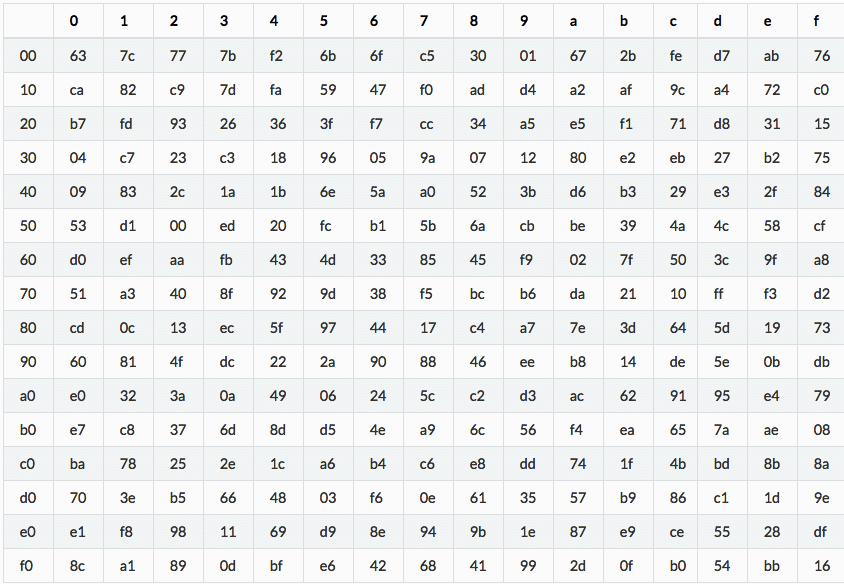
Hàm SubBytes() là một thành phần quan trọng trong thuật toán mã hóa AES, đóng vai trò thiết yếu trong việc thay thế mỗi byte trong khối dữ liệu bằng một giá trị mới dựa trên bảng chọn S. Quá trình này góp phần tăng cường độ bảo mật và chống lại các phân tích mật mã bằng cách phi tuyến tính hóa dữ liệu.

Cơ chế hoạt động của hàm SubBytes():

1. Chia khối dữ liệu thành các byte: Khối dữ liệu đầu vào được chia thành các byte riêng biệt.
2. Tra bảng chọn S: Mỗi byte được sử dụng làm chỉ mục để tra cứu giá trị tương ứng trong bảng chọn S. Bảng chọn Slà một bảng 256x4 chứa 256 giá trị 4 bit, được thiết kế cẩn thận để mang lại tính phi tuyến tính và khả năng chống phân tích mật mã cao.
3. Thay thế byte: Giá trị tra cứu được lấy từ bảng chọn S thay thế cho byte ban đầu.
4. Lặp lại cho tất cả các byte: Quá trình tra cứu bảng chọn Svà thay thế byte được lặp lại cho tất cả các byte trong khối dữ liệu.



*Hình 2: Hàm SubBytes()*

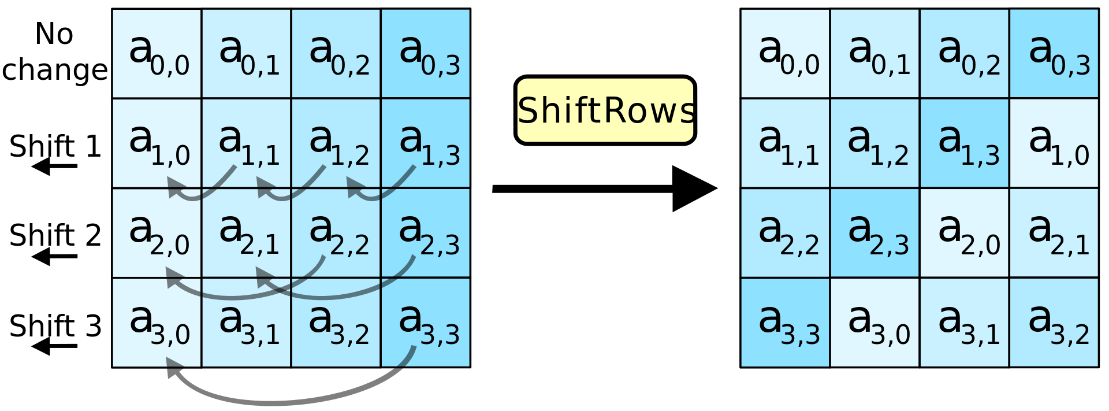


* *Hình 3: Bảng chọn S*
* **2.3. Hàm ShiftRows().**

Hàm ShiftRows() là một thành phần quan trọng trong thuật toán mã hóa AES, đóng vai trò thiết yếu trong việc hoán vị các bytes trong mỗi hàng của khối dữ liệu theo một quy tắc cố định.

Cơ chế hoạt động của hàm ShiftRows():

1. Chia khối dữ liệu thành ma trận: Khối dữ liệu đầu vào được chia thành một ma trận theo kích thước được xác định bởi độ dài khối (ví dụ: 4x4 cho khối 128 bit).
2. Hoán vị từng hàng: Mỗi hàng trong ma trận được hoán vị theo một quy tắc cố định. Quy tắc hoán vị được xác định bởi vị trí hàng và số vòng trong thuật toán.
   * Hàng 1: Không thay đổi.
   * Hàng 2: Dịch sang trái 1 byte.
   * Hàng 3: Dịch sang trái 2 byte.
   * Hàng 4: Dịch sang trái 3 byte.
3. Tạo ma trận mới: Sau khi hoán vị tất cả các hàng, một ma trận mới được tạo thành với các byte được sắp xếp lại theo quy tắc hoán vị.



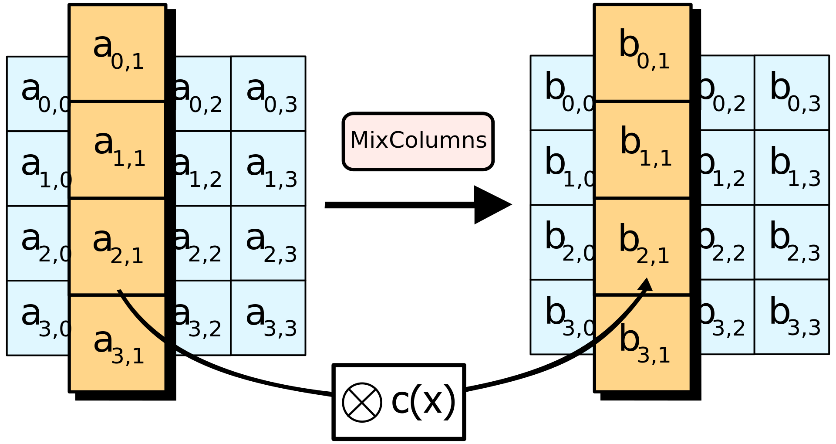
*Hình 4: Hàm ShiftRows().*

* **Hàm MixColumns()**

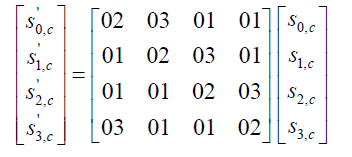
Cơ chế hoạt động của hàm MixColumns() trong thuật toán mã hóa AES**:** Hàm MixColumns() là một thành phần quan trọng trong thuật toán mã hóa AES, đóng vai trò thiết yếu trong việc trộn các cột trong khối dữ liệu theo một phép biến đổi tuyến tính cố định. Quá trình trộn này góp phần tăng cường độ bảo mật và chống lại các phân tích mật mã bằng cách làm cho mối quan hệ giữa các byte trong mỗi cột trở nên phức tạp hơn.

Cơ chế hoạt động của hàm MixColumns():

1. Chia khối dữ liệu thành cột: Khối dữ liệu đầu vào được chia thành các cột theo kích thước được xác định bởi độ dài khối (ví dụ: 4 cột cho khối 128 bit).
2. Áp dụng phép biến đổi tuyến tính: Mỗi cột được nhân với một ma trận cố định (MixColumns matrix) theo phép nhân ma trận. Ma trận MixColumns được thiết kế cẩn thận để đảm bảo tính phi tuyến tính và khả năng khuếch tán cao.
3. Thay thế cột: Cột kết quả sau phép nhân ma trận thay thế cho cột ban đầu.
4. Lặp lại cho tất cả các cột: Quá trình áp dụng phép biến đổi tuyến tính và thay thế cột được lặp lại cho tất cả các cột trong khối dữ liệu.



*Hình 5: Hàm MixColums()*



*Hình 6:Ma trận cố định (MixColumns matrix)*

Lưu ý: Trong hàm MixColumns có 2 phép toán phức tạp cần ghi nhớ đó là phép nhân 2 ma trận và phép chia trên bits. Chúng ta sẽ nói về 2 phép toán đó ở phần dưới đây.

* **Phép nhân 2 ma trận.**

Giả sử chúng ta có hai ma trận 4x4:

Ma trận A:

Ảnh có chứa văn bản, Phông chữ, thuật in máy, ảnh chụp màn hình

Mô tả được tạo tự động

Ma trận B:

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, thuật in máy

Mô tả được tạo tự động

Kích thước của ma trận A là (4x4) và kích thước của ma trận B là (4x4). Do đó, hai ma trận này có thể được nhân với nhau.

Quy tắc thực hiện phép nhân:

Để tính toán phần tử (i, j) của ma trận kết quả (C), ta thực hiện phép nhân vectơ hàng thứ i của ma trận A với vectơ cột thứ j của ma trận B và cộng tích các kết quả thu được.

C[i, j] = Σ(k = 1 to n) A[i, k] \* B[k, j]

Để tính toán phần tử (1, 1) của ma trận kết quả, ta thực hiện:

C[1, 1] = a11 \* b11 + a12 \* b21 + a13 \* b31 + a14 \* b41

Tương tự, để tính toán phần tử (2, 2) của ma trận kết quả, ta thực hiện:

C[2, 2] = a21 \* b12 + a22 \* b22 + a23 \* b32 + a24 \* b42

Áp dụng quy tắc này cho tất cả các phần tử còn lại của ma trận kết quả, ta thu được ma trận kết quả C.

* **Phép chia trên bits.**

Sau khi thực hiện phép nhân ma trận nếu kết quả lớn hơn bits thì ta sẽ thực hiện phép chia trên bits.

VD:

Ta có:

{57}x{83}= lớn hơn bits

Ta sẽ chia nó cho

Thì ta sẽ có kết quả là:

* **Thuật toán mở rộng khóa (KeyExpansion).**

Ảnh có chứa văn bản, biểu đồ, Kế hoạch, trò chơi giải ô chữ

Mô tả được tạo tự độngThuật toán KeyExpansion trong AES đóng vai trò quan trọng trong việc mở rộng khóa ban đầu thành các khóa con được sử dụng trong các vòng lặp của quá trình mã hóa và giải mã. Quá trình mở rộng khóa này đảm bảo tính bảo mật và chống lại các phương pháp tấn công giải mã bằng cách tạo ra nhiều khóa con khác nhau từ một khóa ban đầu duy nhất.

*Hình 7: Lưu đồ giải thuật của mở rộng khóa*

Quy trình mở rộng khóa(với AES-128):

1. Khóa ban đầu: Khóa ban đầu được chia thành các cột 4 byte.
2. Tìm w4: w4 sẽ bằng hàm g() XOR với w0.
3. Tìm các w còn lại: w5 sẽ bằng w1 XOR w4, w6 bằng w2 XOR w5,.. cho tới hế w47.

Tìm hàm g:

Đầu tiên bên trong hàm g() ta sẽ thấy một giá trị là Rcon là một word (4 bytes) Rcon[j] = (RC[j], 0, 0, 0), với RC[1] = 1, RC[j] = 2\*RC[j-1]. Các giá trị của RC[j] trong hệ cơ số 16 là:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| RC[j] | 01 | 02 | 04 | 08 | 10 | 20 | 40 | 80 | 1B | 36 |

Các bước tìm hàm g():

1. Đầu tiên ta sẽ lấy các giá trị của w3 và chia nó làm bốn phần lần lược là: B0, B1, B2, B3.

2. Tiếp đến ta thực hiện dịch sang trái 1 bytes.

3. Sau đó ta sẽ thực hiện hàm SubBytes() như đã nói bên trên.

4. Cuối cùng ta lấy kết quả của hàm SubBytes() đem XOR với Rcon.

5. Như vậy ta đã có key cho một vòng, mỗi vòng thực hiện tương tự để có thể tìm được khóa con.

* + 1. **Giải mã**

3.1. Hàm AddRoundKey().

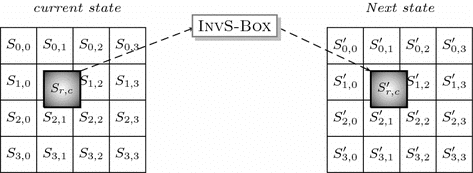
Nguyên tắc hoạt động tương tự với hàm AddRoundKey() ở phần mã hóa.

3.2 Hàm InvSubBytes().

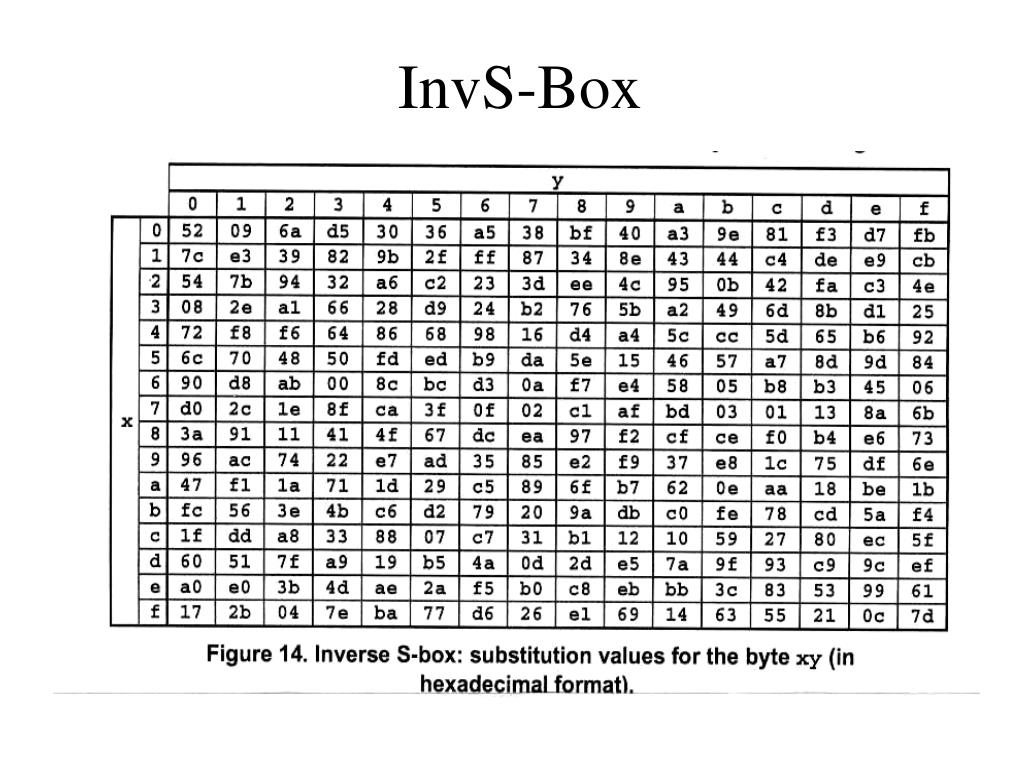
Hàm InvSubBytes() hoạt động tương tự như hàm SubBytes(), nhưng sử dụng một bảng tra cứu ngược (S-box nghịch đảo) để thay thế mỗi byte trong dữ liệu được mã hóa. Bảng S-box nghịch đảo được thiết kế để đảo ngược các phép thay thế byte được thực hiện trong hàm SubBytes().

Quy trình hoạt động:

1. Đọc byte: Hàm InvSubBytes() đọc từng byte trong dữ liệu được mã hóa.
2. Tra cứu bảng chọn S nghịch đảo: Mỗi byte được tra cứu trong bảng chọn Snghịch đảo để tìm giá trị tương ứng.
3. Thay thế byte: Giá trị tìm được trong bảng chọn Snghịch đảo được sử dụng để thay thế byte ban đầu trong dữ liệu được mã hóa.
4. Lặp lại: Quy trình được lặp lại cho tất cả các byte trong dữ liệu được mã hóa.



*Hình 8: Hàm InvSubBytes().*



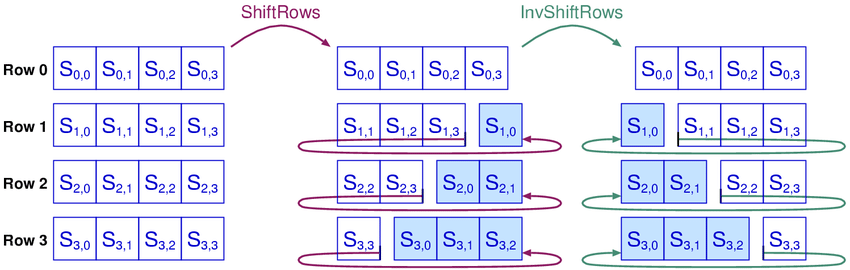
*Hình 9: Bảng chọn S nghịch đảo*

3.3. Hàm InvShiftRows().

Hàm InvShiftRows() hoạt động ngược lại với hàm ShiftRows(). Trong khi ShiftRows() dịch chuyển các hàng trong ma trận trạng thái sang trái, InvShiftRows() dịch chuyển các hàng sang phải để đưa ma trận về lại trạng thái ban đầu.

Quy trình hoạt động:

1. Đọc ma trận: Hàm InvShiftRows() đọc ma trận trạng thái được mã hóa.
2. Dịch chuyển hàng: Mỗi hàng trong ma trận được dịch chuyển sang phải một số vị trí nhất định:
   * Hàng 1: Dịch chuyển sang phải 0 vị trí.
   * Hàng 2: Dịch chuyển sang phải 1 vị trí.
   * Hàng 3: Dịch chuyển sang phải 2 vị trí.
   * Hàng 4: Dịch chuyển sang phải 3 vị trí.
3. Ghi ma trận: Ma trận trạng thái sau khi dịch chuyển được ghi lại.



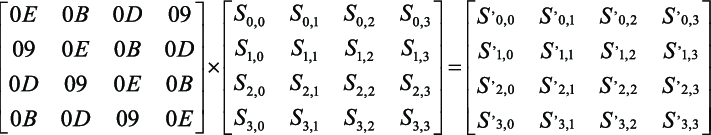
*Hình 10: Hàm InvShiftRows()*

3.4. Hàm InvMixColumns().

Hàm InvMixColumns() thực hiện phép toán ngược của phép trộn cột được sử dụng trong hàm MixColumns(). Phép toán này sử dụng một ma trận cơ sở cố định và phép toán ma trận để biến đổi mỗi cột trong ma trận trạng thái.

Quy trình hoạt động:

1. Đọc cột: Hàm InvMixColumns() đọc từng cột trong ma trận trạng thái được mã hóa.
2. Nhân ma trận cơ sở: Mỗi cột được nhân với ma trận cơ sở ngược (được tính toán từ ma trận cơ sở sử dụng phép toán ma trận ngược).
3. Ghi cột: Kết quả sau khi nhân được ghi lại vào cột tương ứng trong ma trận trạng thái.



*Hình 11: Hàm InvMixCokumns().*

* 1. **ƯU VÀ NHƯỢC ĐIỂM**

**ƯU**

* Tính bảo mật cao
* Tốc độ mã hóa nhanh
* Hiệu quả sử dụng bộ nhớ
* Khả năng linh hoạt
* Hỗ trợ đa nền tảng

**NHƯỢC**

* Khả năng bị tấn công bằng các phương pháp tiên tiến
* Yêu cầu phần cứng mạnh để xử lý

**Ứng dụng của AES**

Liệt kê các lĩnh vực ứng dụng phổ biến của AES:

* Bảo mật dữ liệu trong ổ cứng, USB
* Mã hóa thông tin liên lạc
* Bảo mật dữ liệu trong các hệ thống ngân hàng, tài chính
* Mã hóa email, chữ ký điện tử
* Bảo mật dữ liệu y tế
* Ứng dụng trong các phần mềm, ứng dụng di động

1. **KẾT QUẢ THUẬT TOÁN**
2. **KẾT LUẬN**
3. **TÀI LIỆU THAM KHẢO**