AddRoundKey

Trong thuật toán AES (Advanced Encryption Standard), AddRoundKey là một trong các bước chính trong quá trình mã hóa. Trong bước này, mỗi byte của ma trận dữ liệu đầu vào được kết hợp với một byte tương ứng từ khóa con (subkey). Khóa con được tạo ra từ khóa chính dựa trên vòng mã hóa hiện tại.

Chức năng:

+ Bước khởi tạo: XOR khóa mã với ma trận dữ liệu

+Bước lặp mã hóa và bước tạo ngõ ra: XOR khóa vòng (round key) với ma trận trạng thái.

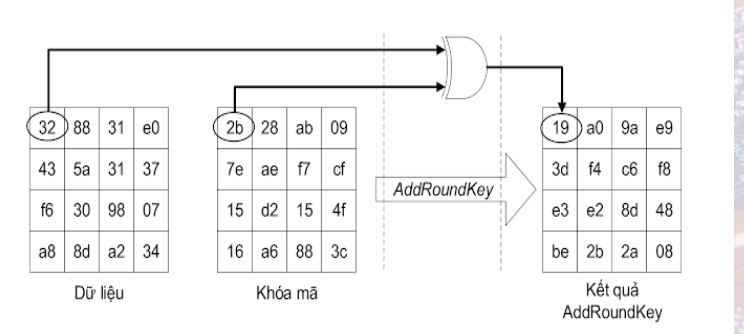
Đối với bước lặp mã hóa và bước tạo ngõ ra, vị trí "khóa mã" là các "khóa vòng" còn dữ liệu là của lần tính trước đó.

Quá trình thực hiện:

+Mỗi byte của ma trận dữ liệu đầu vào được thực hiện một phép XOR với byte tương ứng trong khóa con.

+Phép XOR này là một phép toán tạo ra sự kết hợp (hoặc "thêm") giữa dữ liệu và khóa con, từ đó làm cho dữ liệu trở nên không dễ dàng dự đoán và làm tăng độ an toàn của quá trình mã hóa.

Bước này là một trong những bước quan trọng nhất trong thuật toán AES, vì nó đóng góp vào việc tạo ra tính ngẫu nhiên và phức tạp trong quá trình mã hóa, làm cho việc phá mã trở nên khó khăn hơn.



subBytes

Trong thuật toán AES (Advanced Encryption Standard), SubBytes là một phần của quá trình mã hóa. Trong bước này, các byte trong ma trận dữ liệu đầu vào được thay thế bằng các byte khác sử dụng một hộp thay thế cố định gọi là hộp S-box.

Hộp S-box chứa một bảng 16x16 byte, trong đó mỗi byte được biểu diễn bởi hai hệ số hexa. Quá trình SubBytes thay thế mỗi byte trong ma trận dữ liệu đầu vào bằng byte tương ứng trong hộp S-box.

Quá trình này giúp tăng tính phân tán và phi tuyến trong dữ liệu, làm tăng độ phức tạp của việc phá mã. Nó cũng giúp đảm bảo rằng mỗi byte trong ma trận dữ liệu đầu vào không giữ nguyên giá trị của nó sau mỗi vòng mã hóa, làm cho mật mã trở nên an toàn hơn.

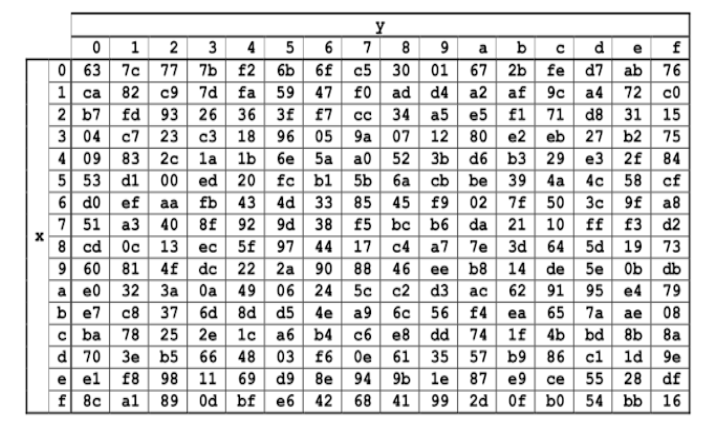
Chức năng:

ó các mục tiêu chính sau:

Tính Phi Tuyến: Bằng cách thay thế mỗi byte trong ma trận dữ liệu đầu vào bằng một byte tương ứng trong hộp S-box, SubBytes tạo ra sự phi tuyến trong dữ liệu, làm cho mỗi byte sau mỗi vòng mã hóa trở nên khác biệt và khó khăn hơn cho bất kỳ kẻ tấn công nào để dự đoán.

Phân Tán Dữ Liệu: Quá trình thay thế của SubBytes làm cho mỗi byte trong dữ liệu đầu vào được "phân tán" thành một byte khác, làm tăng sự phức tạp và đa dạng trong dữ liệu sau mỗi vòng mã hóa.

Tăng Độ An Toàn: Bằng cách tạo ra tính phi tuyến và phân tán, SubBytes đóng góp vào việc tăng cường độ an toàn của quá trình mã hóa AES, làm cho nó trở nên khó khăn hơn cho các kẻ tấn công để phá vỡ mã hóa và thu thập thông tin từ dữ liệu đã được mã hóa.



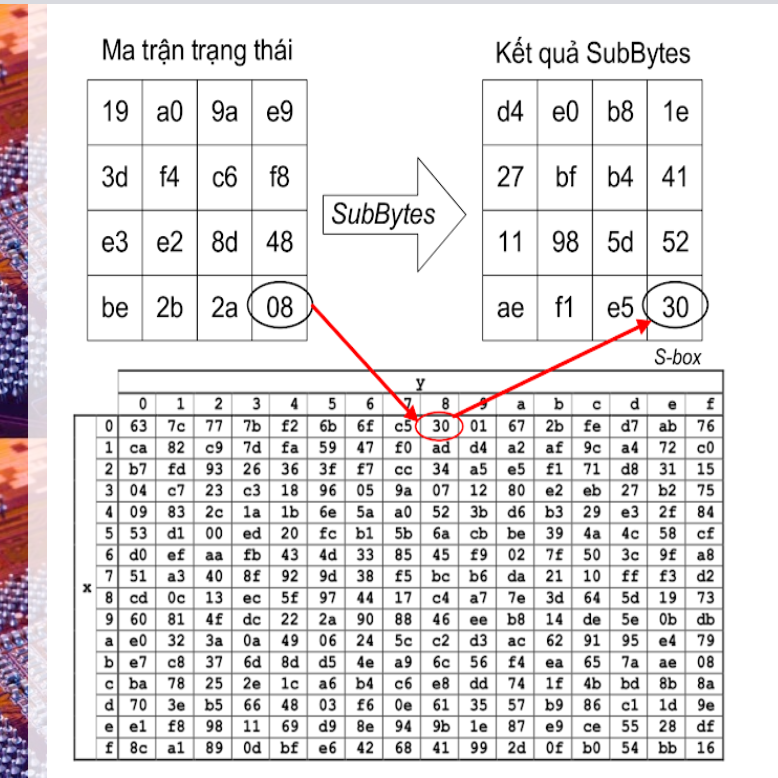
Quá trình thực hiện:

Nhận đầu vào: SubBytes nhận ma trận dữ liệu đầu vào từ bước trước đó trong quá trình mã hóa AES. Đầu vào này thường là một ma trận 4x4 byte.

Thay thế byte: Mỗi byte trong ma trận đầu vào được thay thế bằng một byte tương ứng trong hộp S-box. Cụ thể, byte tại hàng i, cột j của ma trận đầu vào được thay thế bằng byte tại hàng i, cột j trong hộp S-box.

Xuất ra: Sau khi tất cả các byte đã được thay thế, ma trận kết quả mới được tạo thành. Đây là đầu vào cho bước tiếp theo của quá trình mã hóa AES.

Quá trình này được lặp lại trong mỗi vòng của quá trình mã hóa AES, trừ vòng cuối cùng. SubBytes đóng vai trò quan trọng trong việc tạo ra tính phi tuyến và phân tán trong dữ liệu, làm tăng tính an toàn của thuật toán mã hóa.



Phép toán trường hữu hạn:

Phép toán trên trường hữu hạn 𝐺𝐹(28)GF(28) (hoặc GF(256)GF(256)) là một phần quan trọng của lĩnh vực mã hóa và mã hóa tin nhắn. Trong trường này, các phần tử là các số từ 0 đến 255, được biểu diễn bằng các số nhị phân 8-bit. Phép toán cơ bản trong 𝐺𝐹(28)GF(28) bao gồm cộng và nhân.

Phép Cộng (XOR): Trong trường này, phép cộng giống như phép XOR giữa các bit tương ứng của hai số. Ví dụ:

10110110⊕01101011=1101110110110110⊕01101011=11011101

Phép Nhân: Phép nhân trong 𝐺𝐹(28)GF(28) được thực hiện theo một cách đặc biệt gọi là nhân trong trường hữu hạn. Đây không phải là phép nhân thông thường, mà dựa trên một phép nhân modulo và một số luật cụ thể. Ví dụ, một số có thể được nhân với chính nó để tạo ra một giá trị khác. Cụ thể, phép nhân này thường được sử dụng trong các phép tính mã hóa và mã hóa tin nhắn.

Giả sử chúng ta muốn nhân hai số trong 𝐺𝐹(28)GF(28), ví dụ: 1011011010110110 và 0110001101100011.

Đầu tiên, chúng ta chia mỗi số thành các bit và ghi chúng dưới dạng dãy bit:

1011011010110110 (số 1) và 0110001101100011 (số 2).

Tiếp theo, chúng ta thực hiện phép nhân bit theo bit và sau đó thêm các kết quả lại với nhau, nhớ rằng phép nhân trong 𝐺𝐹(28)GF(28) thực hiện modulo với một đa thức nhất định.

10110110×01100011​10110110×01100011​

Đây là một phép nhân bit-by-bit:

  10110110×01100011‾  10110110+      00000000+         00000000+             10110110+++​  10110110×01100011​  10110110      00000000         00000000             10110110​

Sau đó, chúng ta thực hiện một số phép toán để đưa kết quả về dạng chuẩn trong 𝐺𝐹(28)GF(28). Trong trường hợp này, không có phép chia tỏa.

Vậy kết quả cuối cùng của phép nhân 10110110×0110001110110110×01100011 trong 𝐺𝐹(28)GF(28) là 0011001000110010.