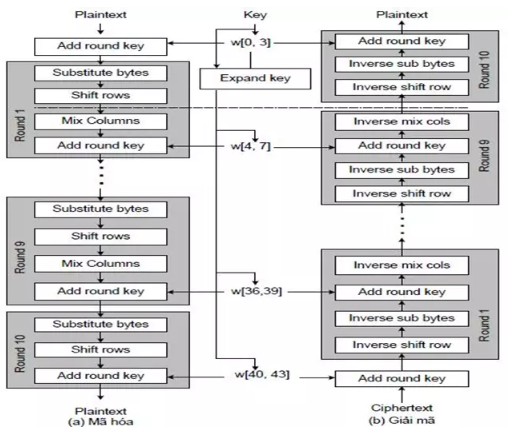
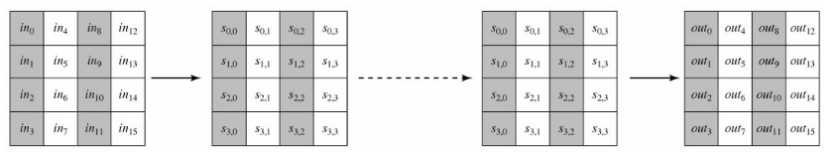
**2.3. Mã hóa và giải mã AES**

Sơ đồ tổng quát của mã hóa và giải mã AES:



*Hình ?: Sơ đồ tổng quát mã hóa và giải mã AES*

Đầu vào cho thuật toán mã hóa và giải mã là một khối 128 bit, khối bit này được mô tả là một ma trận vuông, mỗi ô là 1 byte. Khối này được sao chép vào một mảng trạng thái, được sửa đổi ở mỗi giai đoạn mã hóa hoặc giải mã. Sau giai đoạn cuối cùng, mảng trạng thái này được sao chép vào một ma trận đầu ra.



*Hình ?: Đầu vào, mảng trạng thái và đầu ra*

Tương tự, khóa 128 bit được mô tả như một ma trận vuông, mỗi phần tử là 1 byte. Khóa này sau đó được mở rộng thành một mảng các từ (word), mỗi từ là 4 byte và tổng chiều dài khóa là 44 từ cho khóa 128 bit.



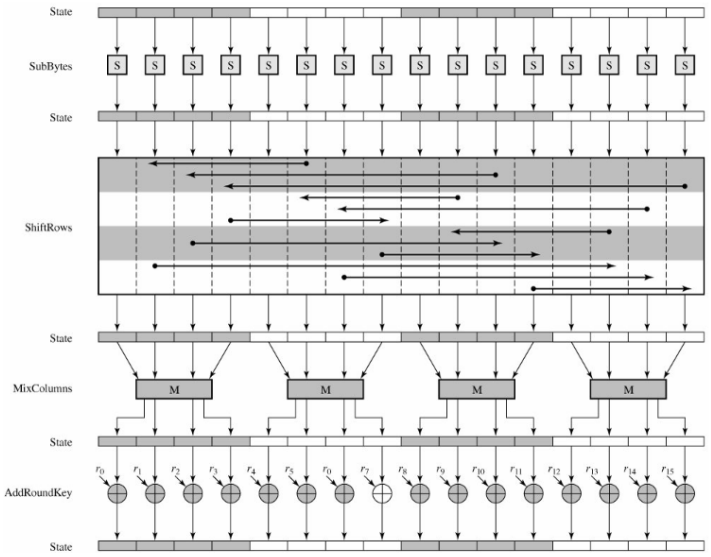
*Hình ?: Khóa và mở rộng khóa*

Lưu ý rằng thứ tự của các byte trong ma trận là theo cột. Vì vậy, 4 byte đầu tiên của bản rõ 128 bit đầu vào chiếm cột đầu tiên của ma trận, bốn byte thứ hai chiếm cột thứ hai, … Tương tự, 4 byte đầu tiên của khóa mở rộng tạo thành một từ, chiếm cột đầu tiên của ma trận.

Cấu trúc của thuật toán AES tương đối đơn giản. Cả thuật toán mã hóa và giải mã đều bắt đầu giai đoạn AddRoundKey, tiếp theo là 9 vòng, mỗi vòng đầy đủ 4 giai đoạn:

* Thay thế các bytes (Substitute bytes): Sử dụng hộp S để thực hiện việc thay thế từng byte của khối.
* Dịch các dòng (ShiftRows): Thực hiện hoán vị.
* Trộn cột (MixColumns): Phép thay thế sử dụng các phép toán số học trên Z256.
* AddRoundKey: Phép XOR của khối hiện tại với một phần của khóa được mở rộng.

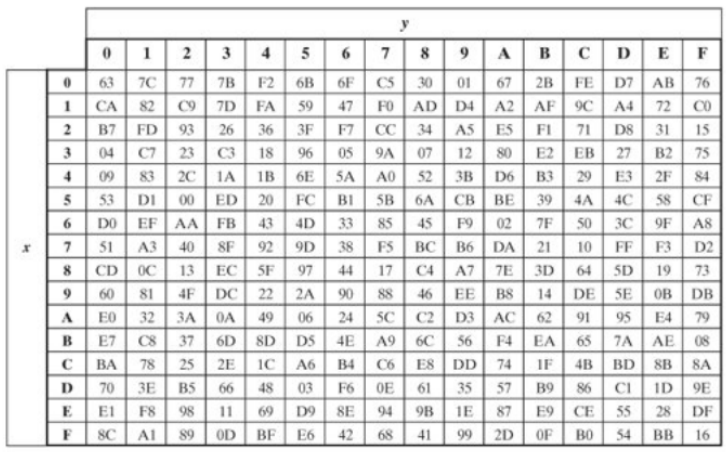
Vòng cuối cùng chỉ có 3 giai đoạn.



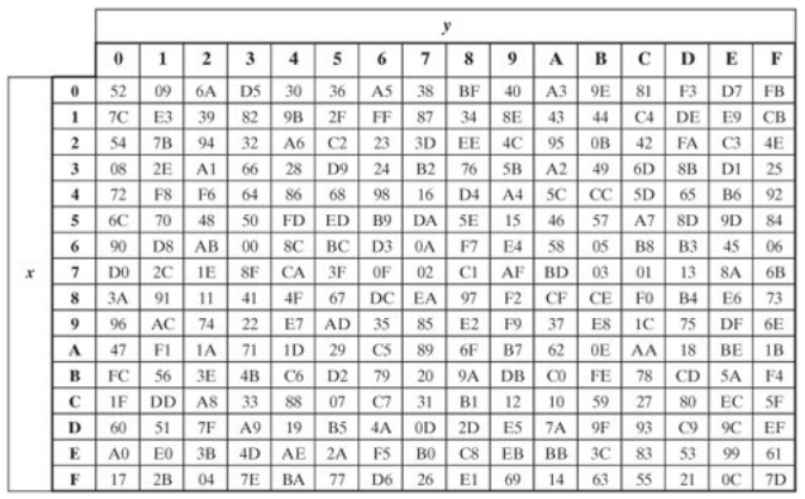
*Hình ?: Minh họa một vòng mã AES*

**2.3.1. Thay thế byte (Substitute bytes)**

Thay thế byte đơn giản chỉ là tra cứu trong bảng 16 x 16, mỗi ô là 1 byte và được gọi là hộp S và hộp S đảo.

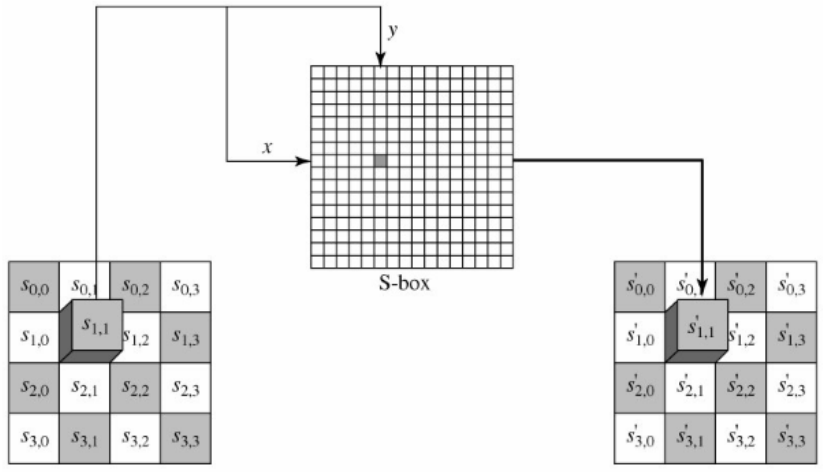


*Hình ?: Hộp S*



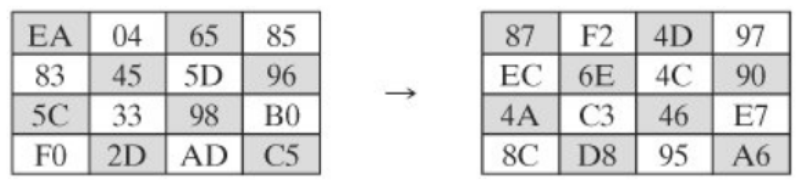
*Hình ?: Hộp S đảo*

Minh họa việc tra cứu hộp S:



*Hình ?: Phép thay thế byte sử dụng hộp S*

Ví dụ minh họa phép thay thế byte:

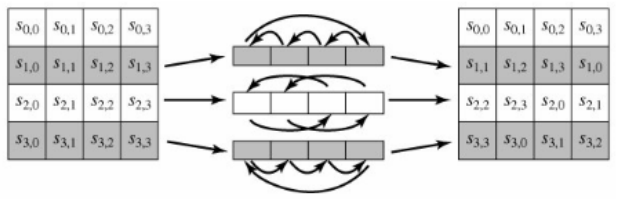


*Hình ?: Minh họa phép thay thế byte*

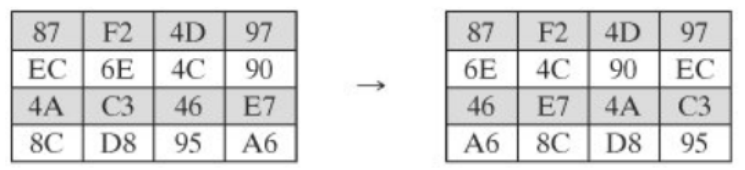
Để tìm byte thay thế của byte EA, ta tra dòng E và cột A trong hộp S thu được byte 87. Như vậy, byte EA sẽ được thay thế bằng byte 87. Tương tự, byte 04 ta tra dòng 0 và cột 4 thu được F2. Ta làm tương tự cho các byte còn lại sẽ thu được ma trận kết quả sau khi thực hiện phép thay thế.

**2.3.2. Dịch dòng (ShiftRows)**

Minh họa phép dịch dòng. Dòng đầu tiên của ma trận trạng thái được giữ nguyên, dòng thứ hai dịch trái 1 byte, dòng thứ 3 dịch trái 2 byte và dòng cuối cùng dịch trái 3 byte.



*Hình ?: Minh họa phép dịch dòng*

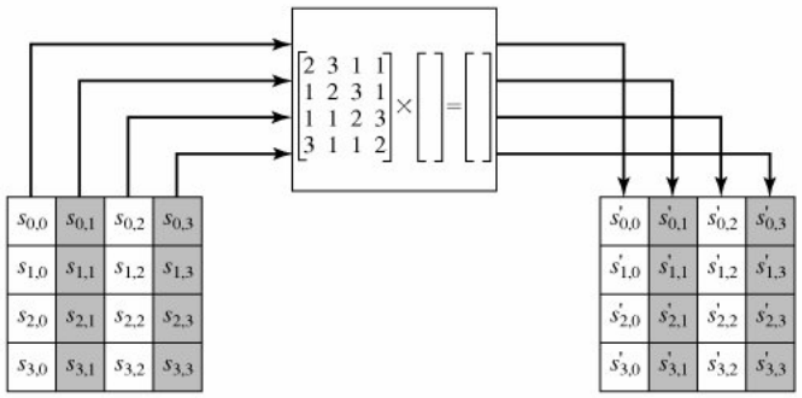


*Hình ?: Ví dụ minh họa phép dịch dòng*

Đối với thuật toán giải mã ta sử dụng phép dịch dòng ngược. Tức là, dòng đầu tiên của ma trận trạng thái giữ nguyên, dòng thứ 2 dịch phải 1 byte, dòng thứ 3 dịch phải 2 byte và dòng cuối cùng dịch phải 3 byte.

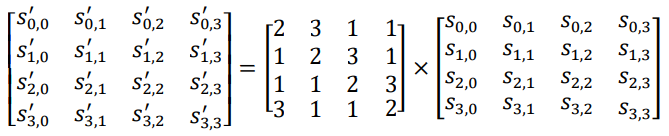
**2.3.3. Trộn cột (MixColumns)**

Minh họa phép trộn cột:

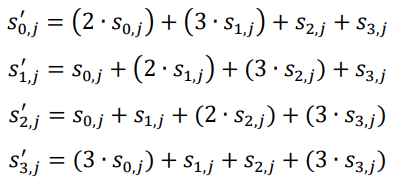


*Hình ?: Minh họa phép trộn cột*

Như vậy, kết quả của phép trộn cột sẽ được xác định theo công thức sau:

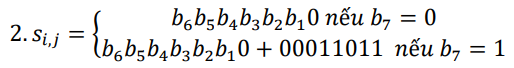


Áp dụng phép nhân hai ma trận ta thu được:



Trong đó, phép nhân (.) được thực hiện theo luật:

* Giả sử si,j được biểu diễn dưới dạng 8 bít b7b6b5b4b3b2b1b0.
* Khi nhân với 2 sẽ được thực hiện theo công thức sau:

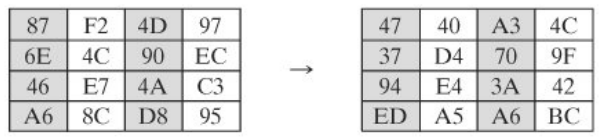


* Khi nhân với 3 sẽ được thực hiện theo công thức sau:



Phép cộng (+) trong các công thức trên là phép XOR bit.

Ví dụ minh họa phép trộn cột:



*Hình ?: Ví dụ minh họa phép trộn cột*

Diễn giải cách xác định phần tử đầu tiên trong ma trận sau khi thực hiện phép trộn cột:

s’00 = 2.(87) + 3.(6E) + 46 + A6

Chuyển các số từ hệ 16 sang hệ 2:

87h = 10000111

(b7 = 1 nên 2.(87) = 00001110 XOR 00011011 = 00010101)

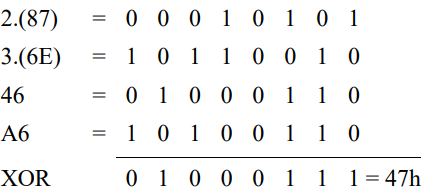
6Eh = 01101110

(b7 của 6E là 0 nên 2.(6E) = 11011100.

Do đó, 3.(6E) = 01101110 XOR 11011100 = 10110010)

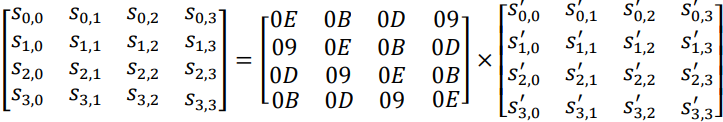
46h = 01000110

A6h = 10100110

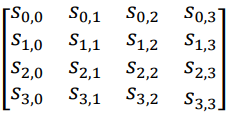


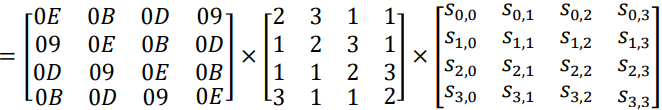
Tính toán tương tự cho các phần tử còn lại ta thu được trạng thái sau khi thực hiện phép trộn cột.

Phép chuyển đổi đảo trộn cột (Inverse mix column transform) trong thuật toán giải mã được thực hiện như sau:

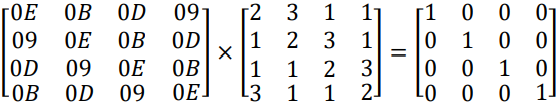


Thay thế công thức của phép trộn cột vào thì ta thu được công thức sau:





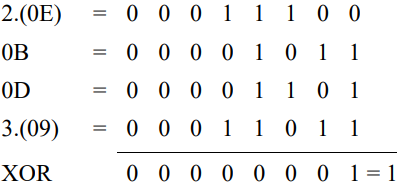
Như vậy thì công thức sau phải được thỏa mãn:



Chứng minh phần tử đầu tiên thỏa mãn yêu cầu:

2.(0E) + 0B + 0D + 3.(09)

= 00011100 + 00001011 + 00001101 + 3.(09)

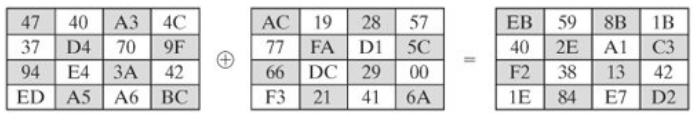


Tính toán tương tự cho các phần tử còn lại.

**2.3.4. Cộng với khóa (Add round key)**

Phép cộng với khóa là thực hiện phép XOR bit của 128 bit của ma trận trạng thái và 128 bit của khóa tương ứng của vòng.

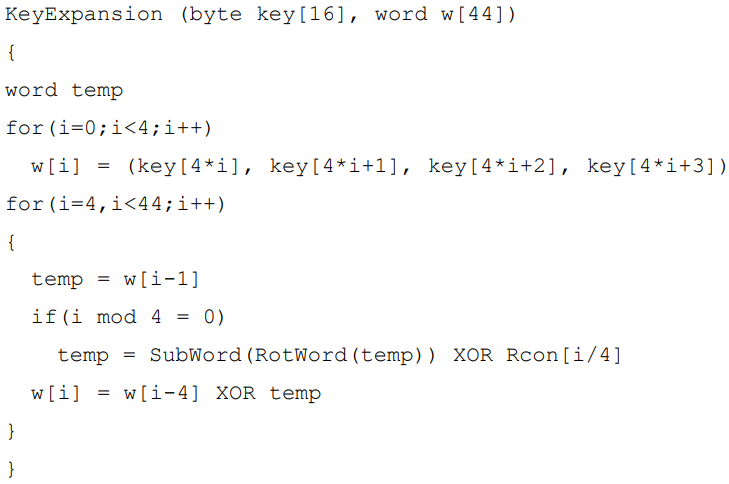
Ví dụ minh họa phép cộng khóa, ma trận đầu tiên là trạng thái và ma trận thứ 2 là khóa của vòng:



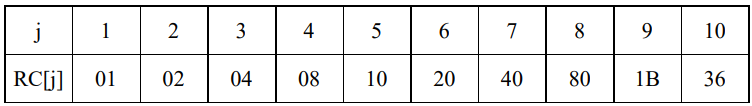
*Hình ?: Ví dụ minh họa phép cộng khóa*

**2.3.5. Mở rộng khóa**

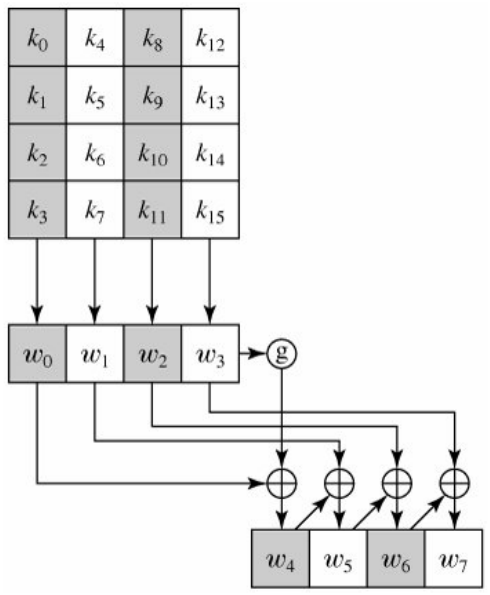
Thuật toán mở rộng khóa có đầu vào là 4 từ (16 byte) khóa và tạo ra một mảng đầu ra 44 từ (176 byte). Mã giả của thuật toán được mô tả như sau:



Trong đó, phép toán RotWord là thực hiện phép dịch trái 1 byte, tức là đầu vào l từ có 4 byte [b0, b1, b2, b3] thì kết quả sau khi thực hiện phép dịch trái 1 byte sẽ là [b1, b2, b3, b0]. Phép toán SubWord là phép thay thế byte sử dụng bảng S. Hằng số cho mỗi vòng khóa Rcon[j] = (RC[j], 0, 0, 0), với RC[1] = 1, RC[j] = 2.RC[j-1] và phép nhân (.) được thực hiện theo quy luật như trong thuật toán trộn cột. Giá trị của RC[j] được xác định như hình dưới ở hệ thập lục phân (Hexadecimal).



*Hình ?: Giá trị của RC[j]*



*Hình ?: Minh họa cách xác định khóa của vòng 1*