# MỞ ĐẦU

# DANH MỤC HÌNH VẼ

Table of Contents

[**MỞ ĐẦU 1**](#_Toc61578942)

[**DANH MỤC HÌNH VẼ 2**](#_Toc61578943)

[**Chương 1. Giới thiệu VỀ MÃ HÓA RSA 4**](#_Toc61578944)

[1.1. Lịch sử ra đời RSA 4](#_Toc61578945)

[1.2. Hoạt động của thuật toán RSA 4](#_Toc61578946)

[1.2.1. Mô tả sơ lược 4](#_Toc61578947)

[1.2.2. Tạo khóa 5](#_Toc61578948)

[1.2.3. Mã hóa 6](#_Toc61578949)

[1.2.4. Giải mã 7](#_Toc61578950)

[1.2.5. Ví dụ 7](#_Toc61578951)

[1.3. Tạo chữ ký số cho văn bản 8](#_Toc61578952)

[1.4. Vấn đề an toàn của RSA 8](#_Toc61578953)

[1.5. Các vấn đề đặt ra trong thực tế 10](#_Toc61578954)

[1.5.1. Quá trình tạo khóa 10](#_Toc61578955)

[1.5.2. Phân phối khóa 11](#_Toc61578956)

[1.5.3. Tấn công dựa trên thời gian 11](#_Toc61578957)

[1.6. Sự cần thiết của việc chuyển đổi bản rõ 12](#_Toc61578958)

[Chương 2. Giới thiệu về MÃ HÓA AES 14](#_Toc61578959)

[*2.3.1* *Quá trình mã hóa* 16](#_Toc61578960)

[*2.3.2* *Bước SubBytes* 17](#_Toc61578961)

[*2.3.3* *Bước ShiftRows* 18](#_Toc61578962)

[*2.3.4* *Bước MixColumns* 18](#_Toc61578963)

[*2.3.5* *Bước AddRoundKey* 19](#_Toc61578964)

[*2.5.1* *ECB (Electronic Code Book)* 21](#_Toc61578965)

[*2.5.2* *CBC (Cipher Block Chaining)* 22](#_Toc61578966)

[*2.5.3* *CFB (Cipher Feedback)* 23](#_Toc61578967)

[*2.5.4* *OFB (Output Feedback)* 24](#_Toc61578968)

[*2.5.5* *XTS* 25](#_Toc61578969)

[Chương 3. Bảo mật hệ thống camera an ninh 26](#_Toc61578970)

[*3.1.1* *Hoạt động của hệ thống* 26](#_Toc61578971)

[*3.1.2* *Sơ đồ khối chức năng của hệ thống* 26](#_Toc61578972)

[*3.2.1* *Bảo mật truyền thông giữa Raspberry và Server* 27](#_Toc61578973)

[3.2.2 Bảo mật lưu trữ video 29](#_Toc61578974)

[KẾT LUẬN 30](#_Toc61578975)

# Giới thiệu VỀ MÃ HÓA RSA

## Lịch sử ra đời RSA

Thuật toán được Ron Rivest, Adi Shamir và Len Adleman mô tả lần đầu tiên vào năm 1977 tại Học viện Công nghệ Massachusetts (MIT). Tên của thuật toán lấy từ 3 chữ cái đầu của tên 3 tác giả.

Trước đó, vào năm 1973, Clifford Cocks, một nhà toán học người Anh làm việc tại GCHQ, đã mô tả một thuật toán tương tự. Với khả năng tính toán tại thời điểm đó thì thuật toán này không khả thi và chưa bao giờ được thực nghiệm. Tuy nhiên, phát minh này chỉ được công bố vào năm 1997 vì được xếp vào loại tuyệt mật.

Thuật toán RSA được MIT đăng ký bằng sáng chế tại Hoa Kỳ vào năm 1983 (Số đăng ký 4,405,829). Bằng sáng chế này hết hạn vào ngày 21 tháng 9 năm 2000. Tuy nhiên, do thuật toán đã được công bố trước khi có đăng ký bảo hộ nên sự bảo hộ hầu như không có giá trị bên ngoài Hoa Kỳ. Ngoài ra, nếu như công trình của Clifford Cocks đã được công bố trước đó thì bằng sáng chế RSA đã không thể được đăng ký.

## Hoạt động của thuật toán RSA

### **Mô tả sơ lược**

Thuật toán RSA có hai khóa: khóa công khai (hay khóa công cộng) và khóa bí mật (hay khóa cá nhân). Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.

Ta có thể mô phỏng trực quan một hệ mật mã khoá công khai như sau : Bob muốn gửi cho Alice một thông tin mật mà Bob muốn duy nhất Alice có thể đọc được. Để làm được điều này, Alice gửi cho Bob một chiếc hộp có khóa đã mở sẵn và giữ lại chìa khóa. Bob nhận chiếc hộp, cho vào đó một tờ giấy viết thư bình thường và khóa lại (như loại khoá thông thường chỉ cần sập chốt lại, sau khi sập chốt khóa ngay cả Bob cũng không thể mở lại được-không đọc lại hay sửa thông tin trong thư được nữa). Sau đó Bob gửi chiếc hộp lại cho Alice. Alice mở hộp với chìa khóa của mình và đọc thông tin trong thư. Trong ví dụ này, chiếc hộp với khóa mở đóng vai trò khóa công khai, chiếc chìa khóa chính là khóa bí mật.

### **Tạo khóa**

Giả sử Alice và Bob cần trao đổi thông tin bí mật thông qua một kênh không an toàn (ví dụ như Internet). Với thuật toán RSA, Alice đầu tiên cần tạo ra cho mình cặp khóa gồm khóa công khai và khóa bí mật theo các bước sau:

1. Chọn 2 số nguyên tố lớn và với, lựa chọn ngẫu nhiên và độc lập.
2. Tính:
3. Tính giá trị hàm số Ơle của :
4. Chọn một số tự nhiên  sao cho  và là số nguyên tố cùng nhau với .
5. Tính:  sao cho .

Một số lưu ý:

* + - Các số nguyên tố thường được chọn bằng phương pháp thử xác suất.
    - Các bước 4 và 5 có thể được thực hiện bằng giải thuật Euclid mở rộng.
    - Đối với bước 3, PKCS #1 v2.1 sử dụng  thay cho , trong đó LCM (Least Common Multiple) là hàm tìm bội số chung nhỏ nhất.

Khi đó, khóa công khai là bộ , với  được gọi là số mũ công khai, hay số mũ mã hóa; còn khóa bí mật là bộ , với  được gọi là số mũ bí mật, hay số mũ giải mã. Một dạng khác của khóa bí mật là bộ bao gồm:

* + -  và  là hai số nguyên tố chọn ban đầu,
    -  và  (thường được gọi là  và ),
    -  (thường được gọi là )

Dạng khóa bí mật này cho phép thực hiện giải mã và ký nhanh hơn với việc sử dụng định lý số dư Trung Quốc. Ở dạng này, tất cả thành phần của khóa bí mật phải được giữ bí mật. Alice gửi khóa công khai cho Bob, và giữ bí mật khóa cá nhân của mình. Ở đây,  và  giữ vai trò rất quan trọng. Chúng là các phân tố của  và cho phép tính  khi biết . Nếu không sử dụng dạng sau của khóa bí mật (dạng CRT) thì  và  sẽ được xóa ngay sau khi thực hiện xong quá trình tạo khóa.

### **Mã hóa**

Giả sử Bob muốn gửi thông điệp M cho Alice. Nói chung, thông điệp M là một số nguyên  thỏa mãn . Lúc này Bob có  và biết  cũng như  do Alice gửi. Bob sẽ tính  là bản mã hóa của  theo công thức:

|  |  |
| --- | --- |
|  | (1.1) |

Hàm trên có thể tính dễ dàng sử dụng phương pháp tính hàm mũ (theo môđun) bằng thuật toán bình phương và nhân. Cuối cùng Bob gửi  cho Alice.

### **Giải mã**

Alice nhận  từ Bob và biết khóa bí mật . Alice có thể tìm được  từ  theo công thức sau:

|  |  |
| --- | --- |
|  | (1.2) |

Tính đúng đắn của công thức giải mã (1.2) được giải thích như sau:

|  |  |
| --- | --- |
|  | (1.3) |

* + - Do  và  theo định lý Ferma nhỏ, nên:

|  |  |
| --- | --- |
| và | (1.4) |

* + - Do  và  là hai số nguyên tố cùng nhau nên áp dụng định lý số dư Trung Quốc ta được:

|  |  |
| --- | --- |
|  | (1.5) |

Tức là

|  |  |
| --- | --- |
|  | (1.6) |

### **Ví dụ**

Để làm rõ hơn hoạt động của hệ mật RSA, sau đây ta lấy một ví dụ với những số cụ thể. Tất nhiên, ở đây ta sử dụng những số nhỏ để tiện tính toán còn trong thực tế phải dùng các số có giá trị đủ lớn.

Lấy:

* + -  – số nguyên tố thứ nhất (giữ bí mật hoặc hủy sau khi tạo khóa)
    -  – số nguyên tố thứ hai (giữ bí mật hoặc hủy sau khi tạo khóa)
    -  – mô-đun (công bố công khai)
    -  – số mũ công khai
    -  – số mũ bí mật
    -  – thông điệp cần mã hóa

Khi đó, ta có khóa công khai là cặp , còn khóa bí mật là cặp . Việc mã hóa được thực hiện như sau:



Và công thức để giải mã là:



Một lần nữa nhắc lại rằng, các phép tính trên đều có thể được thực hiện hiệu quả nhờ giải thuật bình phương và nhân.

## Tạo chữ ký số cho văn bản

Thuật toán RSA còn được dùng để tạo chữ ký số cho văn bản. Giả sử Alice muốn gửi cho Bob một văn bản có chữ ký của mình. Để làm việc này, Alice tạo ra một giá trị băm (hash value) của văn bản cần ký và tính giá trị mũ  (theo mô-đun ) của nó (giống như khi Alice thực hiện giải mã). Giá trị cuối cùng chính là chữ ký điện tử của văn bản đang xét. Khi Bob nhận được văn bản cùng với chữ ký điện tử, anh ta tính giá trị mũ  (theo mô-đun ) của chữ ký đồng thời với việc tính giá trị băm của văn bản. Nếu hai giá trị này như nhau thì Bob biết rằng người tạo ra chữ ký biết khóa bí mật của Alice và văn bản đã không bị thay đổi sau khi ký.

## Vấn đề an toàn của RSA

Độ an toàn của hệ thống RSA dựa trên hai vấn đề của toán học: bài toán phân tích ra thừa số nguyên tố các số nguyên lớn và bài toán RSA. Nếu hai bài toán trên là khó (không tìm được thuật toán hiệu quả để giải chúng) thì không thể thực hiện được việc phá mã toàn bộ đối với RSA.

Bài toán RSA là bài toán tính căn bậc  theo mô-đun  (với  là hợp số), tức là tìm số  sao cho , trong đó  chính là khóa công khai và  là bản mã. Hiện nay phương pháp triển vọng nhất giải bài toán này là phân tích  ra thừa số nguyên tố. Khi thực hiện được điều này, kẻ tấn công sẽ tìm ra số mũ bí mật  từ khóa công khai và có thể giải mã theo đúng quy trình của thuật toán. Nếu kẻ tấn công tìm được 2 số nguyên tố  và  sao cho:  thì có thể dễ dàng tìm được giá trị  và qua đó xác định  từ . Chưa có một phương pháp nào được tìm ra trên máy tính để giải bài toán này trong thời gian đa thức (polynomial-time). Tuy nhiên người ta cũng chưa chứng minh được điều ngược lại (sự không tồn tại của thuật toán).

Tại thời điểm năm 2005, số lớn nhất có thể được phân tích ra thừa số nguyên tố có độ dài 663 bít với phương pháp phân tán trong khi khóa của RSA hiện nay thường có độ dài từ 1024 tới 2048 bít. Một số chuyên gia cho rằng khóa 1024 bít có thể sớm bị phá vỡ (cũng có nhiều người phản đối việc này). Với khóa 4096 bít thì hầu như không có khả năng bị phá vỡ trong tương lai gần. Do đó, người ta thường cho rằng RSA đảm bảo an toàn với điều kiện  được chọn đủ lớn. Nếu  có độ dài 256 bít hoặc ngắn hơn, nó có thể bị phân tích trong vài giờ với máy tính cá nhân dùng các phần mềm có sẵn. Nếu  có độ dài 512 bít, nó có thể bị phân tích bởi vài trăm máy tính tại thời điểm năm 1999. Một thiết bị lý thuyết có tên là TWIRL do Shamir và Tromer mô tả năm 2003 đã đặt ra câu hỏi về độ an toàn của khóa 1024 bít. Vì vậy hiện nay người ta khuyến cáo sử dụng khóa có độ dài tối thiểu 2048 bít.

Năm 1993, Peter Shor công bố thuật toán Shor chỉ ra rằng: máy tính lượng tử (trên lý thuyết) có thể giải bài toán phân tích ra thừa số trong thời gian đa thức. Tuy nhiên, máy tính lượng tử vẫn chưa thể phát triển được tới mức độ này trong nhiều năm nữa.

## Các vấn đề đặt ra trong thực tế

### **Quá trình tạo khóa**

Việc tìm ra 2 số nguyên tố đủ lớn  và  thường được thực hiện bằng cách thử xác suất các số ngẫu nhiên có độ lớn phù hợp (dùng phép kiểm tra nguyên tố cho phép loại bỏ hầu hết các hợp số).

Hai số  và  còn cần được chọn không quá gần nhau để phòng trường hợp phân tích n bằng phương pháp phân tích Fermat. Ngoài ra, nếu  và  có thừa số nguyên tố nhỏ thì  cũng có thể dễ dàng bị phân tích và vì thế  và  cũng cần được thử để tránh khả năng này.

Bên cạnh đó, cần tránh sử dụng các phương pháp tìm số ngẫu nhiên mà kẻ tấn công có thể lợi dụng để biết thêm thông tin về việc lựa chọn (cần dùng các bộ tạo số ngẫu nhiên tốt). Yêu cầu ở đây là các số được lựa chọn cần đồng thời ngẫu nhiên và không dự đoán được. Đây là các yêu cầu khác nhau: một số có thể được lựa chọn ngẫu nhiên (không có kiểu mẫu trong kết quả) nhưng nếu có thể dự đoán được dù chỉ một phần thì an ninh của thuật toán cũng không được đảm bảo. Một ví dụ là bảng các số ngẫu nhiên do tập đoàn Rand xuất bản vào những năm 1950 có thể rất thực sự ngẫu nhiên nhưng kẻ tấn công cũng có bảng này. Nếu kẻ tấn công đoán được một nửa chữ số của  hay  thì chúng có thể dễ dàng tìm ra nửa còn lại (theo nghiên cứu của Donald Coppersmith vào năm 1997)

Một điểm nữa cần nhấn mạnh là khóa bí mật  phải đủ lớn. Năm 1990, Wiener chỉ ra rằng nếu giá trị của  nằm trong khoảng  và  (khá phổ biến) và  thì có thể tìm ra được  từ  và .

Mặc dù  đã từng có giá trị là 3 nhưng hiện nay các số mũ nhỏ không còn được sử dụng do có thể tạo nên những lỗ hổng. Giá trị thường dùng hiện nay là 65537 vì được xem là đủ lớn và cũng không quá lớn ảnh hưởng tới việc thực hiện hàm mũ.

Tốc độ RSA có tốc độ thực hiện chậm hơn đáng kể so với DES và các thuật toán mã hóa đối xứng khác. Trên thực tế, Bob sử dụng một thuật toán mã hóa đối xứng nào đó để mã hóa văn bản cần gửi và chỉ sử dụng RSA để mã hóa khóa để giải mã (thông thường khóa ngắn hơn nhiều so với văn bản).

Phương thức này cũng tạo ra những vấn đề an ninh mới. Một ví dụ là cần phải tạo ra khóa đối xứng thật sự ngẫu nhiên. Nếu không, kẻ tấn công sẽ bỏ qua RSA và tập trung vào việc đoán khóa đối xứng.

### **Phân phối khóa**

Cũng giống như các thuật toán mã hóa khác, cách thức phân phối khóa công khai là một trong những yếu tố quyết định đối với độ an toàn của RSA. Quá trình phân phối khóa cần chống lại được tấn công đứng giữa (man-in-the-middle attack). Giả sử Eve có thể gửi cho Bob một khóa bất kỳ và khiến Bob tin rằng đó là khóa (công khai) của Alice. Đồng thời Eve có khả năng đọc được thông tin trao đổi giữa Bob và Alice. Khi đó, Eve sẽ gửi cho Bob khóa công khai của chính mình (mà Bob nghĩ rằng đó là khóa của Alice). Sau đó, Eve đọc tất cả văn bản mã hóa do Bob gửi, giải mã với khóa bí mật của mình, giữ một bản sao đồng thời mã hóa bằng khóa công khai của Alice và gửi cho Alice. Về nguyên tắc, cả Bob và Alice đều không phát hiện ra sự can thiệp của người thứ ba. Các phương pháp chống lại dạng tấn công này thường dựa trên các chứng thực khóa công khai (digital certificate) hoặc các thành phần của hạ tầng khóa công khai (public key infrastructure - PKI).

### **Tấn công dựa trên thời gian**

Vào năm 1995, Paul Kocher mô tả một dạng tấn công mới lên RSA: nếu kẻ tấn công nắm đủ thông tin về phần cứng thực hiện mã hóa và xác định được thời gian giải mã đối với một số bản mã lựa chọn thì có thể nhanh chóng tìm ra khóa . Dạng tấn công này có thể áp dụng đối với hệ thống chữ ký điện tử sử dụng RSA. Năm 2003, Dan Boneh và David Brumley chứng minh một dạng tấn công thực tế hơn: phân tích thừa số RSA dùng mạng máy tính (Máy chủ web dùng SSL). Tấn công đã khai thác thông tin rò rỉ của việc tối ưu hóa định lý số dư Trung quốc mà nhiều ứng dụng đã thực hiện.

Để chống lại tấn công dựa trên thời gian là đảm bảo quá trình giải mã luôn diễn ra trong thời gian không đổi bất kể văn bản mã. Tuy nhiên, cách này có thể làm giảm hiệu suất tính toán. Thay vào đó, hầu hết các ứng dụng RSA sử dụng một kỹ thuật gọi là che mắt. Kỹ thuật này dựa trên tính nhân của RSA: thay vì tính , Alice đầu tiên chọn một số ngẫu nhiên  và tính . Kết quả của phép tính này là  và tác động của  sẽ được loại bỏ bằng cách nhân kết quả với nghịch đảo của . Đối với mỗi văn bản mã, người ta chọn một giá trị của . Vì vậy, thời gian giải mã sẽ không còn phụ thuộc vào giá trị của văn bản mã.

## Sự cần thiết của việc chuyển đổi bản rõ

Nếu áp dụng trực tiếp lược đồ RSA vào thực tế thì RSA sẽ gặp phải một số vấn đề sau:

Nếu m = 0 hoặc m = 1 sẽ tạo ra các bản mã có giá trị là 0 và 1 tương ứng.

Khi mã hóa với số mũ nhỏ (chẳng hạn e = 3) và m cũng có giá trị nhỏ, giá trị  cũng nhận giá trị nhỏ (so với n). Như vậy phép môđun không có tác dụng và có thể dễ dàng tìm được m bằng cách khai căn bậc e của c (bỏ qua môđun).

RSA là phương pháp mã hóa xác định (không có thành phần ngẫu nhiên) nên kẻ tấn công có thể thực hiện tấn công lựa chọn bản rõ bằng cách tạo ra một bảng tra giữa bản rõ và bản mã. Khi gặp một bản mã, kẻ tấn công sử dụng bảng tra để tìm ra bản rõ tương ứng.

Trên thực tế, ta thường gặp 2 vấn đề đầu khi gửi các bản tin ASCII ngắn với m là nhóm vài ký tự ASCII. Một đoạn tin chỉ có 1 ký tự NULL sẽ được gán giá trị m = 0 và cho ra bản mã là 0 bất kể giá trị của e và N. Tương tự, một ký tự ASCII khác, SOH, có giá trị 1 sẽ luôn cho ra bản mã là 1. Với các hệ thống dùng giá trị e nhỏ thì tất cả ký tự ASCII đều cho kết quả mã hóa không an toàn vì giá trị lớn nhất của m chỉ là 255 và  nhỏ hơn giá trị  chấp nhận được. Những bản mã này sẽ dễ dàng bị phá mã.

Để tránh gặp phải những vấn đề trên, RSA trên thực tế thường bao gồm một hình thức chuyển đổi ngẫu nhiên hóa  trước khi mã hóa. Quá trình chuyển đổi này phải đảm bảo rằng  không rơi vào các giá trị không an toàn. Sau khi chuyển đổi, mỗi bản rõ khi mã hóa sẽ cho ra một trong số khả năng trong tập hợp bản mã. Điều này làm giảm tính khả thi của phương pháp tấn công lựa chọn bản rõ (một bản rõ sẽ có thể tương ứng với nhiều bản mã tuỳ thuộc vào cách chuyển đổi).

Một số tiêu chuẩn, chẳng hạn như PKCS, đã được thiết kế để chuyển đổi bản rõ trước khi mã hóa bằng RSA. Các phương pháp chuyển đổi này bổ sung thêm bít vào M. Các phương pháp chuyển đổi cần được thiết kế cẩn thận để tránh những dạng tấn công phức tạp tận dụng khả năng biết trước được cấu trúc của bản rõ. Phiên bản ban đầu của PKCS dùng một phương pháp đặc ứng (ad-hoc) mà về sau được biết là không an toàn trước tấn công lựa chọn bản rõ thích ứng (adaptive chosen ciphertext attack). Các phương pháp chuyển đổi hiện đại sử dụng các kỹ thuật như chuyển đổi mã hóa bất đối xứng tối ưu (Optimal Asymmetric Encryption Padding - OAEP) để chống lại tấn công dạng này. Tiêu chuẩn PKCS còn được bổ sung các tính năng khác để đảm bảo an toàn cho chữ ký RSA (Probabilistic Signature Scheme for RSA - RSA-PSS).

# Giới thiệu về MÃ HÓA AES

* 1. **Lịch sử ra đời AES**

AES là viết tắt của Advanced Encryption Standard, chuẩn mã hóa 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 hóa đối xứng dạng khối, nghĩa là mỗi khối văn bản có một kích thước nhất định (128 bit) được mã hóa, khác với mã hóa dạng chuỗi khi từng kí tự được mã hóa. Đối xứng nghĩa là khóa để mã hóa và giải mã đều là một.

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.

* 1. **Hoạt động của AES**

AES là kiểu mã hóa khối, mỗi khối kích thước 128 bit. Khóa đối xứng với 3 kích thước là 128, 192 và 256 bit, trong đó 2 kích thước sau được chính phủ Mỹ dùng cho các tài liệu mật cấp cao, được gọi là “Top Secret”.

Rijndael ban đầu được phép thêm khối và tăng độ dài khóa nhưng chuẩn sau này bị bỏ, giữ chuẩn kích thước như đã nói ở trên. AES là chuẩn mã hóa duy nhất được phát hành rộng rãi được NSA chấp thuận dùng để bảo vệ thông tin chính phủ ở mức cao cấp nhất.

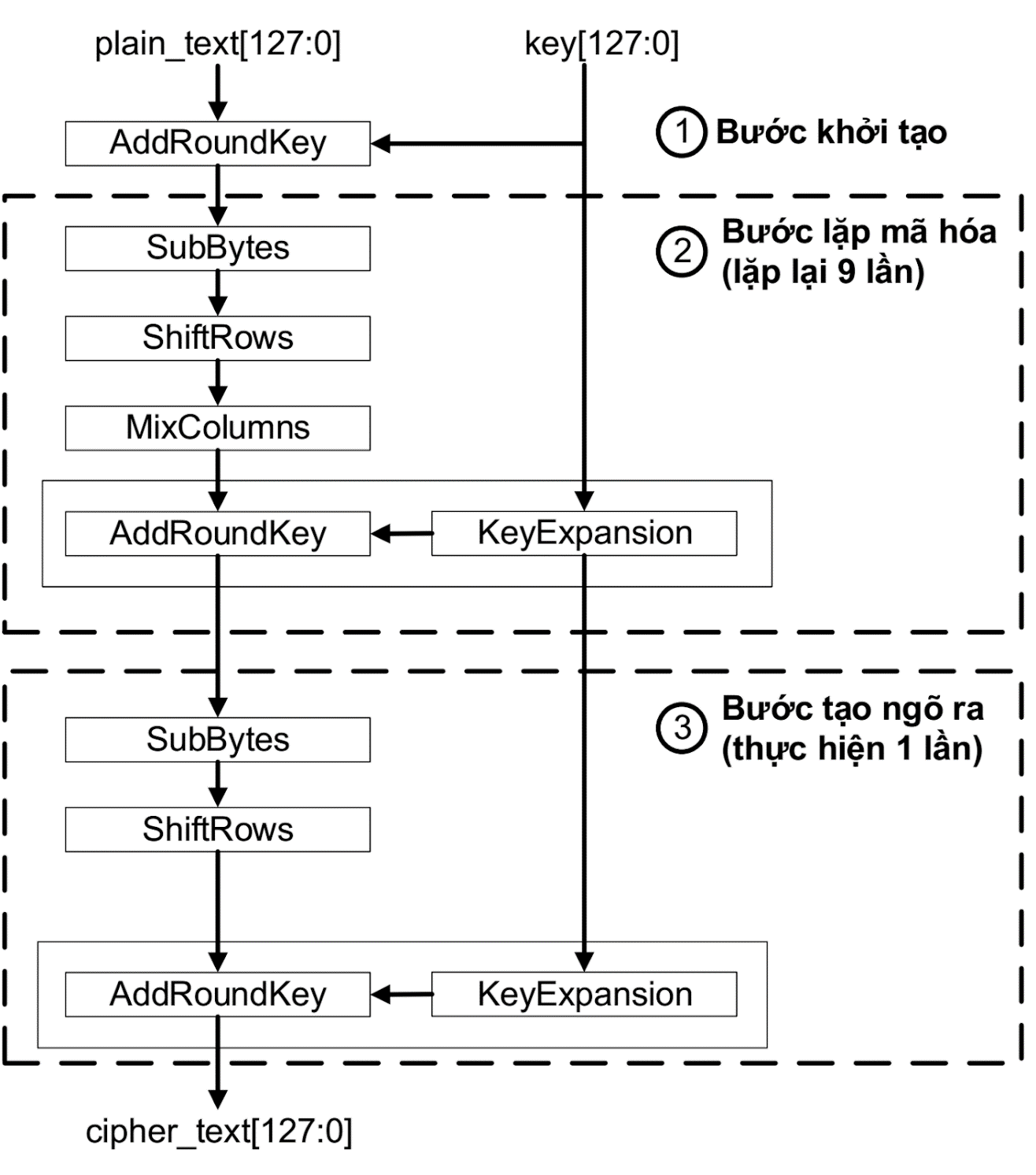
AES dùng thuật toán mã hóa khối mạng thay thế hoán đổi (SPN - Substitution Permutation Network). Dữ liệu được chuyển thành dạng an toàn trong vài bước, bắt đầu là khối plain text kích thước chuẩn, sau đó chèn vào hàng và sau đó là mã hóa. Mỗi lần đều có các bước thay thế, chuyển đổi, hòa trộn.

Cũng như 3DES có 3 bước mã hóa, AES cũng có nhiều bước nhưng được thực hiện nhiều hơn, phụ thuộc vào độ dài khóa, với khóa 128 bit là 10 lần, khóa 192 bit là 12 lần và khóa 256 bit là 14 lần.

Trong quá trình này, khóa mã hóa được tạo và cũng phải có khóa này để giải mã. Nếu không, dữ liệu sẽ chỉ là mớ lộn xộn không thể đọc được. Cả người gửi và người nhận đều phải biết khóa mã hóa và giải mã.

* 1. **Thuật toán AES**

## *Quá trình mã hóa*



* Khởi động vòng lặp

1. AddRoundKey — Mỗi cột của trạng thái đầu tiên lần lượt được kết hợp với một khóa con theo thứ tự từ đầu dãy khóa.

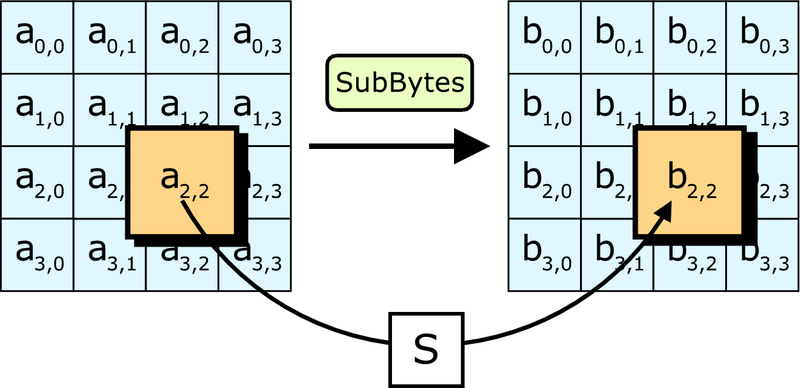
* Vòng lặp

1. SubBytes — đây là phép thế (phi tuyến) trong đó mỗi byte trong trạng thái sẽ được thế bằng một byte khác theo bảng tra (Rijndael S-box).
2. ShiftRows — dịch chuyển, các hàng trong trạng thái được dịch vòng theo số bước khác nhau.
3. MixColumns — quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính.
4. AddRoundKey

* Vòng lặp cuối

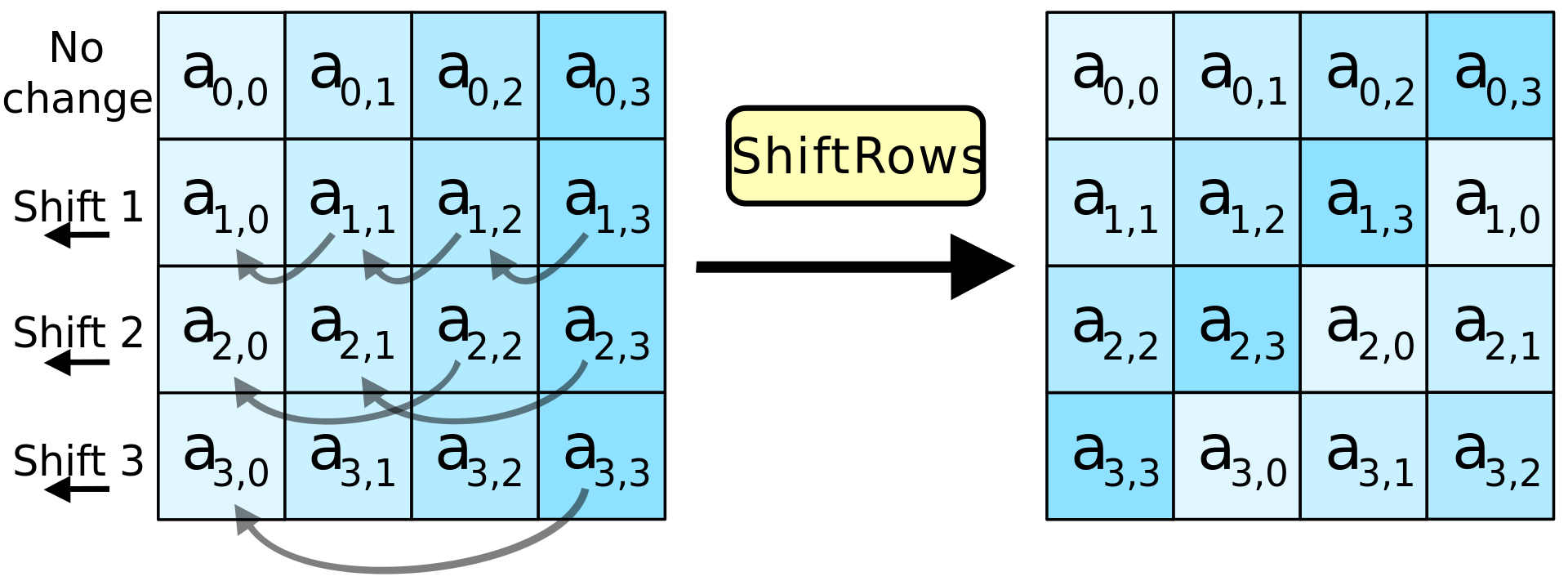
1. SubBytes
2. ShiftRows
3. AddRoundKey

## *Bước SubBytes*



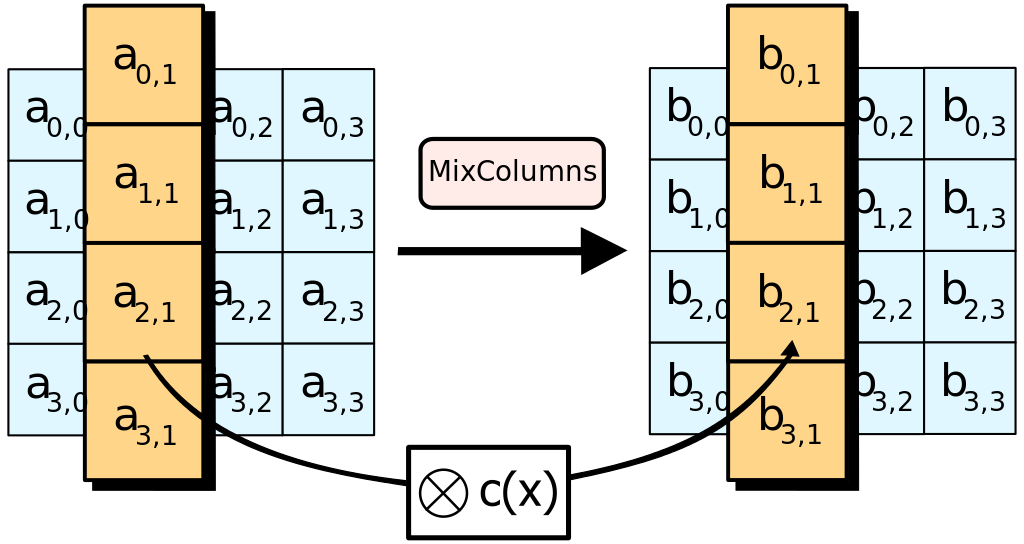
Các byte được thế thông qua bảng tra S-box. Đây chính là quá trình phi tuyến của thuật toán. Hộp S-box này được tạo ra từ một phép biến đổi khả nghịch trong trường hữu hạn GF () có tính chất phi tuyến. Để chống lại các tấn công dựa trên các đặc tính đại số, hộp S-box này được tạo nên bằng cách kết hợp phép nghịch đảo với một phép biến đổi affine khả nghịch. Hộp S-box này cũng được chọn để tránh các điểm bất động (fixed point).

## *Bước ShiftRows*



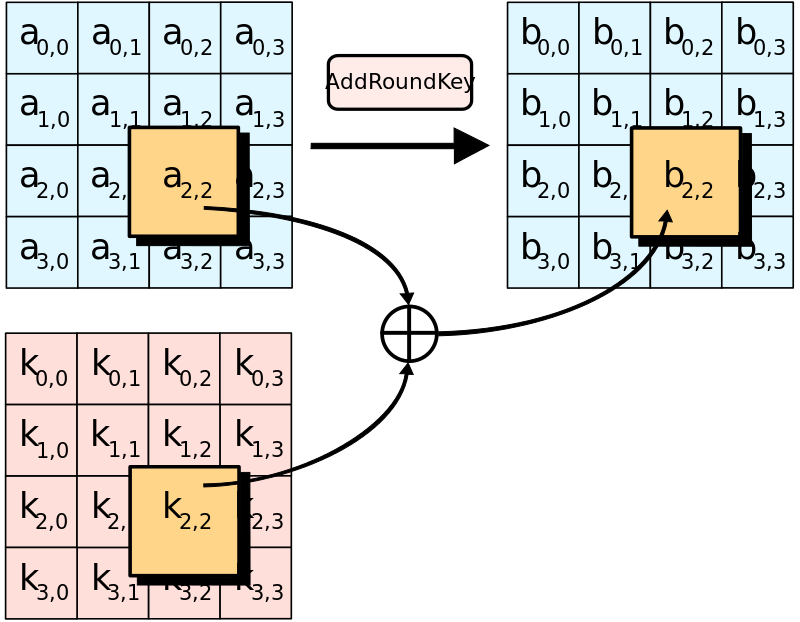
Các hàng được dịch vòng một số bước nhất định. Đối với AES, hàng đầu được giữ nguyên. Mỗi byte của hàng thứ 2 được dịch vòng trái một vị trí. Tương tự, các hàng thứ 3 và 4 được dịch vòng 2 và 3 vị trí. Do vậy, mỗi cột khối đầu ra của bước này sẽ bao gồm các byte ở đủ 4 cột khối đầu vào. Đối với Rijndael với độ dài khối khác nhau thì số vị trí dịch chuyển cũng khác nhau.

## *Bước MixColumns*



Bốn byte trong từng cột được kết hợp lại theo một phép biến đổi tuyến tính khả nghịch. Mỗi khối 4 byte đầu vào sẽ cho một khối 4 byte ở đầu ra với tính chất là mỗi byte ở đầu vào đều ảnh hưởng tới cả bốn byte đầu ra. Cùng với bước ShiftRows, MixColumns đã tạo ra tính chất khuếch tán cho thuật toán. Mỗi cột được xem như một đa thức trong trường hữu hạn và được nhân với đa thức . Vì thế, bước này có thể được xem là phép nhân ma trận trong trường hữu hạn.

## *Bước AddRoundKey*



Tại bước này, khóa con được kết hợp với các khối. Khóa con trong mỗi chu trình được tạo ra từ khóa chính với quá trình tạo khóa con Rijndael; mỗi khóa con có độ dài giống như các khối. Quá trình kết hợp được thực hiện bằng cách XOR từng bit của khóa con với khối dữ liệu.

* 1. **Vấn đề an toàn của AES**

Vào thời điểm năm 2006, dạng tấn công lên AES duy nhất thành công là tấn công kênh bên (side channel attack]). Vào tháng 6 năm 2003, chính phủ Hoa kỳ tuyên bố AES có thể được sử dụng cho thông tin mật.

"Thiết kế và độ dài khóa của thuật toán AES (128, 192 và 256 bít) là đủ an toàn để bảo vệ các thông tin được xếp vào loại tối mật (secret). Các thông tin tuyệt mật (top secret) sẽ phải dùng khóa 192 hoặc 256 bít. Các phiên bản thực hiện AES nhằm mục đích bảo vệ hệ thống an ninh hay thông tin quốc gia phải được NSA kiểm tra và chứng nhận trước khi sử dụng."

Điều này đánh dấu lần đầu tiên công chúng có quyền tiếp xúc với thuật toán mật mã mà NSA phê chuẩn cho thông tin tuyệt mật. Nhiều phần mềm thương mại hiện nay sử dụng mặc định khóa có độ dài 128 bít.

Phương pháp thường dùng nhất để tấn công các dạng mã hóa khối là thử các kiểu tấn công lên phiên bản có số chu trình thu gọn. Đối với khóa 128 bít, 192 bít và 256 bít, AES có tương ứng 10, 12 và 14 chu trình. Tại thời điểm năm 2006, những tấn công thành công được biết đến là 7 chu trình đối với khóa 128 bít, 8 chu trình với khóa 192 bít và 9 chu trình với khóa 256 bít.

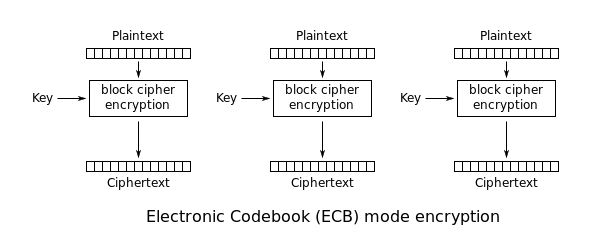
Một số nhà khoa học trong lĩnh vực mật mã lo ngại về an ninh của AES. Họ cho rằng ranh giới giữa số chu trình của thuật toán và số chu trình bị phá vỡ quá nhỏ. Nếu các kỹ thuật tấn công được cải thiện thì AES có thể bị phá vỡ. Ở đây, phá vỡ có nghĩa chỉ bất cứ phương pháp tấn công nào nhanh hơn tấn công kiểu duyệt toàn bộ. Vì thế một tấn công cần thực hiện 2120 cũng được coi là thành công mặc dù tấn công này chưa thể thực hiện trong thực tế. Tại thời điểm hiện nay, nguy cơ này không thực sự nguy hiểm và có thể bỏ qua. Tấn công kiểu duyệt toàn bộ quy mô nhất đã từng thực hiện là do distributed.net thực hiện lên hệ thống 64 bit RC5 vào năm 2002 (Theo định luật Moore thì nó tương đương với việc tấn công vào hệ thống 66 bit hiện nay).

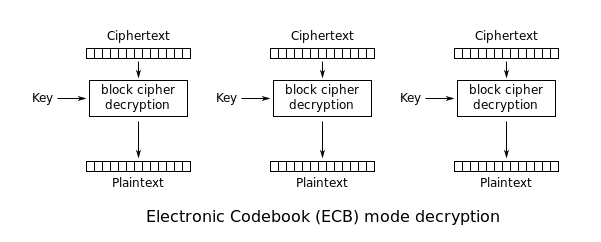
Một vấn đề khác nữa là cấu trúc toán học của AES. Không giống với các thuật toán mã hóa khác, AES có mô tả toán học khá đơn giản. Tuy điều này chưa dẫn đến mối nguy hiểm nào nhưng một số nhà nghiên cứu sợ rằng sẽ có người lợi dụng được cấu trúc này trong tương lai.

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.

* 1. **Các chế độ mã hóa AES**

## *ECB (Electronic Code Book)*

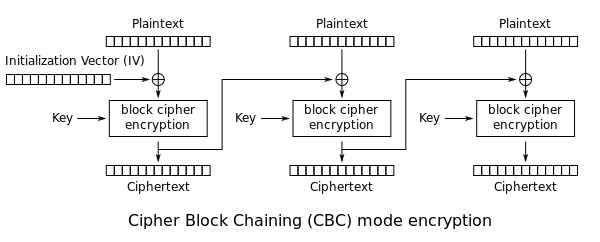


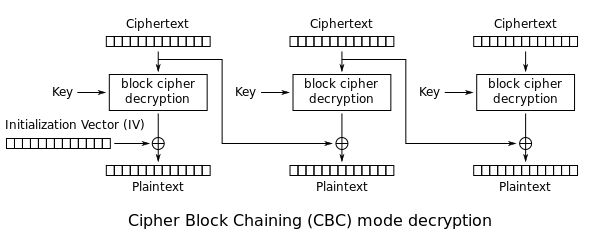


**ECB** mã hóa từng khối dữ liệu một cách độc lập và cùng một khối văn bản gốc sẽ dẫn đến cùng một khối bản mã.

Nhược điểm của ECB: lặp trên bản mã sẽ nhận biết được việc lặp trên bản rõ, đặc biệt nếu dóng đúng khối và thường xảy ra với hình ảnh hoặc với bản tin mà thay đổi rất ít sẽ trở thành đối tượng để thám mã.

## *CBC (Cipher Block Chaining)*



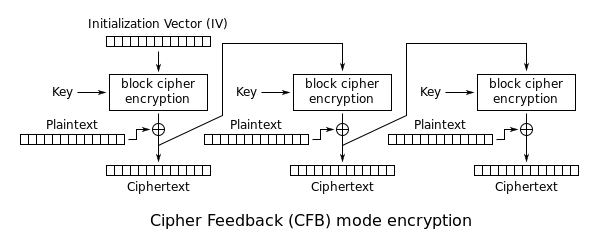


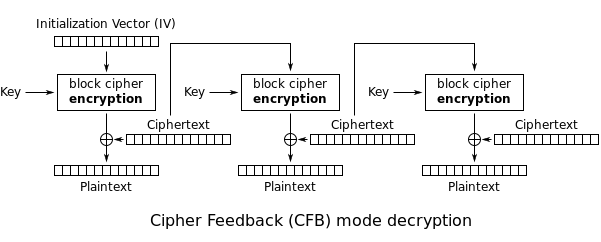
Trong chế độ CBC, mỗi khối bản rõ được XOR với khối bản mã trước đó trước khi được mã hóa. Bằng cách này, mỗi khối bản mã phụ thuộc vào tất cả các khối bản rõ được xử lý cho đến thời điểm đó. Để làm cho mỗi thông báo là duy nhất, một vectơ khởi tạo phải được sử dụng trong khối đầu tiên.

Việc giải mã với IV không chính xác khiến khối bản rõ đầu tiên bị hỏng nhưng các khối bản rõ tiếp theo sẽ đúng. Điều này là do mỗi khối được XOR với bản mã của khối trước đó, không phải bản rõ, vì vậy người ta không cần phải giải mã khối trước đó trước khi sử dụng nó làm IV để giải mã khối hiện tại. Điều này có nghĩa là một khối bản rõ có thể được khôi phục từ hai khối bản mã liền kề. Do đó, việc giải mã có thể được thực hiện song song.

Nhược điểm của CBC: Cần giá trị vectơ ban đầu IV được biết trước bởi người gửi và người nhận. Tuy nhiên nếu IV được gửi công khai, kẻ tấn công có thể thay đổi bit đầu tiên và thay đổi cả IV để bù trừ. Vậy IV cần phải có giá trị cố định trước hoặc mã hoá trong chế độ ECB và gửi trước phần còn lại của mẩu tin.Thêm nữa là chế độ mã CBC thực hiện chậm, vì khối sau phải chờ các khối trước thực hiện xong.

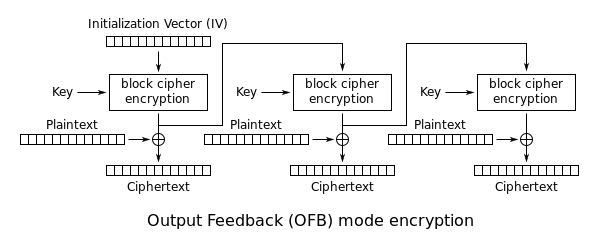
## *CFB (Cipher Feedback)*

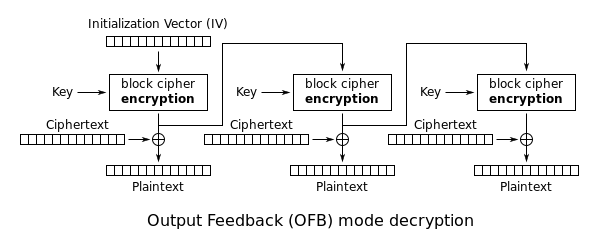




CFB ở dạng đơn giản nhất của nó được sử dụng toàn bộ đầu ra của mật mã khối. Trong biến thể này, nó rất giống với CBC, biến mật mã khối thành mật mã dòng tự đồng bộ hóa. Giải mã CFB trong biến thể này gần giống với mã hóa CBC được thực hiện ngược lại. Giống như chế độ CBC, các thay đổi trong bản rõ truyền mãi mãi trong bản mã và mã hóa không thể được thực hiện song song. Cũng giống như CBC, giải mã có thể được song song hóa.

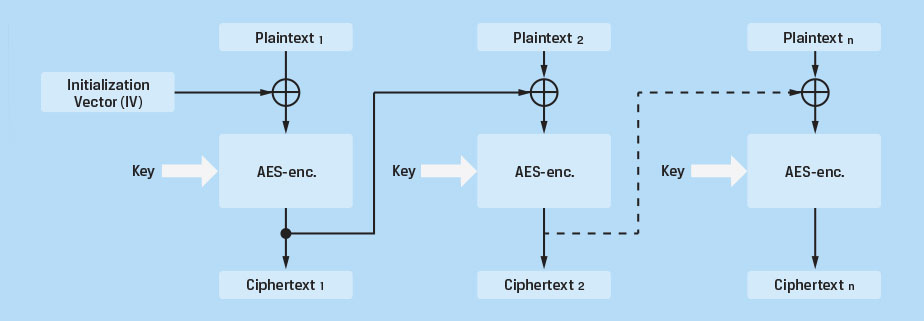
## *OFB (Output Feedback)*





Chế độ làm cho một thuật toán mã hóa khối thành một đồng bộ dòng mật mã . Nó tạo ra các khối keystream , sau đó được XOR với các khối plaintext để lấy bản mã.Vì bản rõ hoặc bản mã chỉ được sử dụng cho XOR cuối cùng, các hoạt động mật mã khối có thể được thực hiện trước, cho phép bước cuối cùng được thực hiện song song khi bản rõ hoặc bản mã có sẵn.

## *XTS*



XTS sử dụng hai khóa AES. Một khóa được sử dụng để thực hiện mã hóa khối AES; khóa còn lại được sử dụng để mã hóa những gì được coi là "Giá trị Tweak." Tweak đã được mã hóa này được điều chỉnh thêm bằng hàm đa thức Galois (GF) và XOR với cả văn bản đơn thuần và bảng mã của mỗi khối. Hàm GF giúp khuyếch tán nhiều hơn và đảm bảo rằng các khối dữ liệu giống nhau sẽ không tạo ra các bản mã giống nhau. Điều này đạt được mục tiêu là mỗi khối sẽ tạo ra bảng mã duy nhất từ văn bản đơn thuần giống nhau mà không sử dụng vectơ khởi tạo và chuỗi. Trên thực tế, hầu hết văn bản được mã hóa kép sử dụng hai khóa riêng biệt. Thực hiện việc giải mã dữ liệu bằng cách đảo ngược quy trình. Do mỗi khối là độc lập và không có chuỗi, nếu dữ liệu mã hóa được lưu trữ bị phá hủy hoặc bị hỏng, chỉ dữ liệu của khối cụ thể đó là không thể khôi phục. Với các chế độ chuỗi, những lỗi này có thể truyền sang các khối khác khi giải mã.

# Bảo mật hệ thống camera an ninh

* 1. **Tổng quan hệ thống camera an ninh**

## *Hoạt động của hệ thống*

* Giám sát

Nhận ảnh từ raspberry pi và phát video trực tiếp trên server.

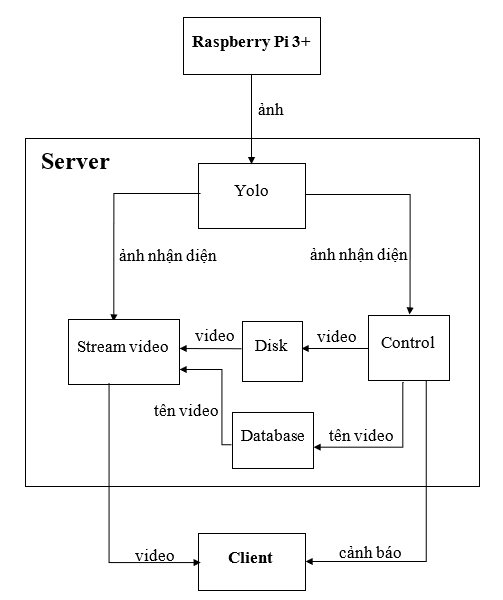
* Cảnh báo

Khi phát hiện người trong hình ảnh camera ghi lại, hệ thống sẽ gửi mail cảnh báo đến người dùng.

* Lưu trữ video

Khi phát hiện người trong hình ảnh camera ghi lại, hệ thống sẽ lưu video và người dùng có thể xem lại video đã lưu trên server.

## *Sơ đồ khối chức năng của hệ thống*

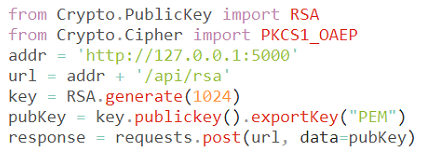


* **Raspberry Pi 3+**: Có chức năng chụp ảnh gửi lên server để xử lí
* **Yolo**: Sử dụng Yolo V4 tiny để xử lí và phát hiện người trong ảnh. Sau đó gửi ảnh đã xử lí đến Stream video và Control
* **Contro**l: Nhận ảnh đã xử lí từ Yolo để lưu video và gửi cảnh báo khi phát hiện người.
* **Stream video**: Có 2 chức năng. Thứ nhất là nhận ảnh từ Yolo và phát trực tiếp đến Client. Thứ hai là đọc video từ ổ đĩa để phát những video đã lưu.
* **Client**: Truy cập website để xem video trực tiếp hoặc video đã lưu.
  1. **Bảo mật hệ thống camera an ninh**

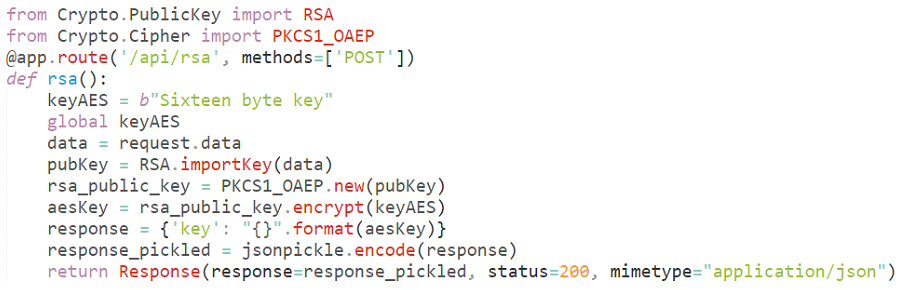
## *Bảo mật truyền thông giữa Raspberry và Server*

Sử dụng mã hóa đối xứng AES để mã hóa dữ liệu truyền từ raspberry pi đến server và sử dụng mã hóa công khai RSA để trao đổi khóa mã hóa AES

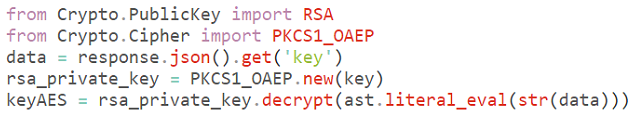
* Trao đổi khóa mã hóa AES bằng mã hóa công khai RSA
* Tạo và gửi khóa công khai RSA trên raspberry pi

****

* Nhận khóa công khai RSA và gửi bản mã của khóa mã hóa AES trên server

****

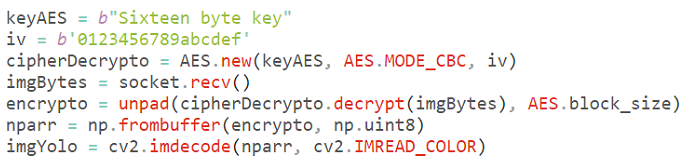
* Nhận bản mã của khóa mã hóa AES và giải mã bằng khóa bí mật RSA



* Mã hóa dữ liệu bằng mã hóa AES
* Mã hóa và gửi bản mã trên raspberry

****

* Nhận bản mã và giải mã trên server

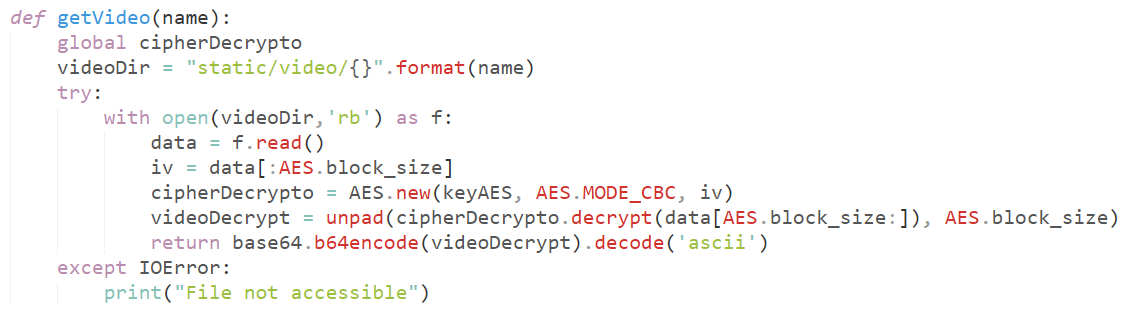
****

### **Bảo mật lưu trữ video**

* Mã hóa video

****

* Giải mã video

****

# KẾT LUẬN