

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
लोक विज्ञ

**BÀI GIẢNG  
AN NINH MẠNG VIỄN THÔNG**

*Chuyên ngành Điện tử truyền thông  
(Lưu hành nội bộ )*

**Biên soạn:**    **TS. Nguyễn Chiến Trinh**  
                        **PGS.TS. Nguyễn Tiến Ban**  
                        **TS. Hoàng Trọng Minh**  
                        **ThS. Nguyễn Thanh Trà**  
                        **ThS. Phạm Anh Thư**

## **LỜI NÓI ĐẦU**

# Mục lục

Mục lục .....	i
Danh mục hình vẽ.....	iii
<b>Chương 1: Tổng quan an toàn mạng truyền thông .....</b>	<b>1</b>
1.1 Khái niệm an toàn mạng truyền thông .....	1
1.2 Kiến trúc an toàn .....	1
1.3 Tấn công mạng .....	2
1.4 Dịch vụ an toàn.....	4
1.5 Các cơ chế an toàn .....	7
1.6 Mô hình an toàn mạng .....	8
1.7 Kết luận chương .....	10
Câu hỏi ôn tập chương 1 .....	10
<b>Chương 2: Mật mã khóa đối xứng .....</b>	<b>12</b>
2.1 Mô hình mật mã hóa khóa đối xứng.....	12
2.2 Mật mã khối và tiêu chuẩn mật mã hóa dữ liệu DES .....	16
2.2.1 Cấu trúc mật mã khối .....	16
2.2.1.1. Cấu trúc chung của mật mã khối.....	17
2.2.1.2 Cấu trúc mật mã khối Feistel.....	19
2.2.2 DES .....	23
2.2.2.1 Cấu trúc DES.....	23
2.2.2.2 Hoán vị khởi tạo và hoán vị kết thúc .....	24
2.2.2.3 Các vòng mật mã của DES .....	25
2.2.2.4 Thuật toán sinh khóa con của DES .....	27
2.2.2.5 Hiệu ứng lan truyền.....	27
2.2.3 Nguyên lí thiết kế mật mã khối .....	29
2.3 Tiêu chuẩn mật mã hóa tiên tiến AES .....	30
2.3.1 Cấu trúc AES .....	30
2.3.2 Các hàm biến đổi AES .....	34
2.3.2.1 Hàm SubBytes.....	34
2.3.2.2 Hàm ShiftRows .....	36
2.3.2.3 Hàm MixColumns .....	37
2.3.2.4 Hàm AddRoundKey .....	38
2.3.3 Tạo khóa AES .....	40
2.3.4 Thực hiện AES .....	41
2.4 Các ứng dụng của mật mã khối.....	45
2.4.1 Mật mã hóa nhiều lần .....	45
2.4.2 Các chế độ và ứng dụng mật mã khối .....	47
2.5 Tạo số giả ngẫu nhiên và mật mã dòng.....	54
2.5.1 Nguyên lí tạo số giả ngẫu nhiên .....	54
2.5.2 Bộ tạo số giả ngẫu nhiên .....	56

2.5.3 Mật mã dòng .....	59
2.5.4 RC4 .....	60
<b>2.6 Kết luận chương 2 .....</b>	<b>62</b>
<b>Câu hỏi ôn tập chương 2 .....</b>	<b>62</b>
<b>Chương 3: Mật mã khóa bất đối xứng .....</b>	<b>64</b>
<b>    3.1. Mật mã khóa công khai và RSA.....</b>	<b>64</b>
3.1.1 Nguyên lý hệ thống mật mã khóa công khai .....	64
<b>        3.1.1.1 Hệ mật khẩu công khai .....</b>	<b>64</b>
<b>        3.1.1.2 Các ứng dụng cho hệ mật khẩu công khai .....</b>	<b>69</b>
<b>        3.1.1.3 Các yêu cầu đối với hệ mật khẩu công khai.....</b>	<b>70</b>
3.1.2 Giải thuật RSA .....	71
<b>    3.2 Trao đổi khóa Diffie-Hellman.....</b>	<b>84</b>
<b>    3.3 Hệ thống mật mã Elgamal.....</b>	<b>89</b>
<b>    3.4 Tạo số giả ngẫu nhiên sử dụng mật mã bất đối xứng.....</b>	<b>93</b>
<b>    Câu hỏi ôn tập chương 3 .....</b>	<b>95</b>
<b>Chương 4: Các giải thuật toàn vẹn dữ liệu .....</b>	<b>96</b>
<b>    4.1 Hàm băm .....</b>	<b>96</b>
4.1.1 Ứng dụng của hàm băm .....	96
4.1.2 Các yêu cầu và độ an toàn hàm băm .....	98
<b>    4.2 Mã xác thực bản tin MAC .....</b>	<b>101</b>
4.2.1 Các yêu cầu xác thực bản tin.....	101
4.2.2 Chức năng xác thực bản tin.....	102
4.2.3 Các yêu cầu cho mã xác thực bản tin .....	110
4.2.4 Tính an toàn của MAC .....	113
4.2.5 MAC dựa trên hàm băm HMAC .....	115
4.2.7 Mật mã được xác thực .....	124
<b>    Câu hỏi ôn tập chương 4 .....</b>	<b>132</b>
<b>Chương 5: Xác thực .....</b>	<b>134</b>
<b>    5.1 Quản lý và phân phối khóa .....</b>	<b>134</b>
5.1.1 Phân phối khóa đối xứng sử dụng mật mã hóa đối xứng .....	134
5.1.2 Phân phối khóa đối xứng bằng mật mã hóa bát đối xứng .....	137
5.1.3 Phân phối khóa công khai .....	139
5.1.4 Chứng thư X.509 .....	142
<b>    5.2 Xác thực người sử dụng .....</b>	<b>147</b>
5.2.1 Nguyên lý xác thực người sử dụng từ xa .....	147
5.2.2 Xác thực người dùng sử dụng mật mã khóa đối xứng.....	150
<b>    Câu hỏi ôn tập chương 5 .....</b>	<b>154</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>155</b>

# Danh mục hình vẽ

Hình 1.1: Tấn công thụ động .....	2
Hình 1.2: Tấn công tích cực.....	3
Hình 1.3: Mối quan hệ giữa các dịch vụ an toàn và các cơ chế an toàn .....	8
Hình 1.4: Mô hình an toàn mạng .....	9
Hình 1.5: Mô hình an toàn truy nhập mạng .....	10
Hình 2.1: Mô hình mật mã khóa đối xứng đơn giản.....	12
Hình 2.2: Mô hình hệ thống mật mã hóa đối xứng.....	13
Hình 2.3: Cấu trúc mật mã khối.....	17
Hình 2.4: Nguyên lý của phép thay thế khối n bit đầu vào n bit đầu ra (n=4).....	18
Hình 2.5: Cấu trúc mật mã hóa và giải mật mã Feistel.....	21
Hình 2.6: Ví dụ về mật mã hóa và giải mật mã Feistel.....	22
Hình 2.7: Thuật toán mật mã DES.....	24
Hình 2.8: Cấu trúc một vòng mật mã DES .....	25
Hình 2.9: Cấu trúc AES .....	31
Hình 2.10: Khóa và khóa được mở rộng.....	32
Hình 2.11: Sơ đồ mật mã và giải mật mã AES .....	33
Hình 2.12: Vòng mật mã AES .....	34
Hình 2.13: Hàm SubBytes .....	35
Hình 2.14: S-box cho mật mã hóa .....	36
Hình 2.15: S-box cho giải mật mã .....	36
Hình 2.16: Ví dụ về biến đổi của hàm SubBytes .....	36
Hình 2.17: Thực hiện dịch hàng của hàm ShiftRows .....	37
Hình 2.18: Ví dụ dịch vòng của hàm ShiftRows .....	37
Hình 2.19: Hàm MixColumns.....	37
Hình 2.20: Hàm AddRoundKey .....	38
Hình 2.21: Các đầu vào cho một vòng mật mã của AES .....	39
Hình 2.22: Thuật toán tạo khóa AES .....	41
Hình 2.23: Mật mã nghịch đảo tương đương .....	43
Hình 2.24: Mật mã hóa nhiều lần.....	46
Hình 2.25: Mô hình mật mã hóa và giải mật mã của ECB .....	48
Hình 2.26: Mô hình mật mã hóa và giải mật mã CBC .....	49
Hình 2.27: Chế độ CFB .....	51
Hình 2.28: Chế độ OFB .....	52
Hình 2.29: Chế độ CTR .....	54
Hình 2.30: Nguyên lý tạo số ngẫu nhiên và giả ngẫu nhiên .....	56
Hình 2.31: Sơ đồ khối bộ tạo BBS.....	58
Hình 2.32: Sơ đồ mật mã dòng .....	60
Hình 2.33: Mật mã hóa RC4 .....	62
Hình 4.1: Ví dụ về sử dụng hàm băm trong nhận thực bản tin.....	97
Hình 4.2: Các cách sử dụng cơ bản của mã hóa bản tin .....	103
Hình 4.3: Điều khiển lỗi trong và ngoài .....	105
Hình 4.4: Phân đoạn TCP .....	105

Hình 4.5: Các cách dùng cơ bản của mã xác thực bản tin MAC .....	108
Hình 4.6: Cấu trúc HMAC.....	117
Hình 4.7: Sự thực hiện HMAC hiệu quả .....	120
Hình 4.8: Thuật toán nhận thực dữ liệu .....	122
Hình 4.9: Mã hóa nhận thực bản tin dựa trên mật mã .....	123
Hình 4.10: Bộ đếm với chuỗi khối mã hóa - mã nhận thực bản tin .....	126
Hình 4.11: Chức năng mã hóa và nhận thực GCM.....	128
Hình 4.12: Bộ đếm galois- Mã nhận thực bản tin.....	129
Hình 4.13: Kiến trúc cơ sở của hàm băm dựa trên PRNG.....	131
Hình 5.1: Số lượng các khóa yêu cầu cho các kết nối ngẫu nhiên giữa các điểm cuối .....	134
Hình 5.2: Mô hình phân cấp khóa.....	135
Hình 5.3: Kịch bản phân phối khóa .....	136
Hình 5.4: thủ tục phân phối khóa bí mật đơn giản.....	138
Hình 5.5: thủ tục phân phối khóa bí mật cung cấp bảo mật và nhận thực .....	138
Hình 5.6: Phân phối khóa tự do .....	139
Hình 5.7: Phân phối khóa qua thư mục khóa công khai .....	140
Hình 5.8: Kịch bản phân phối khóa công khai.....	141
Hình 5.9: Trường mở rộng của chứng chỉ X.509.....	143

## Danh mục bảng biểu

Bảng 2.1: Các kiểu tấn công .....	15
Bảng 2.2: Các kiểu ánh xạ .....	17
Bảng 2.3: Bảng mật mã hóa và giải mật mã cho mật mã khối thay thế của hình 2.4 .....	18
Bảng 2.4: Ví dụ hiệu ứng lan truyền.....	28
Bảng 2.5: Ví dụ hoạt động của bộ tạo BBS .....	59
Bảng 3.1: Mã khóa công khai và truyền thống .....	67
Bảng 3.2: Các ứng dụng cho hệ mật khóa công khai.....	70
Bảng 3.3: Tiến trình tìm ra thừa số trong RSA.....	80
Bảng 4.1: Các yêu cầu hàm băm bảo mật.....	99

## Chương 1: Tổng quan an toàn mạng truyền thông

### 1.1 Khái niệm an toàn mạng truyền thông

Trước đây khi công nghệ máy tính chưa phát triển, khi nói đến vấn đề an toàn bảo mật thông tin (Information Security), chúng ta thường hay nghĩ đến các biện pháp nhằm đảm bảo cho thông tin được trao đổi hay cất giữ một cách an toàn và bí mật. Chẳng hạn là các biện pháp như:

- + Đóng dấu và ký niêm phong một bức thư để biết rằng lá thư có được chuyển nguyên vẹn đến người nhận hay không.
- + Dùng mật mã hóa thông điệp để chỉ có người gửi và người nhận hiểu được thông điệp. Phương pháp này thường được sử dụng trong chính trị và quân sự.
- + Lưu giữ tài liệu mật trong các két sắt có khóa, tại các nơi được bảo vệ nghiêm ngặt, chỉ có những người được cấp quyền mới có thể xem tài liệu.

Với sự phát triển mạnh mẽ của công nghệ thông tin, đặc biệt là sự phát triển của mạng Internet, ngày càng có nhiều thông tin được lưu giữ trên máy vi tính và gửi đi trên mạng Internet. Và do đó xuất hiện nhu cầu về an toàn và bảo mật thông tin trên máy tính.

Có thể phân loại mô hình an toàn mạng thông tin trên máy tính theo hai hướng chính như sau:

- 1) Bảo vệ thông tin trong quá trình truyền thông tin trên mạng (Network Security)
- 2) Bảo vệ hệ thống máy tính, và mạng máy tính, khỏi sự xâm nhập phá hoại từ bên ngoài (System Security)

### 1.2 Kiến trúc an toàn

ITU-T đã đưa ra khuyến nghị X.800 định nghĩa kiến trúc an toàn cho mô hình OSI. Kiến trúc an toàn OSI giúp cho các nhà quản lý trong việc tổ chức cung cấp dịch vụ an toàn. Hơn nữa, do kiến trúc này được phát triển như là chuẩn quốc tế, các nhà cung cấp cơ sở hạ tầng cũng như nhà cung cấp thiết bị và dịch vụ có thể triển khai các đặc tính an toàn cho các sản phẩm và dịch vụ của họ.

Kiến trúc an toàn tập trung vào các kiểu tấn công, các cơ chế an toàn, và các dịch vụ an toàn. Các đặc điểm này được định nghĩa ngắn gọn như sau:

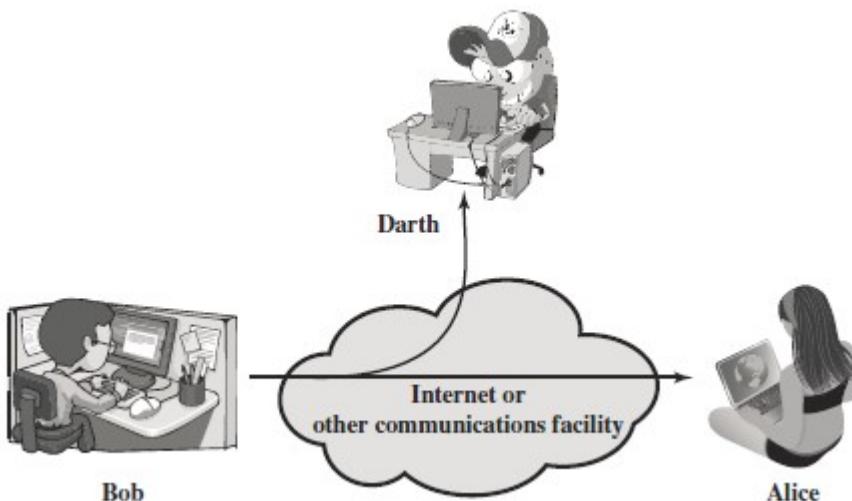
- + Tấn công an toàn: bất kỳ hành động nào mà làm hại đến tính an toàn thông tin của một tổ chức nào đó.
- + Cơ chế an toàn: quá trình được thiết kế để phát hiện, ngăn ngừa, hay khôi phục lại các kiểu tấn công an toàn.
- + Dịch vụ an toàn: dịch vụ truyền thông làm tăng cường tính an toàn của hệ thống xử lý dữ liệu và thông tin của một tổ chức. Các dịch vụ này thường dùng để chống lại các tấn công an toàn, và các dịch vụ này tận dụng một hoặc nhiều cơ chế an toàn để cung cấp dịch vụ.

### 1.3 Tấn công mạng

Về cơ bản, tấn công mạng được chia thành 2 loại đó là tấn công thụ động và tấn công tích cực. Tấn công thụ động là việc cố gắng lấy hoặc lợi dụng thông tin hệ thống nhưng không ảnh hưởng đến các tài nguyên hệ thống. Tấn công tích cực là các hành động cố gắng thay đổi các tài nguyên hệ thống hoặc gây ảnh hưởng đến hoạt động của họ.

#### Các kiểu tấn công thụ động

Các kiểu tấn công thụ động (hình 1.1) về bản chất là các hành động nghe trộm, hoặc giám sát các hoạt động truyền thông. Mục tiêu của kẻ tấn công là lấy được thông tin đang được truyền đi. Hai kiểu của tấn công thụ động là xem trộm các nội dung bản tin và phân tích luồng thông tin.



Hình 1.1: Tấn công thụ động

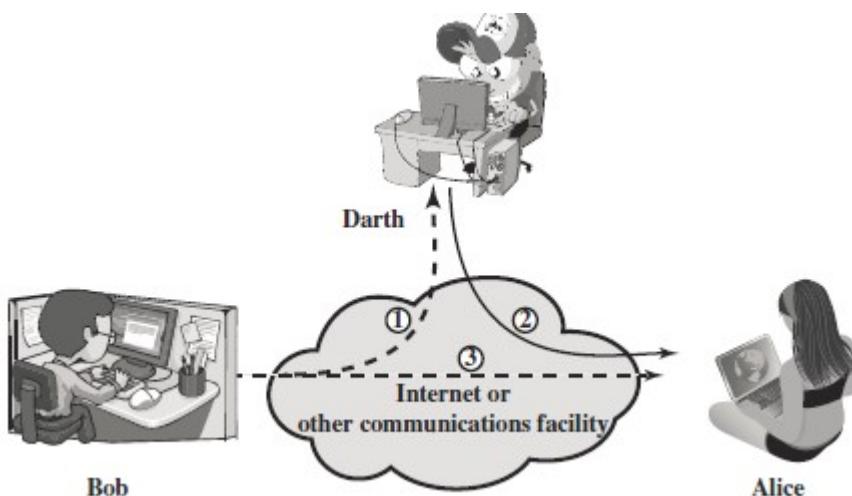
Kiểu tấn công xem trộm nội dung bản tin: cuộc điện thoại, mail điện tử, và file được truyền đi có thể chứa các thông tin bí mật hoặc nhạy cảm. Kẻ tấn công sẽ tấn công để xem trộm được các thông tin bí mật hoặc nhạy cảm đó.

Kiểu tấn công thụ động thứ hai, phân tích luồng thông tin: giả thiết rằng đã có cách để che dấu các nội dung bản tin hoặc lưu lượng thông tin khác để các kẻ tấn công, thậm chí họ chỉ bắt các bản tin, không thể tách thông tin từ bản tin đó. Kỹ thuật chung để che dấu thông tin là mã hóa. Nếu bản tin đã được mã hóa, kẻ tấn công có thể vẫn có khả năng quan sát được mẫu các bản tin này. Kẻ tấn công có thể xác định vị trí và nhận dạng các thiết bị truyền thông và có thể quan sát được tần suất và độ dài các bản tin đang được trao đổi. Thông tin này có thể là hữu ích cho việc đoán bản chất của quá trình truyền thông đang xảy ra.

Các kiểu tấn công thụ động là rất khó để phát hiện, bởi chúng không liên quan đến bất kỳ sự thay đổi nào của dữ liệu. Cụ thể là, lưu lượng bản tin được gửi và nhận theo một cách thông thường nào đó, và cả người gửi và người nhận đều không phát hiện ra sự có mặt của bên thứ ba đang đọc các bản tin hoặc đang quan sát các mẫu lưu lượng. Tuy nhiên, có thể ngăn ngừa kiểu tấn công này bằng cách sử dụng các kiểu mật mã hóa. Do đó, đối với kiểu tấn công này, phòng ngừa tốt hơn là phát hiện.

### Các kiểu tấn công tích cực

Các kiểu tấn công tích cực (hình 1.2) liên quan đến việc sửa đổi dòng dữ liệu hoặc tạo dòng dữ liệu sai lệch và có thể được chia thành bốn loại sau: mạo danh (Masquerade), phát lại bản tin (replay), sửa đổi bản tin, và từ chối dịch vụ.



Hình 1.2: Tấn công tích cực

**Tấn công mạo danh:** tấn công mạo danh là tấn công mà kẻ tấn công mạo danh bên gửi tin để gửi bản tin cho bên nhận. Bên nhận không biết sự mạo danh đó và vẫn nghĩ là bản tin được gửi từ bên gửi hợp lệ.

**Tấn công phát lại:** liên quan đến việc sao chép thụ động dữ liệu và sau đó gửi lại bản sao chép đó cho bên nhận. Thoạt đầu có thể nghĩ rằng việc phát lại này là vô hại, tuy nhiên trong nhiều trường hợp cũng gây ra tác hại không kém so với tấn công mạo danh. Xét tình huống sau: giả sử Alice là ngân hàng còn Bod là một khách hàng. Bod gửi bản tin đề nghị Alice chuyển cho Darth 1000\$. Bod có áp dụng các biện pháp như chữ ký điện tử với mục đích không cho Darth mạo danh cũng như sửa thông tin. Tuy nhiên nếu Darth sao chép và phát lại bản tin đó thì các biện pháp bảo vệ này không có ý nghĩa. Alice tin rằng Bod gửi tiếp một bản tin mới để chuyển thêm cho Darth 1000\$ nữa.

**Thay đổi bản tin:** Darth chặn các bản tin Bod gửi cho Alice và ngăn không cho các bản tin này đến đích. Sau đó Darth thay đổi nội dung của bản tin và gửi tiếp cho Alice. Alice nghĩ rằng nhận được bản tin nguyên bản ban đầu của Bod mà không biết rằng chúng đã bị sửa đổi. Ví dụ, Bod gửi bản tin cho Alice là “Cho phép John đọc được các account file bí mật”, bản tin đó bị sửa đổi thành “Cho phép Fred đọc được các account file bí mật”.

**Tấn công từ chối dịch vụ:** kiểu tấn công này có một mục tiêu cụ thể; ví dụ kẻ tấn công chặn toàn bộ các bản tin được chuyển tới một đích nào đó. Một loại hình khác của kiểu tấn công này là làm sập hoàn toàn mạng, có thể bằng cách làm mất khả năng hoạt động của mạng hoặc làm quá tải mạng với các bản tin gửi liên tiếp tới mạng đó để làm suy giảm hiệu năng mạng.

Kiểu tấn công tích cực có thể có chủ ý cụ thể, ví dụ một kẻ tấn công có thể ngăn cản tất cả các thông báo được chuyển tới một đích nào đó, vô hiệu hóa một mạng hoặc tạo ra tình trạng quá tải với các thông báo của họ làm giảm hiệu năng mạng.

Có thể thấy rằng hai kiểu tấn công chủ động và thụ động có những đặc trưng khác nhau. Kiểu tấn công thụ động khó phát hiện nhưng có biện pháp để ngăn chặn thành công. Mặt khác kiểu tấn công chủ động dễ phát hiện nhưng lại rất khó ngăn chặn tuyệt đối, nó cũng đòi hỏi việc bảo vệ vật lý tất cả các phương tiện truyền thông ở mọi lúc, mọi nơi. Giải pháp để chống lại kiểu tấn công này là phát hiện chúng và khôi phục mạng khi bị phá vỡ hoặc khi thông tin bị trễ.

## 1.4 Dịch vụ an toàn

X.800 định nghĩa dịch vụ an toàn là một dịch vụ được cung cấp bởi lớp giao thức của các hệ thống truyền thông và đảm bảo tính an toàn của các hệ thống hoặc của việc truyền dữ liệu. RFC 4949 định nghĩa dịch vụ an toàn thực hiện các chính sách an toàn và được thực thi bởi các cơ chế an toàn.

X.800 chia các dịch vụ này thành năm loại và 14 dịch vụ cụ thể như sau.

### **Dịch vụ xác thực:**

Dịch vụ xác thực liên quan đến việc đảm bảo rằng quá trình truyền thông được xác thực nghĩa là cả người gửi và người nhận không bị mạo danh. Trong trường hợp chỉ có một thông tin, như là tín hiệu cảnh báo hoặc báo thức, chức năng của dịch vụ xác thực là đảm bảo với người nhận rằng bản tin đó đến từ nguồn được xác thực. Trong trường hợp có sự tương tác xảy ra, ví dụ như sự kết nối của đầu cuối với thiết bị đầu cuối khác, đầu tiên, tại thời điểm khởi tạo kết nối, dịch vụ xác thực đảm bảo rằng hai thực thể đều được xác thực. Sau đó, dịch vụ này phải đảm bảo rằng kết nối là không bị cản trở theo cách đó bên thứ ba có thể mạo danh như là một trong hai bên hợp pháp để thực hiện việc nhận và truyền dẫn không được phép.

Hai loại dịch vụ xác thực được định nghĩa trong X.800:

- + Xác thực toàn bộ các peer: cung cấp chứng thực nhận dạng thực thể peer trong một liên kết. Hai thực thể được gọi là peer nếu chúng thực thi cùng giao thức trong các hệ thống khác nhau. Xác thực peer được thực hiện tại thời điểm thiết lập kết nối hoặc tại các thời điểm trong suốt pha truyền dữ liệu của kết nối.
- + Xác thực dữ liệu: cung cấp chứng thực nguồn dữ liệu. Dịch vụ này không cung cấp bảo vệ chống lại việc nhân bản hoặc chỉnh sửa dữ liệu. Kiểu dịch vụ này hỗ trợ các ứng dụng không có tương tác trước đó giữa các thực thể truyền thông như thư điện tử.

### **Điều khiển truy nhập**

Trong ngữ cảnh an toàn mạng, điều khiển truy nhập có khả năng hạn chế và điều khiển việc truy nhập tới các hệ thống và các ứng dụng qua các liên kết truyền thông. Để đạt được điều này, mỗi thực thể cố gắng truy nhập đầu tiên phải được nhận dạng, hoặc nhận thực, thì mới được phép truy cập các phần tử mạng, thông tin lưu trữ, luồng thông tin, dịch vụ và ứng dụng mạng.

### **Dịch vụ bảo mật dữ liệu**

Dịch vụ bảo mật dữ liệu là thực hiện bảo vệ dữ liệu được truyền đi khỏi các kiểu tấn công thụ động. Có một số mức bảo vệ được định nghĩa. Mức rộng nhất là bảo vệ toàn bộ dữ liệu của người sử dụng được truyền đi giữa hai bên qua một khoảng thời gian nào đó. Mức hẹp nhất của dịch vụ bảo mật dữ liệu là bảo vệ một bản tin đơn hoặc thậm chí một vài trường cụ thể nào đó trong một bản tin. Một khía cạnh khác của dịch vụ bảo mật là bảo vệ luồng dữ liệu khỏi kẻ tấn công. Điều đó yêu cầu kẻ tấn công không thể theo dõi được phía nguồn, phía đích, tần suất, độ dài, hay các đặc tính khác của lưu lượng trên một phương tiện truyền thông.

### **Dịch vụ toàn vẹn dữ liệu**

Cũng giống như dịch vụ bảo mật dữ liệu, dịch vụ toàn vẹn dữ liệu có khả năng áp dụng cho một dòng bản tin, một bản tin, hay một số trường xác định trong một bản tin.

Dịch vụ toàn vẹn hướng kết nối đảm bảo rằng các bản tin được nhận mà không bị lặp, chèn, chỉnh sửa, sai thứ tự, hay truyền lại. Sự phá hoại dữ liệu có thể được khôi phục bởi dịch vụ này. Do đó, dịch vụ toàn vẹn hướng kết nối giải quyết được kiểu tấn công từ chối dịch vụ và chỉnh sửa dòng bản tin. Mặt khác, dịch vụ toàn vẹn hướng phi kết nối, chỉ thực hiện với từng bản tin riêng biệt, thường cung cấp sự bảo vệ chống lại việc chỉnh sửa bản tin.

### **Dịch vụ không thể chối bỏ (Nonrepudiation)**

Dịch vụ không thể chối bỏ ngăn chặn việc bên gửi hay bên nhận chối bỏ bản tin đã được truyền. Khi bản tin được gửi đi bên nhận có thể chứng minh rằng ben gửi hợp pháp đã gửi nó đi. Khi bản tin nhận được bên gửi có thể chứng minh rằng bản tin đó đã nhận được bởi bên nhận hợp pháp.

### **Các dịch vụ khả dụng**

Cả X.800 và RFC 4949 đều định nghĩa tính khả dụng là đặc tính của hệ thống hoặc tài nguyên hệ thống có khả năng truy cập và sử dụng dựa trên nhu cầu bởi một thực thể hệ thống được cấp quyền, tùy thuộc vào các đặc tả hiệu năng của hệ thống đó (nghĩa là hệ thống là khả dụng nếu nó cung cấp các dịch vụ theo thiết kế hệ thống bất cứ khi nào người sử dụng yêu cầu). Có rất nhiều kiểu tấn công có thể làm mất hoặc giảm tính khả dụng. Có một số cách tự động đối phó với các kiểu tấn công này như xác thực và mã hóa, trong khi một số cách khác yêu cầu một số biện pháp vật lý để phòng ngừa hoặc khôi phục việc mất tính khả dụng của các phần tử của các hệ thống.

## 1.5 Các cơ chế an toàn

Các cơ chế an toàn được định nghĩa trong X.800 được phân chia thành các cơ chế được thực thi trong lớp giao thức cụ thể, như TCP hay giao thức lớp ứng dụng, và các cơ chế không cụ thể với bất kỳ lớp giao thức nào hoặc dịch vụ an toàn nào. X.800 phân biệt các cơ chế mật mã hóa thuật nghịch và các cơ chế mật mã hóa không thuận nghịch. Cơ chế mật mã hóa thuật nghịch chỉ đơn giản là thuật toán mật mã cho phép dữ liệu được mã hóa và sau đó giải mã. Cơ chế mật mã hóa không thuận nghịch gồm các thuật toán hàm băm và các mã xác thực bản tin được sử dụng trong các ứng dụng xác thực và chữ ký điện tử. Các cơ chế an toàn xác định gồm:

- + **Mật mã hóa:** sử dụng các thuật toán mật mã để biến đổi dữ liệu thành một dạng dữ liệu khác. Sự biến đổi và khôi phục dữ liệu phụ thuộc vào thuật toán và không hoặc nhiều khóa bí mật.
- + **Chữ ký số:** dữ liệu được gắn thêm vào, hoặc biến đổi mật mã của, một đơn vị dữ liệu để cho phép bên nhận dữ liệu đó xác định được bên gửi và tính toàn vẹn dữ liệu, và chống lại được sự giả mạo.
- + **Điều khiển truy nhập:** Các cơ chế điều khiển truy nhập được dùng để đảm bảo rằng chỉ có một số người dùng được gán quyền mới có thể truy nhập tới các tài nguyên thông tin (tệp, tiến trình, cổng truyền thông) và các tài nguyên phần cứng (máy chủ in, Processor, Gateway).
- + **Tính toàn vẹn dữ liệu:** các cơ chế được sử dụng để đảm bảo tính toàn vẹn của một đơn vị dữ liệu hoặc của luồng dữ liệu.
- + **Trao đổi xác thực:** được sử dụng để đảm bảo định danh của người dùng bằng cách trao đổi thông tin.
- + **Đệm lưu lượng:** chèn các bit vào các khoảng trống của luồng dữ liệu để gây khó khăn cho kiểu tấn công phân tích lưu lượng.
- + **Điều khiển định tuyến:** cho phép lựa chọn các tuyến an toàn cụ thể nào đó và cho phép thay đổi định tuyến đặc biệt là khi có lỗ hổng an toàn đang xảy ra.
- + **Chứng thực:** sử dụng bên tin tưởng thứ 3 để đảm bảo các đặc tính xác định nào đó của việc trao đổi dữ liệu.

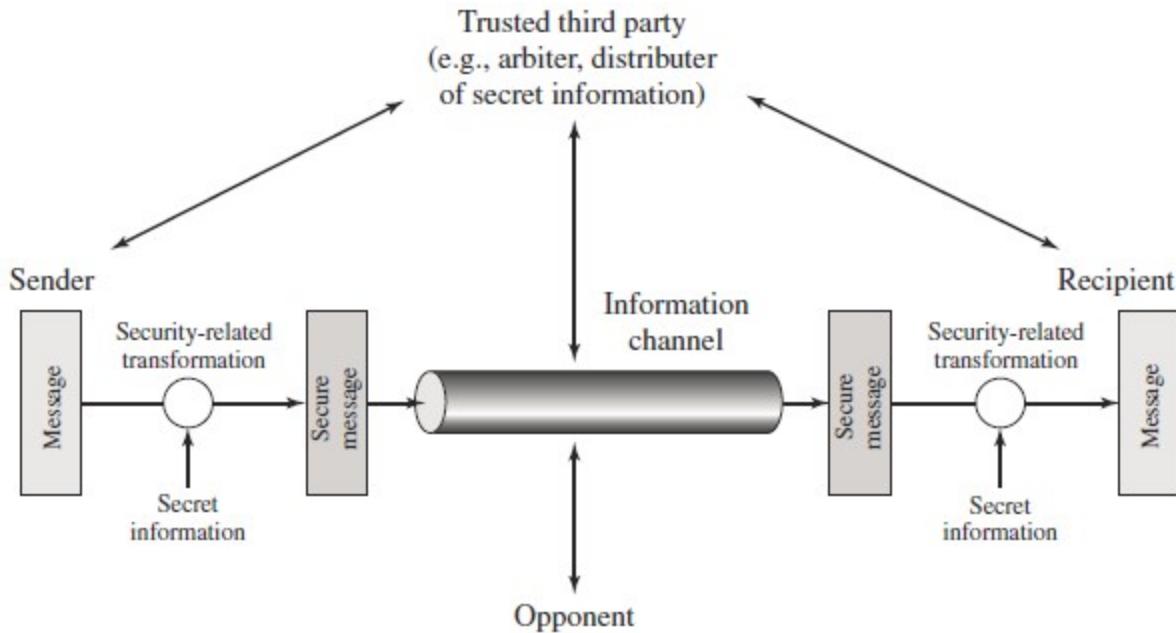
Hình dưới đây chỉ ra mối quan hệ giữa các dịch vụ an toàn và các cơ chế an toàn.

SERVICE	MECHANISM							
	Enchipherment	Digital signature	Access control	Data integrity	Authentication exchange	Traffic padding	Routing control	Notarization
Peer entity authentication	Y	Y			Y			
Data origin authentication	Y	Y						
Access control			Y					
Confidentiality	Y						Y	
Traffic flow confidentiality	Y					Y	Y	
Data integrity	Y	Y		Y				
Nonrepudiation		Y		Y				Y
Availability				Y	Y			

Hình 1.3: Mối quan hệ giữa các dịch vụ an toàn và các cơ chế an toàn

## 1.6 Mô hình an toàn mạng

Mô hình an toàn mạng được mô tả trong hình 1.4.



Hình 1.4: Mô hình an toàn mạng

Bản tin được truyền từ bên gửi đến bên nhận qua mạng Internet. Kênh thông tin logic được thiết lập bằng cách định nghĩa một tuyến qua mạng Internet từ nguồn tới đích và bằng cách sử dụng các giao thức truyền thông (TCP/IP).

Các khía cạnh an toàn được yêu cầu khi cần bảo vệ quá trình truyền thông khỏi kẻ tấn công. Tất cả các kĩ thuật cung cấp tính an toàn đều có hai thành phần:

- + Phép biến đổi an toàn lên thông tin được gửi đi. Ví dụ như mật mã hóa bản tin hay thêm mã vào nội dung bản tin.
- + Một số thông tin an toàn được chia sẻ bởi bên gửi và bên nhận. Ví dụ như khóa bí mật được sử dụng để mật mã hóa bản tin trước khi gửi đi.

Bên thứ ba chứng thực có thể được yêu cầu để đạt được truyền dẫn an toàn. Ví dụ, bên thứ ba có thể chịu trách nhiệm phân phối thông tin bí mật tới bên gửi và bên nhận mà không bị phát hiện bởi bất cứ kẻ tấn công nào.

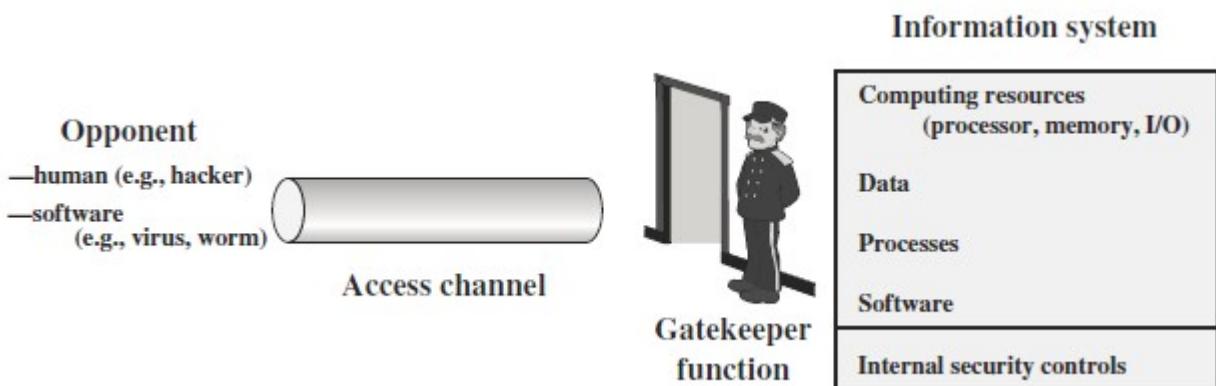
Có bốn nhiệm vụ cơ bản khi thiết kế dịch vụ an toàn cụ thể:

1. Thiết kế một thuật toán cho việc thực hiện biến đổi liên quan đến an toàn. Thuật toán này phải đảm bảo rằng kẻ tấn công không thể đánh bại được mục đích của nó.
2. Tạo thông tin bí mật được sử dụng cùng với thuật toán
3. Phát triển các phương pháp phân phối và chia sẻ thông tin bí mật

4. Chỉ rõ giao thức được sử dụng bởi bên gửi và bên nhận mà sử dụng thuật toán an toàn và thông tin bí mật để đạt được dịch vụ an toàn cụ thể.

Hình 1.5 trình bày mô hình an toàn truy nhập mạng nhằm bảo vệ hệ thống thông tin khỏi các truy nhập không mong muốn. Có hai loại tấn công đó là tấn công từ con người (hacker) và tấn công bằng các phần mềm như virus hay worm.

Các cơ chế an toàn cần thiết để đối phó với các truy nhập không mong muốn được phân thành hai loại. Loại thứ nhất là chức năng gatekeeper. Loại này bao gồm các thủ tục đăng nhập dựa trên mật khẩu được thiết kế để bảo vệ và loại bỏ các worm, virus và các kiểu tấn công tương tự khác. Loại thứ hai bao gồm các loại điều khiển trong nội bộ nhằm mục đích giám sát các hoạt động và phân tích thông tin lưu trữ để phát hiện ra sự có mặt của kẻ xâm nhập không mong muốn.



Hình 1.5: Mô hình an toàn truy nhập mạng

## 1.7 Kết luận chương

Chương 1 đã giới thiệu những khái niệm cơ bản liên quan đến các vấn đề an toàn mạng truyền thông và kiến trúc an toàn. Cũng trong chương này, hai kiểu tấn công chính đó là tấn công thụ động và tấn công tích cực được trình bày. Các dịch vụ an toàn và cơ chế an toàn đã được định nghĩa và sự tương quan giữa các dịch vụ và các cơ chế này cũng được đưa ra.

## Câu hỏi ôn tập chương 1

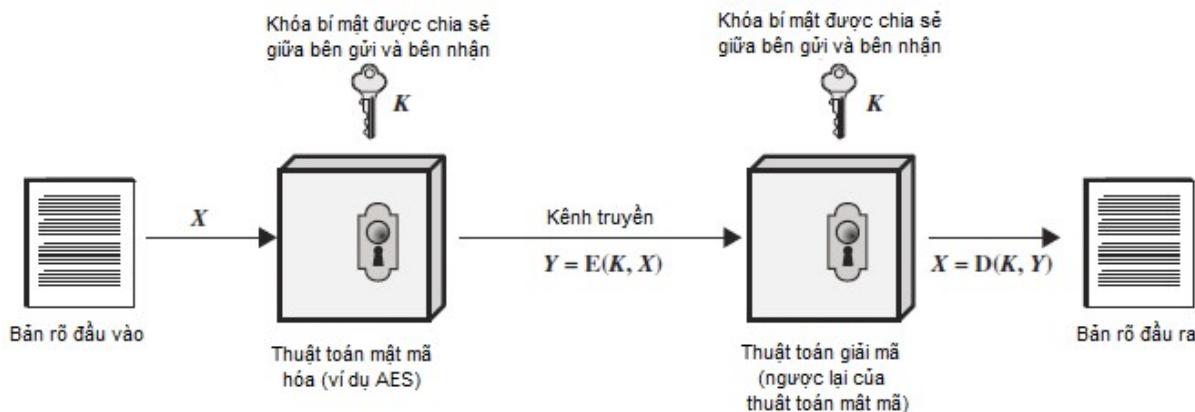
1. Trình bày kiến trúc an toàn mạng truyền thông.
2. Trình bày sự khác nhau giữa kiểu tấn công thụ động và kiểu tấn công tích cực.

3. Liệt kê và định nghĩa ngắn gọn các kiểu tấn công thụ động và tích cực
4. Trình bày các loại dịch vụ an toàn.
5. Trình bày các cơ chế an toàn.

## Chương 2: Mật mã khóa đối xứng

### 2.1 Mô hình mật mã hóa khóa đối xứng

Sơ đồ mật mã hóa đối xứng bao gồm 5 thành phần như chỉ ra trong hình vẽ 2.1 dưới đây.



Hình 2.1: Mô hình mật mã hóa khóa đối xứng đơn giản

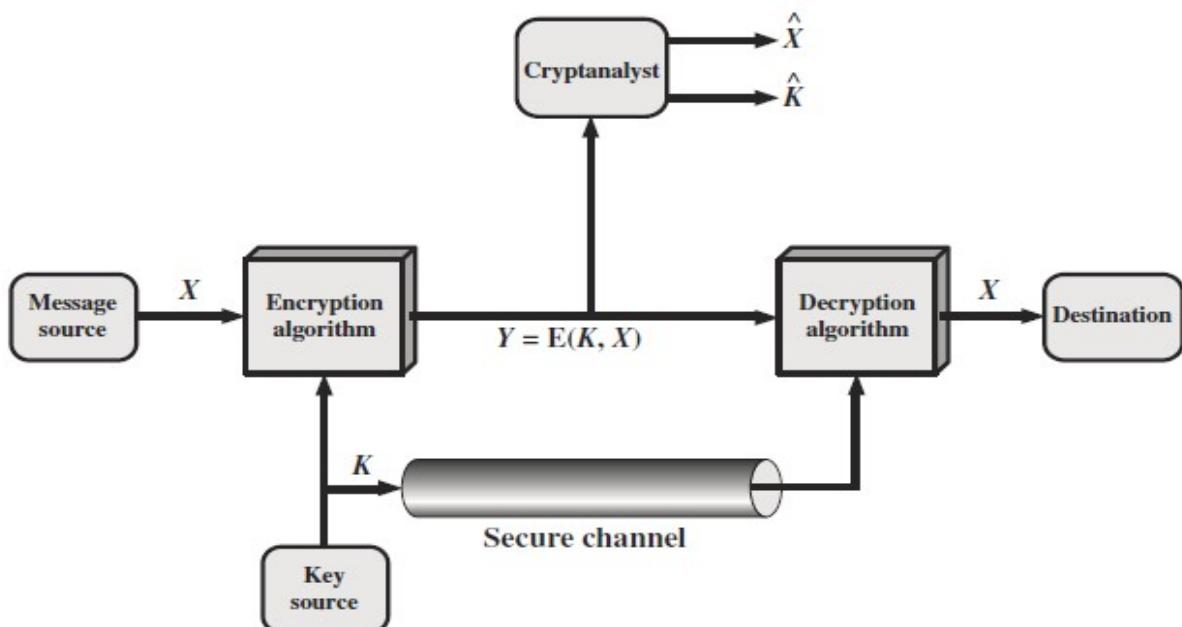
Năm thành phần của mô hình mật mã hóa khóa đối xứng đơn giản bao gồm:

- + Bản rõ: đây là dữ liệu hoặc bản tin ban đầu, được xem như là đầu vào của khối thuật toán mật mã.
- + Thuật toán mật mã hóa: thuật toán mật mã hóa thực hiện rất nhiều phép biến đổi và thay thế trên bản rõ.
- + Khóa bí mật: khóa bí mật cũng là một đầu vào của khối thuật toán mật mã hóa. Khóa là một giá trị độc lập với bản rõ và thuật toán. Thuật toán sẽ cho ra một đầu ra khác nhau phụ thuộc vào khóa cụ thể được sử dụng tại thời điểm đó. Các phép biến đổi và thay thế chính xác được thực hiện bởi thuật toán phụ thuộc vào khóa đó.
- + Bản mã: đây là bản tin đầu ra của thuật toán mật mã. Bản mã này phụ thuộc vào bản rõ và khóa bí mật. Với một bản tin xác định, hai khóa khác nhau sẽ tạo ra hai bản mã khác nhau.
- + Thuật toán giải mã: là thuật toán thực hiện ngược lại với thuật toán mật mã hóa. Khối này nhận bản mã và khóa bí mật để tạo ra bản rõ ban đầu.

Có hai yêu cầu cho việc sử dụng an toàn mật mã hóa truyền thống:

- + Một thuật toán mật mã hóa đủ mạnh được yêu cầu: tối thiểu là thuật toán mật mã hóa đó phải đảm bảo rằng kẻ tấn công (opponent) mặc dù biết được thuật toán và lấy được một hoặc nhiều bản mã nhưng không thể giải mật mã bản mã đó hoặc tìm ra khóa. Yêu cầu này thường được phát biểu như sau: kẻ tấn công không có khả năng giải mật mã bản mã hoặc khôi phục khóa thậm chí anh ta sở hữu một số các bản mã cùng với bản rõ được tạo ra từ mỗi bản mã đó.
- + Bên gửi và bên nhận phải có bản sao của khóa bí mật, và khóa phải được giữ bí mật giữa người gửi và người nhận, hay nói cách khác khóa phải được chuyển một cách an toàn từ người gửi đến người nhận.

Giả sử rằng việc giải mật mã bản tin là không thể thực hiện được dựa trên bản mã và sự hiểu biết về thuật toán mật mã hóa/giải mật mã. Nói cách khác, không cần phải giữ bí mật thuật toán mật mã hóa mà chỉ cần giữ bí mật khóa. Đặc điểm này của mật mã hóa đối xứng làm cho nó được sử dụng rộng rãi. Thực tế là thuật toán không cần được giữ bí mật nghĩa là các nhà sản xuất có thể và đã phát triển các mạch (chip) có chi phí thấp để thực thi các thuật toán mật mã hóa dữ liệu. Các chip này sẵn có và được tích hợp vào một số sản phẩm. Với việc sử dụng mật mã hóa đối xứng, vấn đề bảo mật được thực hiện ở việc bảo mật khóa bí mật. Như vậy, các phần tử cần thiết của sơ đồ mật mã hóa đối xứng được mô tả như trong hình 2.2.



Hình 2.2: Mô hình hệ thống mật mã hóa đối xứng

Nguồn bản tin tạo ra bản tin trong chế độ bản rõ,  $X = [X_1, X_2, \dots, X_M]$ .  $M$  phần tử của  $X$  là các chữ cái trong bảng chữ cái (alphabet). Theo truyền thống, bảng chữ cái gồm 26 chữ cái in hoa. Ngày nay, bảng chữ cái nhị phân  $\{0,1\}$  được sử dụng. Đối với mật mã hóa, khóa có dạng  $K = [K_1, K_2, \dots, K_J]$  được tạo ra. Nếu khóa đó được tạo ra tại phía nguồn bản tin, thì nó cũng phải được cung cấp cho bên nhận bằng một kênh an toàn. Nếu bên thứ ba tạo ra khóa bí mật, thì khóa đó sẽ được phân phối an toàn tới cả bên gửi và nhận.

Với bản tin  $X$  và khóa bí mật  $K$  là đầu vào, các thuật toán mật mã hóa tạo ra các bản mã  $Y = [Y_1, Y_2, \dots, Y_N]$ , được viết như sau:

$$Y = E(K, X)$$

Công thức này chỉ ra rằng  $Y$  được tạo ra bằng cách sử dụng thuật toán mật mã hóa  $E$  là một hàm của bản rõ,  $X$ , với một hàm xác định được quyết định bởi giá trị của khóa  $K$ .

Bên nhận mong muốn, có khóa bí mật, có khả năng thực hiện phép biến đổi sau:

$$X = D(K, Y)$$

Kẻ tấn công, thu được  $Y$  nhưng không có khóa  $K$  hoặc  $X$ , có thể cố gắng để khôi phục  $X$  hoặc  $K$  hoặc cả  $X$  và  $K$ . Giả thiết rằng kẻ tấn công đó biết thuật toán mật mã hóa  $E$  và thuật toán giải mã  $D$ . Nếu kẻ tấn công chỉ quan tâm đến một bản tin cụ thể, thì chỉ có gàng khôi phục  $X$  bằng cách tạo ra ước lượng bản rõ,  $\hat{X}$ . Tuy nhiên, thường thì kẻ tấn công quan tâm đến khả năng đọc được các bản tin tiếp theo, trong trường hợp đó phải khôi phục  $K$  bằng cách tạo ra ước lượng  $\hat{K}$ .

### Mật mã (Cryptography)

Các hệ thống mật mã được mô tả bởi ba khía cách độc lập dưới đây:

- Kiểu các cách thức được sử dụng để biến đổi từ bản rõ thành bản mã.** Tất cả các thuật toán mật mã hóa được dựa trên hai nguyên lý chung: thay thế, trong đó mỗi phần tử trong bản rõ (bit, chữ cái, nhóm bít hoặc nhóm chữ cái) được ánh xạ thành một phần tử khác; và hoán đổi vị trí, trong đó các phần tử trong bản rõ được sắp xếp lại. Yêu cầu cơ bản là không có thông tin nào bị mất (nghĩa là tất cả các hoạt động đó có thể được khôi phục). Hầu hết các hệ thống, còn được gọi là các hệ thống sản phẩm, bao gồm nhiều giai đoạn thay thế và biến đổi.

2. **Số khóa được sử dụng.** Nếu cả bên gửi và bên nhận sử dụng chung khóa, hệ thống đó được gọi là hệ thống mật mã hóa đối xứng, một khóa, khóa bí mật, hay truyền thống. Nếu bên gửi và nhận sử dụng các khóa khác nhau, hệ thống đó được gọi là hệ thống mật mã hóa bất đối xứng, hai khóa, hay khóa công khai.
3. **Cách mà bản rõ được xử lý.** Mật mã khỏi xử lý đầu vào là một khối các phần tử tại một thời điểm, tạo ra khối đầu ra cho mỗi khối đầu vào. Mật mã dòng (stream cypher) xử lý các phần tử đầu vào một cách liên tục, tạo ra phần tử một đầu ra tại một thời điểm.

### Giải mã các mật mã và tấn công Brute-Force

Mục tiêu tấn công hệ thống mật mã hóa là để khôi phục khóa đang dùng chứ không phải đơn giản là khôi phục bản rõ của một bản mã. Có hai cách chung để tấn công sơ đồ mật mã hóa truyền thống gồm:

- + Giải mã các mật mã (Cryptanalysis): các tấn công này dựa trên bản chất của thuật toán cộng với sự hiểu biết về các đặc tính chung của bản rõ hoặc thậm chí một vài cặp bản rõ – bản mã mẫu. Kiểu tấn công này lợi dụng các đặc tính của thuật toán để cố gắng suy luận ra bản rõ cụ thể hoặc để suy ra khóa được sử dụng.
- + **Kiểu tấn công Brute – Force:** kẻ tấn công thử các khóa có thể lên một đoạn bản mã cho tới khi biên dịch được thành bản rõ. **Trung bình, một nửa số khóa có thể phải được thử để đạt được thành công.**

Nếu một trong hai kiểu tấn công thực hiện thành công việc suy luận khóa, tất cả các bản tin trước đó và sau này đều đã được mật mã hóa sẽ bị tấn công.

Bảng 2.1 tóm tắt tất cả các kiểu tấn công giải mật mã các mật mã dựa trên khối lượng thông tin được biết bởi kẻ tấn công. Trong hầu hết các trường hợp, thậm chí thuật toán mật mã hóa không được biết, nhưng nhìn chung, có thể giả thiết rằng kẻ tấn công biết thuật toán được sử dụng cho việc mật mã hóa.

Bảng 2.1: Các kiểu tấn công

Kiểu tấn công	Thông tin được kẻ tấn công biết
Chỉ biết bản mã	<ul style="list-style-type: none"> <li>+ Thuật toán mật mã hóa</li> <li>+ Bản mã</li> </ul>
Biết một số cặp bản	<ul style="list-style-type: none"> <li>+ Thuật toán mật mã hóa</li> </ul>

rõ – bản mã (known-plaintext)	<ul style="list-style-type: none"> <li>+ Bản mã</li> <li>+ Một hoặc một số cặp bản rõ – bản mã được tạo ra với khóa bí mật.</li> </ul>
Biết bản rõ được lựa chọn (choose-plaintext)	<ul style="list-style-type: none"> <li>+ Thuật toán mật mã hóa</li> <li>+ Bản mã</li> <li>+ Bản tin bản rõ được lựa chọn bởi kẻ tấn công, cùng với bản mã tương ứng được tạo ra với khóa bí mật.</li> </ul>
Biết bản mã được lựa chọn (choose ciphertext)	<ul style="list-style-type: none"> <li>+ Thuật toán mật mã hóa</li> <li>+ Bản mã</li> <li>+ Bản mã được lựa chọn bởi kẻ tấn công, cùng với bản mã tương ứng được giải mã với khóa bí mật.</li> </ul>
Văn bản được lựa chọn (choose text)	<ul style="list-style-type: none"> <li>+ Thuật toán mật mã hóa</li> <li>+ Bản mã</li> <li>+ Bản tin bản rõ được lựa chọn bởi kẻ tấn công, cùng với bản mã tương ứng được tạo ra với khóa bí mật.</li> <li>+ Bản mã được lựa chọn bởi kẻ tấn công, cùng với bản mã tương ứng được giải mã với khóa bí mật.</li> </ul>

Một kiểu tấn công khác là tấn công Brute-Force bằng cách thử tất cả khóa có thể. Nếu không gian khóa là rất lớn, kiểu tấn công này rất khó để thực hiện. Do đó, kẻ tấn công phải dựa trên việc phân tích bản mã, thường áp dụng các thử nghiệm thông kê. Để sử dụng phương pháp này, kẻ tấn công phải có một vài ý tưởng chung về kiểu bản rõ đang được che dấu, như là bản Tiếng Anh hay Tiếng Pháp, file EXE, ...

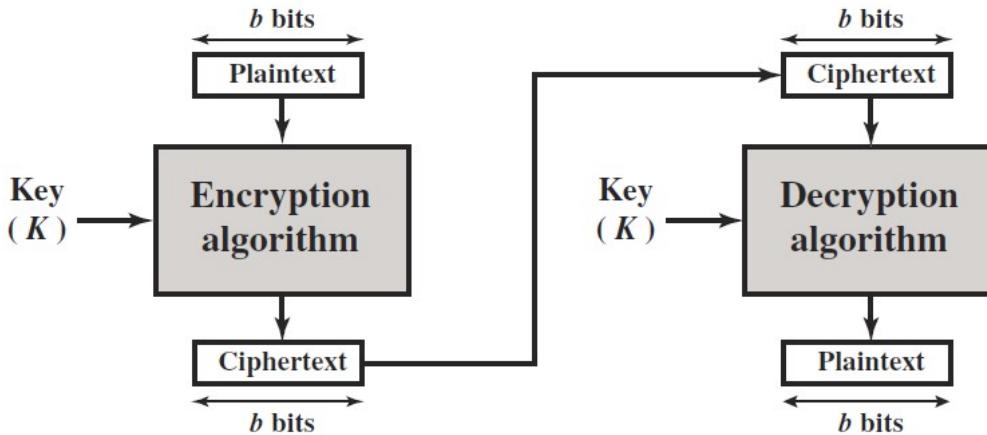
## 2.2 Mật mã khối và tiêu chuẩn mật mã hóa dữ liệu DES

### 2.2.1 Cấu trúc mật mã khối

Hiện nay, rất nhiều các thuật toán mật mã hóa khối đối xứng được sử dụng dựa trên cấu trúc mật mã khối Feistel. Do đó, trong phần này chúng tôi giới thiệu cấu trúc chung của mật mã khối và cấu trúc của mật mã khối Feistel.

### 2.2.1.1. Cấu trúc chung của mật mã khối

Mật mã khối là một kiểu mật mã trong đó bản rõ được xử lý theo khối và được sử dụng để tạo ra khối bản mã có chiều dài bằng chiều dài bản rõ. Thông thường, kích thước khối được sử dụng là 64 hoặc 128 bit. Cấu trúc bộ mật mã khối được mô tả như trong hình 2.3.



Hình 2.3: Cấu trúc mật mã khối

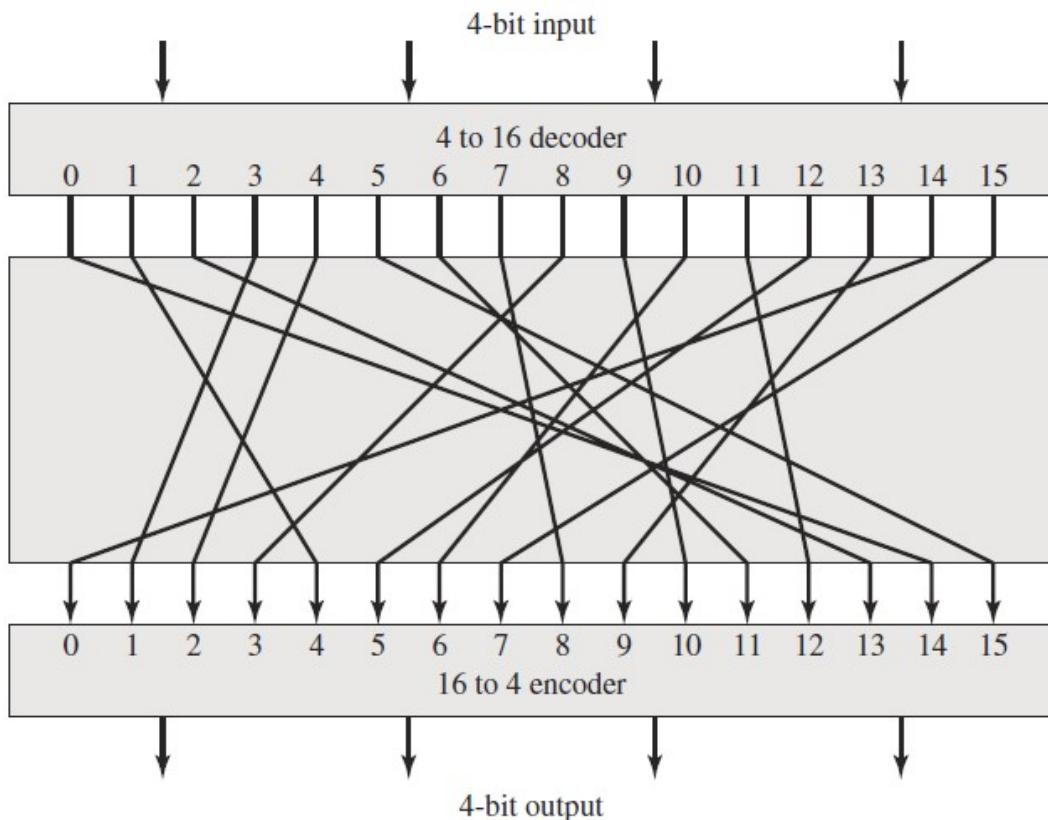
Mật mã khối hoạt động trên khối bản rõ  $n$  bit để tạo ra khối bản mã  $n$  bit. Có  $2^n$  khối bản rõ khác nhau có thể và, để việc mật mã hóa đó là biến đổi thuận nghịch (nghĩa là có thể giải mật mã), mỗi khối bản rõ phải tương ứng với một khối bản mã duy nhất. Sự biến đổi đó được gọi là biến đổi thuận nghịch, hoặc không phải một chiều. Các ví dụ dưới đây minh chứng các biến đổi một chiều và không phải một chiều cho trường hợp  $n=2$ .

Bảng 2.2: Các kiểu ánh xạ

Ánh xạ thuận nghịch		Ánh xạ một chiều	
Bản rõ	Bản mã	Bản rõ	Bản mã
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

Trong trường hợp ánh xạ một chiều, bản mã 01 có thể được tạo ra từ một trong hai khối bản rõ. Như vậy nếu phép ánh xạ thuận nghịch được sử dụng, số phép biến đổi khác nhau là  $2^n!$  (vì đối với bản rõ đầu tiên sẽ có  $2^n$  lựa chọn bản mã đầu ra, đối với bản rõ thứ 2 sẽ có  $2^n-1$  lựa chọn bản mã còn lại,...).

Hình 2.4 mô tả nguyên lý của mật mã thay thế chung đối với  $n = 4$ . Khối đầu vào 4 bit, là một trong 16 tổ hợp đầu vào, được ánh xạ bởi một mật mã thay thế để tạo ra một trong 16 tổ hợp đầu ra duy nhất. Nghĩa là, 4 bit bản rõ đầu vào sẽ được thay thế bởi 4 bit bản mã đầu ra tương ứng. Các ánh xạ mật mã hóa và giải mật mã có thể được định nghĩa bởi một bảng, như chỉ ra trong bảng 2.2. Đây là một dạng phổ biến nhất của mật mã khối và có thể được sử dụng để định nghĩa bất kỳ ánh xạ thuận nghịch nào giữa bản rõ và bản mã.



Hình 2.4: Nguyên lý của phép thay thế khối  $n$  bit đầu vào  $n$  bit đầu ra ( $n=4$ )

Bảng 2.3: Bảng mật mã hóa và giải mật mã cho mật mã khối thay thế của hình 2.4

Bản rõ	Bản mã
0000	1110
0001	0100
0010	1101
0011	0001

Bản mã	Bản rõ
0000	1110
0001	0011
0010	0100
0011	1000

0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

### 2.2.1.2 Cấu trúc mật mã khối Feistel

Cấu trúc mật mã khối Feistel do Horst Feistel đề xuất, là sự kết hợp của các phép thay thế và hoán vị. Trong mô hình mật mã Feistel, bản rõ sẽ được biến đổi qua một số vòng để cho ra bản mã cuối cùng. Mô hình mật mã khối Feistel được mô tả trong hình 2.5.

Các phép biến đổi trong cấu trúc mật mã Feistel được mô tả như sau:

$$P \xrightarrow{K_1} C_1 \xrightarrow{K_2} C_2 \xrightarrow{K_3} \dots \xrightarrow{K_{n-1}} C_n$$

Trong đó, P là bản rõ,  $C_i$  ( $i=1, 2, \dots, n$ ) là các bản mã.

Bản rõ và các bản mã được chia thành hai nửa trái và phải như sau:

$$P = (LE_0, RE_0)$$

$$C_i = (LE_i, RE_i) \quad i=1, 2, \dots, n$$

Qua mỗi vòng, quy tắc biến đổi các nửa trái nửa phải như sau:

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \oplus F(LE_{i-1}, K_i)$$

Trong đó, toán tử  $\oplus$  thể hiện phép XOR,  $K_i$  là khóa con cho vòng thứ  $i$ . Khóa con này được tạo ra từ khóa K ban đầu theo một thuật toán sinh khóa con sao cho mỗi khóa con là khác nhau và khác khóa K. F là một hàm mật mã hóa giống nhau ở tất cả các vòng. Hàm F thể hiện phép thay thế, còn việc tráo đổi các nửa trái và nửa phải thể hiện phép hoán vị.

Bản mã của hệ thống sẽ là bản mã đầu ra của vòng cuối cùng được hoán vị.

$$C = (RE_n, LE_n)$$

Quá trình giải mật mã được thực hiện ngược lại cũng với số vòng như ở phần mật mã hóa. Khi đó, đầu vào bộ giải mật mã sẽ là bản mã C với giá trị như sau:

$$LD_0 = RE_n$$

$$RD_0 = LE_n$$

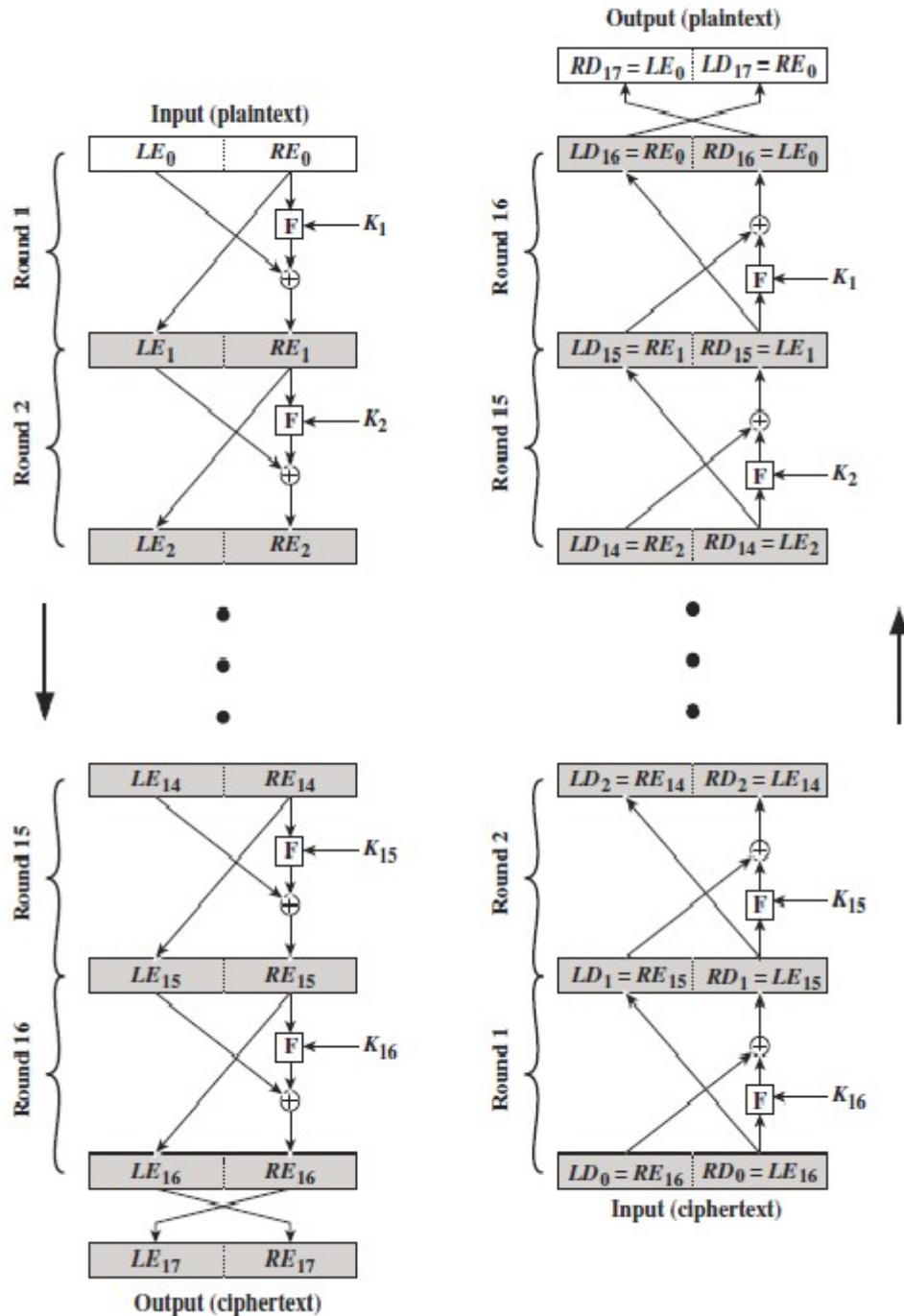
Qua các vòng các bản mã được giải như sau:

$$LD_i = RD_{i-1}$$

$$RD_i = LD_{i-1} \oplus F(RD_{i-1}, K_i)$$

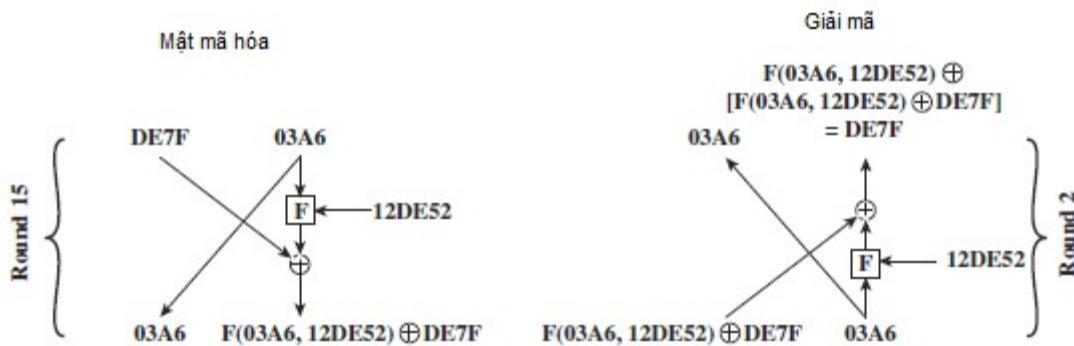
Sau vòng cuối cùng, bản rõ được giải ra với giá trị như sau:

$$P = (RD_n, LD_n)$$



Hình 2.5: Cấu trúc mật mã hóa và giải mật mã Feistel

Một ví dụ về việc mật mã hóa và giải mật mã theo Feistel như chỉ ra trong hình 2.6.



Hình 2.6: Ví dụ về mật mã hóa và giải mật mã Feistel

Việc hiện thực hóa chính xác hệ thống mật mã Feistel phụ thuộc vào việc lựa chọn các tham số và các đặc tính thiết kế dưới đây:

- + **Kích thước khối:** kích thước khối lớn có nghĩa là an toàn cao hơn (với giả thiết là tất cả các tham số khác là như nhau) nhưng tốc độ mật mã hóa/giải mật mã bị giảm đối với một thuật toán cho trước. Thông thường, kích thước khối 64 bit là kích thước phổ biến, được sử dụng trong thiết kế mật mã khối. Tuy nhiên, hệ thống mật mã mới AES sử dụng kích thước khối 128 bit.
- + **Kích thước khóa:** kích thước khóa lớn có nghĩa là an toàn cao hơn nhưng có thể làm giảm tốc độ mật mã hóa/giải mật mã. An toàn cao hơn có nghĩa là chống lại được các tấn công brute-force tốt hơn. Kích thước khóa 64 bit hoặc ít hơn 64 bit hiện nay đang được coi là không đủ, và khóa 128 bit đã trở thành một kích thước phổ biến.
- + **Số vòng:** bản chất của mật mã Feistel là một vòng mật mã đơn không đủ để cung cấp tính an toàn nhưng nhiều vòng mật mã sẽ làm tăng tính an toàn. Kích thước phổ biến là 16 vòng.
- + **Thuật toán tạo khóa con:** Tính phức tạp trong thuật toán này sẽ gây khó khăn cho kẻ tấn công.
- + **Hàm F:** tương tự như thuật toán tạo khóa con, hàm F càng phức tạp thì độ an toàn càng cao.

## 2.2.2 DES

Mật mã tiêu chuẩn DES (Data Encryption Standard) được đưa ra năm 1977 bởi cục tiêu chuẩn quốc gia, giờ là Viện tiêu chuẩn và kĩ thuật quốc gia (NIST) Hoa Kỳ. Thuật toán của mật mã này được gọi là DEA (Data Encryption Algorithm). Với DEA, dữ liệu được **mật mã hóa theo khối 64 bit sử dụng khóa 56 bit**. Thuật toán này biến đổi 64 bit đầu vào trong một chuỗi các bước thành 64 bit đầu ra. DES ngày càng trở thành thuật toán mật mã đối xứng phổ biến, đặc biệt trong các ứng dụng tài chính.

### 2.2.2.1 Cấu trúc DES

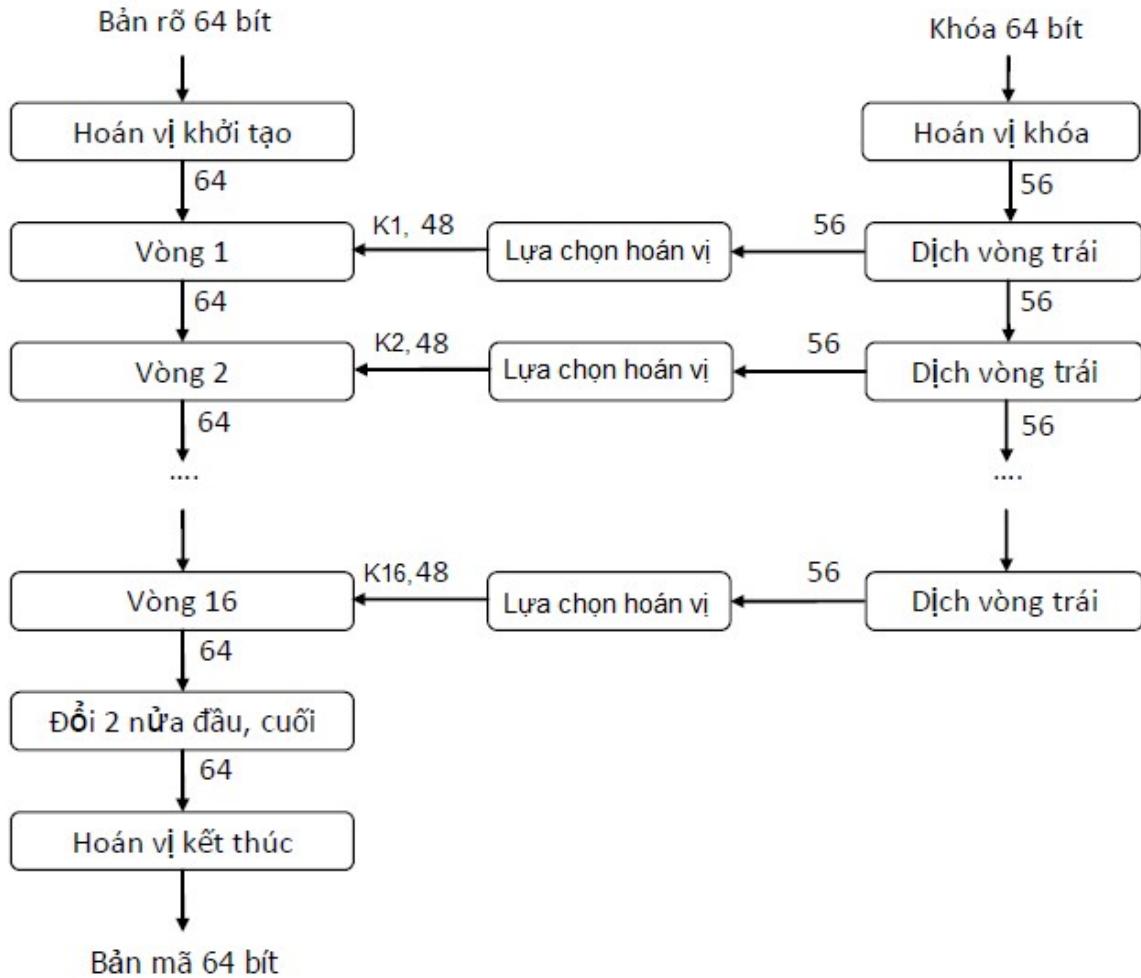
Mật mã DES có các đặc điểm sau:

- + Là mã thuộc mã Feistel có 16 vòng, ngoài ra DES có thêm một hoán vị khởi tạo trước khi bắt đầu vòng 1 và một hoán vị kết thúc sau vòng 16.
- + Kích thước khối là 64 bit.
- + Kích thước khóa là 56 bit
- + Mỗi vòng của DES dùng khóa con có kích thước 48 bit được trích ra từ khóa chính.

Cấu trúc mật mã hóa của mã DES được mô tả như hình 2.7.

Như chỉ ra ở nửa hình bên trái của hình 2.7, quá trình xử lý bǎn rõ diễn ra trong ba giai đoạn. Đầu tiên, bǎn rõ 64 bit được chuyển tới khói hoán vị khởi tạo để sắp xếp lại các bit và cho ra chuỗi bit đã được hoán vị. Tiếp theo đó là 16 vòng mật mã Feistel. Đầu ra của vòng cuối cùng (vòng 16) gồm 64 bit là một hàm của bǎn rõ đầu vào và khóa K. Sau đó, nửa trái và nửa phải của 64 bit này sẽ được tráo đổi cho nhau. Cuối cùng, các bit đã được tráo đổi đó được đưa qua bộ hoán vị kết thúc, đây là một hàm hoán vị nghịch đảo của hoán vị khởi tạo, và cho ra 64 bit bǎn mã.

Phần bên phải của hình 2.7 mô tả cách thức khóa 56 bit được sử dụng. Ban đầu, khóa 64 bit được chuyển qua bộ hoán vị khóa. Sau đó, đối với mỗi 16 vòng, khóa con K được tạo ra bằng cách kết hợp dịch vòng trái và hoán vị. Hàm hoán vị là giống nhau ở mỗi vòng, nhưng khóa con khác nhau được tạo ra bởi các dịch vòng trái được lặp lại ở các bit khóa.



Hình 2.7: Thuật toán mật mã DES

### 2.2.2.2 Hoán vị khởi tạo và hoán vị kết thúc

Giả sử bản rõ 64 bit được đánh số từ trái qua phải là 0, 1, 2, ..., 63 hay  $b_0 b_1 b_2 \dots b_{63}$ , khi đó hoán vị khởi tạo sẽ hoán đổi các bit theo quy tắc sau:

57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7
56	48	40	32	24	16	8	0
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6

$(b_0 b_1 b_2 \dots b_{62} b_{63}) \rightarrow (b_{57} b_{49} b_{41} \dots b_{14} b_6)$

Hoán vị kết thúc hoán đổi các bit theo quy tắc sau:

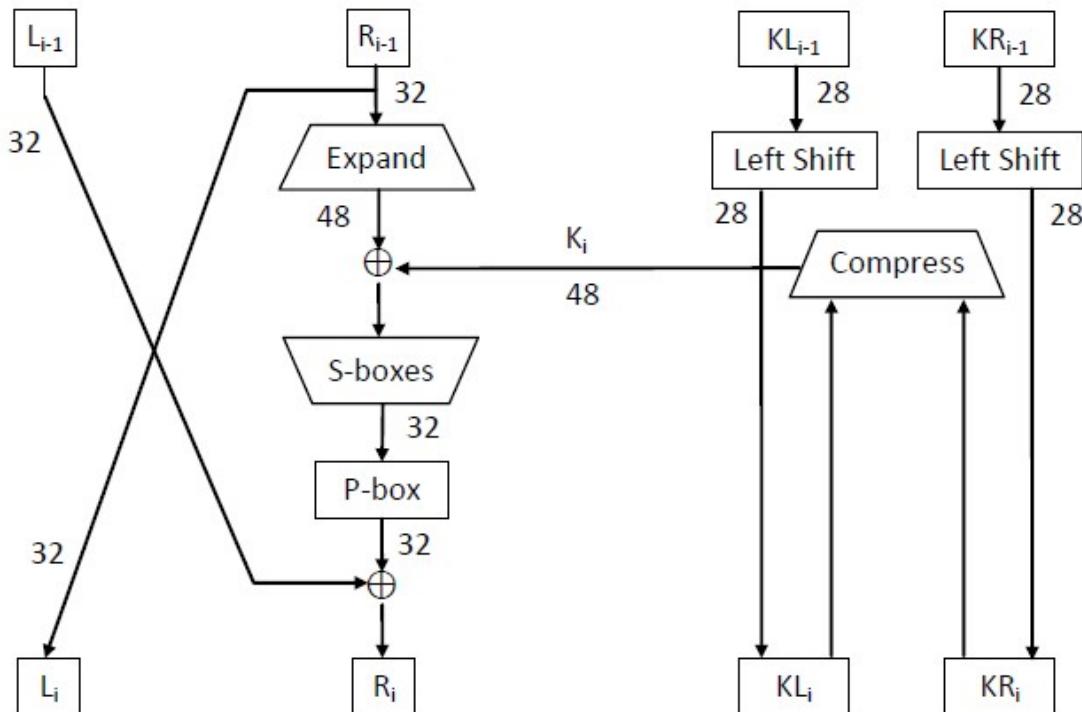
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25
32	0	40	8	48	16	56	24

Đối với các kiểu tấn công biết bản rõ hay bản rõ được lựa chọn, hoán vị khởi tạo và hoán vị kết thúc không có ý nghĩa bảo mật, sự tồn tại của hai hoán vị trên được cho là do yếu tố lịch sử để lại.

### 2.2.2.3 Các vòng mật mã của DES

Hình 2.8 dưới đây minh họa một vòng Feistel của DES. Trong đó, hàm F được mô tả như sau:

$$F(R_{i-1}, K_i) = P\text{-box}(S\text{-boxes}(Expand(R_{i-1}) \oplus K_i))$$



Hình 2.8: Cấu trúc một vòng mật mã DES

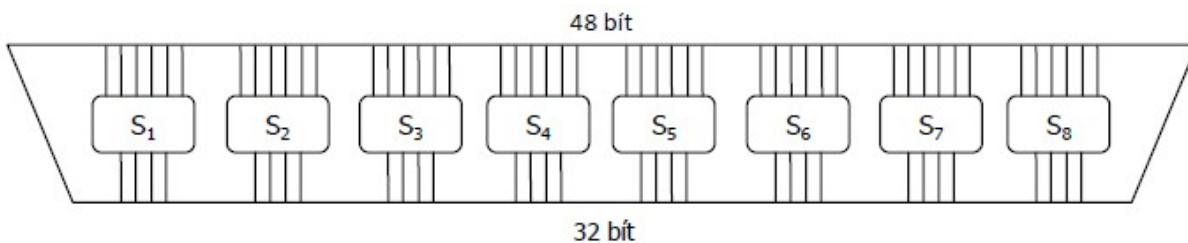
Trong đó, hàm  $Expand(R_{i-1})$  mở rộng  $R_{i-1}$  từ 32 bit thành 48 bit. Ngược lại, hàm  $S-boxes$  nén 48 bit thành 32 bit. Hàm  $P-box$  thực hiện hoán vị 32 bit. Cụ thể, hoạt động của các hàm này như sau:

- + Hàm  $Expand(R_{i-1})$ : đánh số các bit của  $R_{i-1}$  theo thứ tự từ trái qua phải là 0, 1, 2,..., 31. Hàm này sẽ thực hiện vừa hoán vị vừa mở rộng 32 bit thành 48 bit theo quy tắc sau:

31	0	1	2	3	4
3	4	5	6	7	8
7	8	9	10	11	12
11	12	13	14	15	16
15	16	17	18	19	20
19	20	21	22	23	24
23	24	25	26	27	28
27	28	29	30	31	0

↑  
48 bit  
↓

- + Hàm  $S-boxes$ : biến đổi 48 bit thành 32 bit. S-boxes được chia thành 8 hàm  $S$ -box con, mỗi hàm biến đổi 6 bit thành 4 bit.



Hàm S-box đầu tiên hoạt động như sau:

		$b_1 b_2 b_3 b_4$																
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
$b_0 b_5$		00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01		0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000	
10		0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000	
11		1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101	

- + Hàm P-box: thực hiện hoán vị 32 bit đầu vào theo quy tắc:

15	6	19	20	28	11	27	16
0	14	22	25	4	17	30	9
1	7	23	13	31	26	2	8
18	12	29	5	21	10	3	24

### 2.2.2.4 Thuật toán sinh khóa con của DES

Đầu tiên, khóa 64 bit được chuyển qua bộ hoán vị và nén thành khóa 56 bit theo quy tắc dưới đây:

56	48	40	32	24	16	8
0	57	49	41	33	25	17
9	1	58	50	42	34	26
18	10	2	59	51	43	35
62	54	46	38	30	22	14
6	61	53	45	37	29	21
13	5	60	52	44	36	28
20	12	4	27	19	11	3

56 bit

Sau đó, khóa 56 bit đó được chia thành hai nửa trái  $KL$  và phải  $KR$ , mỗi nửa có kích thước 28 bit. Tại vòng thứ  $i$  ( $i=1, \dots, 16$ ),  $KL_{i-1}$  và  $KR_{i-1}$  được dịch vòng trái  $r_i$  bit để tạo ra hai nửa  $KL_i$  và  $KR_i$  với  $r_i$  được xác định như sau:

$$r_i = \begin{cases} 1 & i \in \{1, 2, 6, 16\} \\ 2 & i \notin \{1, 2, 6, 16\} \end{cases}$$

Cuối cùng, khóa  $K_i$  của vòng thứ  $i$  được tạo ra bằng cách hoán vị và nén 56 bit  $KL_{i-1}$  và  $KR_{i-1}$  thành 48 bit theo quy tắc sau:

13	16	10	23	0	4	2	27
14	5	20	9	22	18	11	3
25	7	15	6	26	19	12	1
40	51	30	36	46	54	29	39
50	44	32	47	43	48	38	55
33	52	45	41	49	35	28	31

48 bit

### 2.2.2.5 Hiệu ứng lan truyền

Một tính chất quan trọng cần thiết của mọi thuật toán mã hóa là chỉ cần một thay đổi nhỏ trong bản rõ hay trong khóa sẽ dẫn đến thay đổi lớn trong bản mã. Cụ thể, chỉ cần thay đổi một bít trong bản rõ hay khóa thì dẫn đến sự thay đổi của nhiều bít bản mã. Tính chất này được gọi là hiệu ứng lan truyền. Nhờ có tính chất này mà kỉ phá mã không thể

giới hạn miền tìm kiếm của bản rõ hay của khóa (dù phá mã theo known-plaintext hay chosen-plaintext) nên phải thực hiện kiểu tấn công Brute Force (**vết cạn khóa**). DES là một phương pháp mã hóa có hiệu ứng lan truyền này. Để hiểu rõ hiệu ứng lan truyền trong DES, xét hai bản rõ sau:

P1: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

P2: 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Hai bản rõ trên được mã hóa bằng DES với khóa:

K: 0000001 1001011 0100100 1100010 0011100 0011000 0011100 0110010

Bảng 2.1a cho biết số bit khác nhau của hai bản mã P1 và P2 qua 16 vòng khác nhau của DES. Số bit khác nhau của hai bản rõ là 1 bit, nhưng đến vòng thứ 2 số bit khác nhau của hai bản mã đã là 21, và đến vòng cuối cùng thì số bit khác nhau là 34 bit. Cũng tương tự như vậy, ta xét bản rõ 64 bit sau:

P: 01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

Dùng hai khóa có số bit khác nhau là 1 sau đây để mã hóa bản rõ trên:

K1: 1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

K2: 0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

Bảng 2.1b chỉ ra số bit khác nhau của các bản mã qua 16 vòng của DES. Như chỉ ra trong bảng 2.1b, sau 16 vòng, số bit khác nhau của các bản mã do hai khóa khác nhau tạo ra là 35 bit.

Bảng 2.4: Ví dụ hiệu ứng lan truyền

Vòng thứ	Số bit khác nhau
0	1
1	6
2	21
3	35
4	39
5	34
6	32
7	31
8	29
9	42
10	44
11	32
12	30
13	30
14	26
15	29
16	34

a)

Vòng thứ	Số bit khác nhau
0	0
1	2
2	14
3	28
4	32
5	30
6	32
7	35
8	34
9	40
10	38
11	31
12	33
13	28
14	26
15	34
16	35

b)

### 2.2.3 Nguyên lý thiết kế mật mã khối

Mặc dù đã có rất nhiều nghiên cứu để nâng cao tính an toàn trong việc thiết kế mật mã khối, nhưng các nguyên lý cơ bản vẫn không thay đổi nhiều so với hoạt động của mã Feistel và DES từ những năm 1970. Trong mục này, ba khía cạnh cốt lõi trong việc thiết kế mật mã khối được trình bày, gồm số vòng tạo mã, thiết kế hàm F, và thuật toán tạo khóa.

#### Số vòng tạo mã:

Tính an toàn mật mã của mật mã Feistel xuất phát từ ba khía cạnh của việc thiết kế đó là số vòng tạo mã, hàm F, và thuật toán tạo khóa. Số vòng tạo mã càng lớn thì càng gây khó khăn cho kẻ tấn công, thậm chí trong cả trường hợp hàm F tương đối yếu. Nhìn chung, số vòng tạo mã nên được chọn sao cho độ khó tấn công phức tạp hơn kiểu tấn công tìm khóa brute-force. Tiêu chí này vẫn được sử dụng trong việc thiết kế DES. Đối với mật mã DES sử dụng 16 vòng mã hóa, tấn công mật mã yêu cầu 255.1 hành động, trong khi kiểu tấn công brute force yêu cầu 255 hành động. Như vậy, nếu DES có 15 hoặc ít hơn 15 vòng tạo mã, số hành động yêu cầu để phá mã sẽ nhỏ hơn số hành động yêu cầu phá mã theo kiểu brute-force.

Tiêu chí này được quan tâm rất nhiều bởi vì nó có thể dễ dàng điều chỉnh tính an toàn của thuật toán và so sánh với các thuật toán khác.

#### Thiết kế hàm F

Thành phần quan trọng nhất của mật mã khối Feistel là hàm F. Hàm F có chức năng cung cấp sự hỗn loạn trong mật mã Feistel, do đó nó phải là khó có thể khôi phục lại phép thay thế được thực hiện bởi hàm F đó. Một tiêu chí rõ ràng nhất đối với hàm F đó là tính phi tuyến. Hàm F càng phi tuyến, thì càng gây khó khăn cho kẻ tấn công.

Có một số tiêu chí khác được xem xét khi thiết kế hàm F. Trong đó, các thuật toán phải có hiệu ứng lan truyền tốt. Điều này có nghĩa là, khi thay đổi 1 bit đầu vào thì sẽ làm thay đổi nhiều bit ở đầu ra. Tiêu chí lan truyền nghiêm ngặt (SAC) được phát biểu rằng bất kỳ bit đầu ra j nào của S-box sẽ thay đổi với xác suất bằng  $\frac{1}{2}$  khi bất kỳ một bit đầu vào i nào bị thay đổi với mọi i và j. Mặc dù SAC chỉ mô tả với S-box, tiêu chí tương tự cũng được áp dụng với hàm F. Một tiêu chí khác đó là tiêu chí độc lập bit (BIC) phát biểu rằng các bit đầu ra j và k sẽ thay đổi một cách độc lập khi bất kỳ một bit đầu vào i nào bị thay đổi với mọi i, j, và k.

**Thuật toán tạo khóa:**

Với bất kỳ mật mã khối Feistel nào, một khóa chính sẽ được sử dụng để tạo ra một khóa con cho mỗi vòng tạo mã. Thông thường, các khóa con sẽ được lựa chọn để hạn chế tối đa được việc suy diễn ra các khóa con và việc khôi phục khóa chính. Thuật toán tạo khóa nên đảm bảo được hai tiêu chí SAC và BIC.

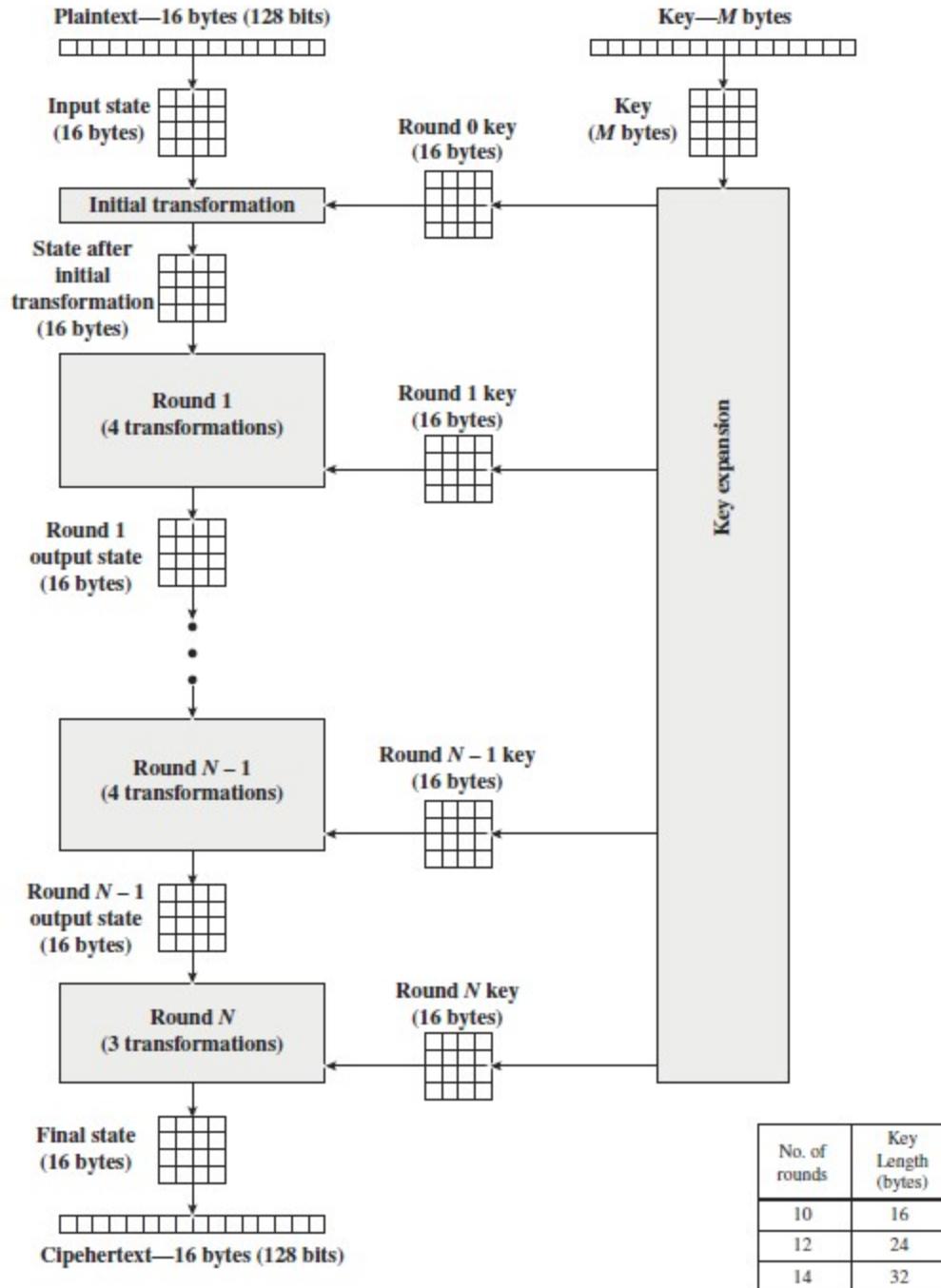
### 2.3 Tiêu chuẩn mật mã hóa tiên tiến AES

Vào những năm 1990, nhận thấy nguy cơ của mật mã hóa DES là kích thước khóa ngắn, có thể bị phá mã trong tương lai gần, Cục tiêu chuẩn quốc gia Hoa Kỳ đã kêu gọi xây dựng một phương pháp mật mã hóa mới. Cuối cùng một thuật toán có tên là Rijndael được chọn và đổi tên thành Advanced Encryption Standard hay AES được công bố bởi NIST, Hoa Kỳ vào năm 2001. Giống như DES, mật mã hóa AES là một mật mã khối đối xứng gồm nhiều vòng. Khác với DES, mã hóa AES không phải là một mã hóa Feistel.

#### 2.3.1 Cấu trúc AES

Hình 2.9 đưa ra cấu trúc chung của quá trình mật mã hóa AES. Kích thước khối bẩn rõ được sử dụng là 128 bit, hay 16 byte. Độ dài khóa có thể là 16, 24, hoặc 32 byte (128, 192, hoặc 256 bit). Thuật toán được sử dụng như là AES-128, AES-192, hay AES-256, phụ thuộc vào độ dài khóa.

Đầu vào của các thuật toán mật mã hóa và giải mật mã là khối 128 bit. Khối này được sắp xếp thành ma trận vuông có kích thước  $4 \times 4$  byte, được sửa đổi tạo mỗi giai đoạn mật mã hóa hoặc giải mật mã. Sau giai đoạn cuối cùng, đầu ra cũng sẽ là ma trận vuông có kích thước  $4 \times 4$  byte. Tương tự như vậy, khóa  $M$  byte cũng được sắp xếp thành ma trận vuông, sau đó được đưa tới bộ mở rộng khóa để tạo thành mảng các từ khóa.



Hình 2.9: Cấu trúc AES

Hình 2.10 mô tả việc mở rộng khóa 128 bit. Mỗi từ khóa gồm 4 byte, và toàn bộ mảng khóa là 44 từ cho khóa 128 bit. Chú ý rằng thứ tự theo byte trong ma trận được sắp xếp theo cột. Ví dụ, bốn byte đầu của đầu vào bản rõ 128 bit nằm ở cột thứ nhất của ma trận, bốn byte tiếp theo nằm ở cột thứ 2,... Tương tự, bốn byte đầu tiên của khóa được mở rộng, từ khóa, nằm ở cột đầu tiên của ma trận w.

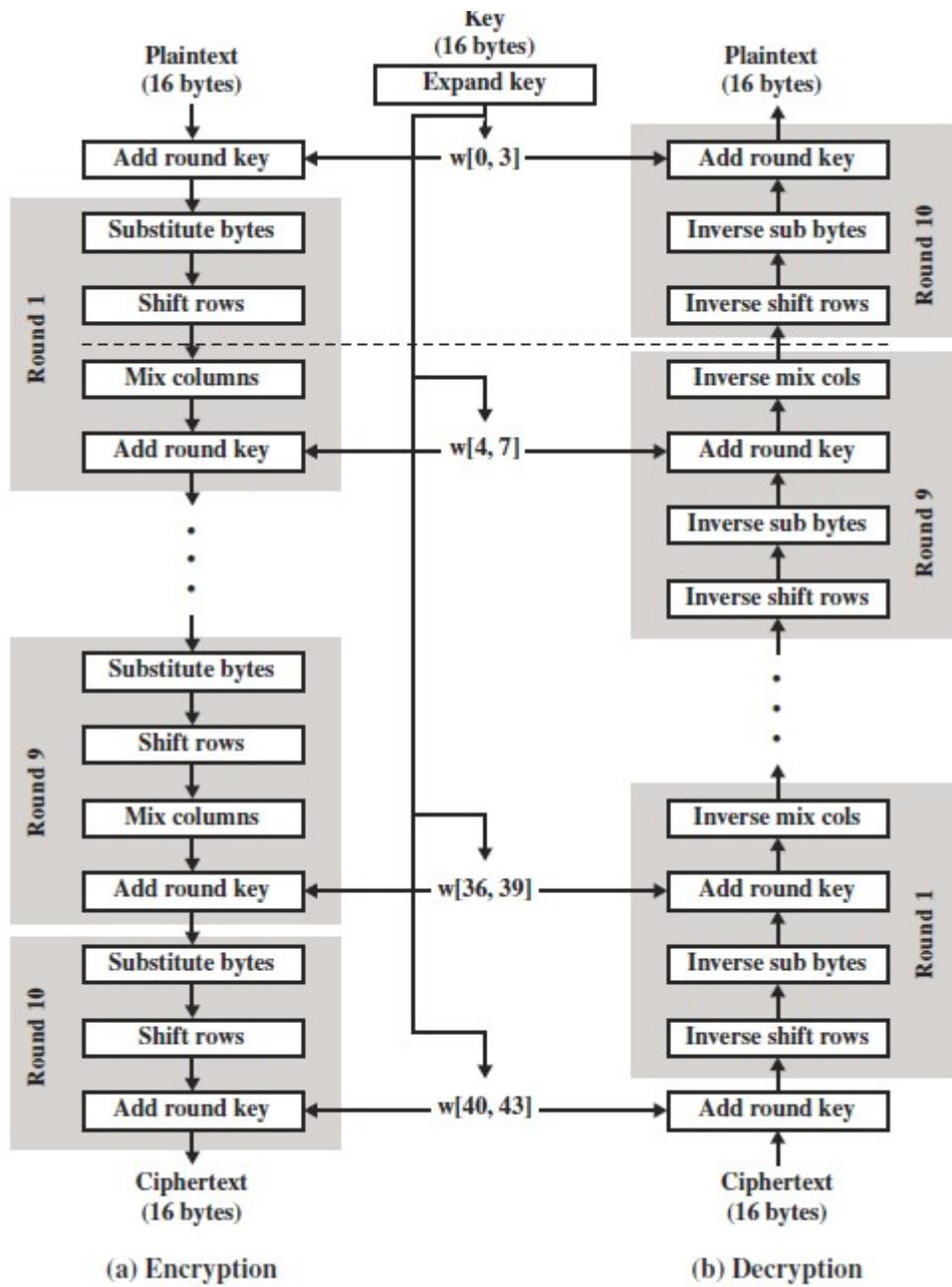
Hệ mật mã bao gồm N vòng, trong đó số vòng phụ thuộc vào độ dài khóa: 10 vòng cho khóa 16 byte, 12 vòng cho khóa 24 byte, và 14 vòng cho khóa 32 byte. N-1 vòng đầu bao gồm bốn hàm biến đổi: SubBytes, ShiftRows, MixColumns, và AddRoundKey. Vòng cuối cùng chỉ bao gồm 3 phép biến đổi, và có một phép biến đổi khởi tạo (AddRoundKey) trước vòng đầu tiên, có thể coi đó là vòng số 0. Mỗi phép biến đổi lấy 1 hoặc nhiều ma trận  $4 \times 4$  làm đầu vào và tạo ra đầu ra cũng là ma trận  $4 \times 4$ . Như chỉ ra trong hình 2.9, đầu ra mỗi vòng là một ma trận  $4 \times 4$ , với đầu ra của vòng cuối cùng sẽ là bản mã.



Hình 2.10: Khóa và khóa được mở rộng

Hàm mở rộng khóa tạo ra  $N+1$  khóa cho các vòng, mỗi khóa là một ma trận  $4 \times 4$ . Khóa mỗi vòng là một trong những đầu vào của biến đổi AddRoundKey của mỗi vòng.

Hình 2.11 đưa ra sơ đồ mật mã AES một cách chi tiết hơn, chỉ rõ thứ tự các phép biến đổi trong mỗi vòng và chỉ ra hàm giải mã tương ứng.



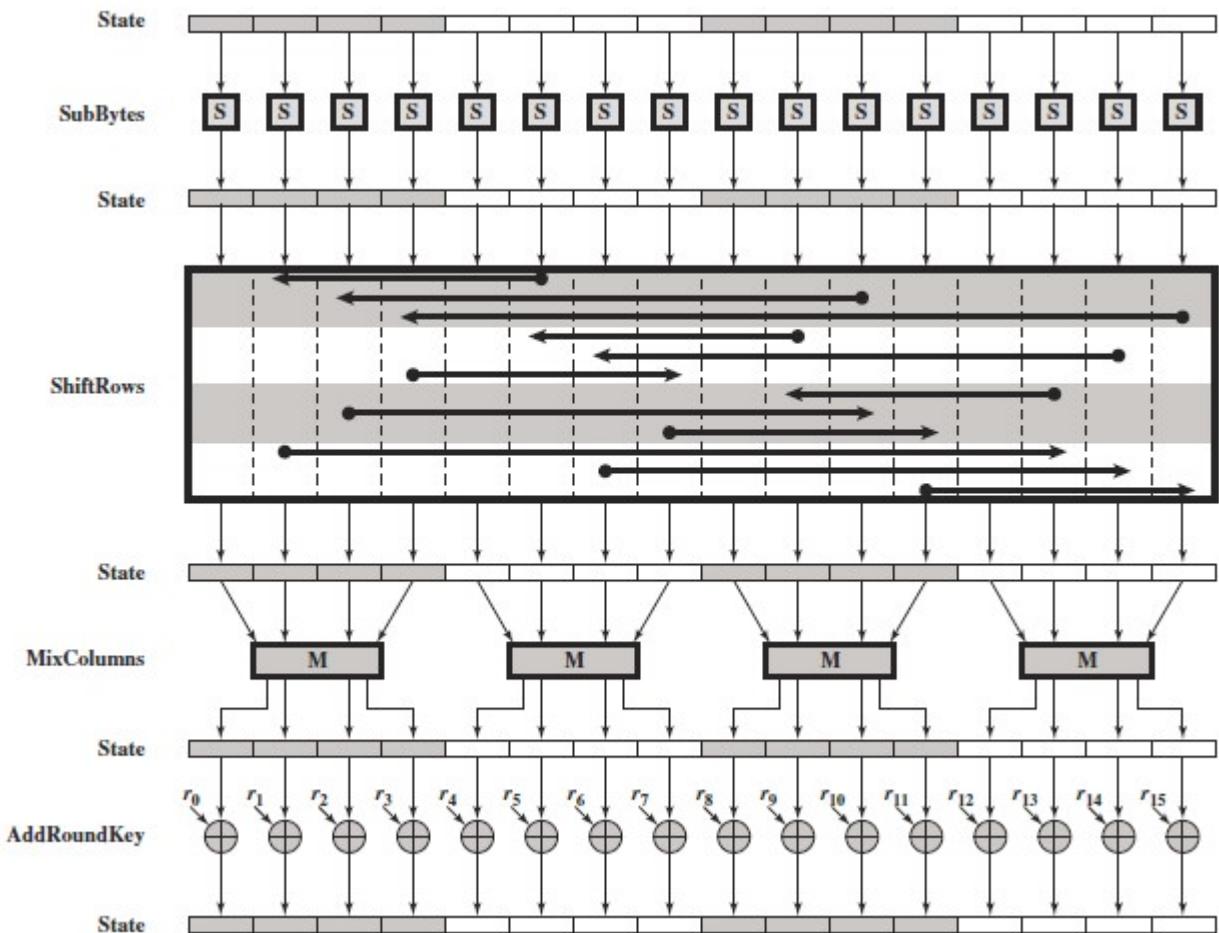
Hình 2.11: Sơ đồ mật mã và giải mật mã AES

Sơ đồ mật mã gồm 10 vòng, mỗi vòng mật mã AES được thực hiện như trong hình 2.12. Trong mỗi vòng mật mã, một phép hoán vị và ba phép thay thế được sử dụng:

- + **Substitute bytes:** sử dụng S-box để thực hiện thay thế các byte của khối đầu vào
  - + **ShiftRows:** đây là phép hoán vị đơn giản
  - + **MixColumns:** tại khối này phép thay thế khác được sử dụng

- + AddRoundKey: khối này thực hiện phép XOR của của khối mật mã đầu vào và một phần khóa mở rộng.

Các hàm biến đổi này có thể dễ dàng được khôi phục. Đối với các hàm SubBytes, ShiftRows, và MixColumns, một hàm nghịch đảo được sử dụng để giải mật mã. Đối với hàm AddRoundKey, giải mật mã có thể thực hiện bằng cách thực hiện phép XOR giữa từ mã đó với chính khóa sử dụng, do  $A \oplus B \oplus B = A$ .



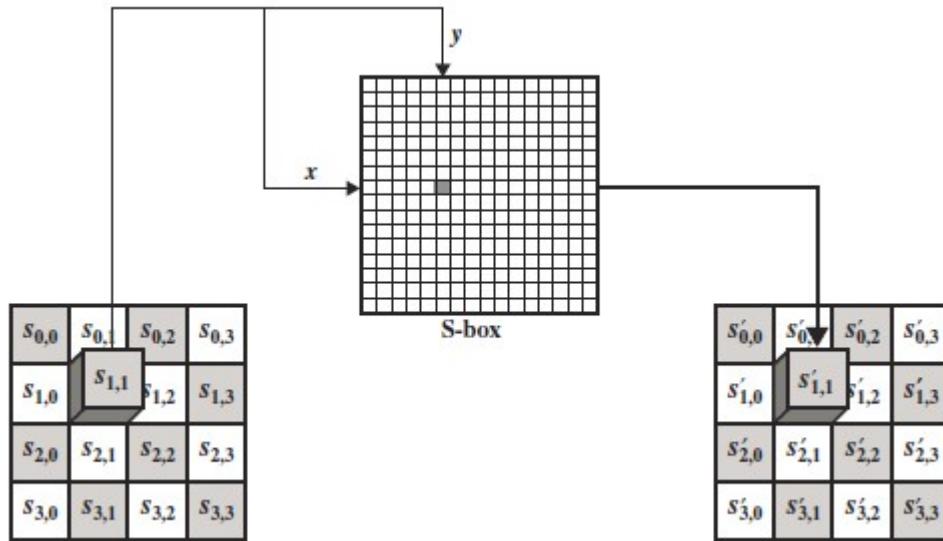
Hình 2.12: Vòng mật mã AES

### 2.3.2 Các hàm biến đổi AES

#### 2.3.2.1 Hàm SubBytes

Hàm biến đổi này chỉ đơn giản là phép thay thế, như trong hình 2.13. AES định nghĩa ma trận byte có kích thước  $16 \times 16$ , được gọi là S-box. S-box chứa toàn bộ 256 giá trị có thể của các byte. Mỗi byte đầu vào của khối S-box được ánh xạ thành một byte mới theo cách sau: 4 bit bên trái của byte đầu vào được xem như là giá trị hàng và 4 bit bên

phải được xem như là giá trị cột. Các giá trị cột và hàng đó được coi là các chỉ số cột và hàng trong S-box để lựa chọn giá trị đầu ra 8 bit duy nhất. Ví dụ, giá trị theo mã hexa {95} tương ứng với hàng 9 và cột 5 của S-box, chứa giá trị {2A}. Như vậy, giá trị {95} được ánh xạ thành giá trị {2A}.



Hình 2.13: Hàm SubBytes

Ma trận S-box và S-box ngược được chỉ ra ở hình 2.14 và hình 2.15.

		$y$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$x$	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Hình 2.14: S-box cho mật mã hóa

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Hình 2.15: S-box cho giải mã

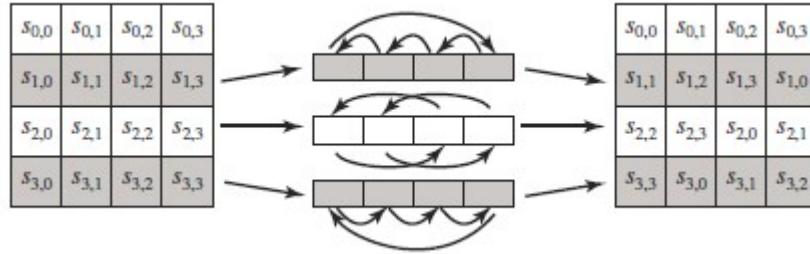
Dưới đây là ví dụ của hàm biến đổi SubBytes (hình 2.16).

EA	04	65	85	→	87	F2	4D	97
83	45	5D	96		EC	6E	4C	90
5C	33	98	B0		4A	C3	46	E7
F0	2D	AD	C5		8C	D8	95	A6

Hình 2.16: Ví dụ về biến đổi của hàm SubBytes

### 2.3.2.2 Hàm ShiftRows

Hàm ShiftRows thực hiện biến đổi dịch vòng các hàng của ma trận đầu vào, như chỉ ra trong hình 2.17.



Hình 2.17: Thực hiện dịch hàng của hàm ShiftRows

Như chỉ ra trong hình 2.17, hàng đầu tiên của ma trận đầu vào không bị dịch, trong khi hàng thứ 2, các byte được dịch vòng trái 1 byte. Đối với hàng thứ 3, dịch vòng trái 2 byte được thực hiện, và với hàng cuối cùng, dịch vòng trái 3 byte được thực hiện. Dưới đây là một ví dụ của hàm ShiftRows.

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

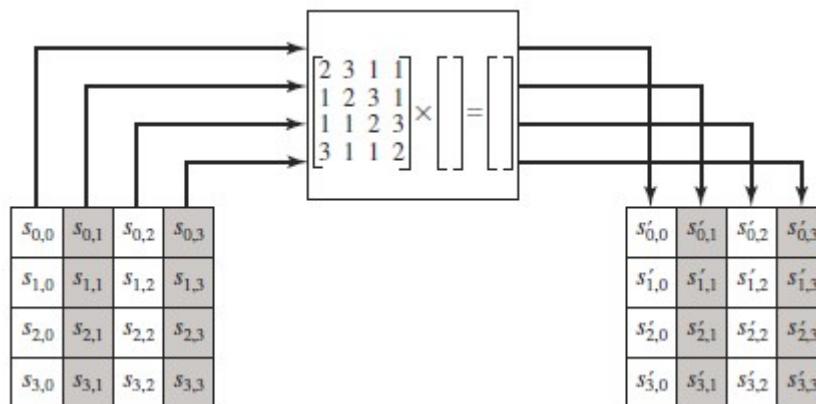
87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Hình 2.18: Ví dụ dịch vòng của hàm ShiftRows

Hàm ShiftRows ngược, để thực hiện giải mật mã được gọi là InvShiftRows, thực hiện dịch vòng theo hướng ngược lại cho mỗi ba hàng cuối.

### 2.3.2.3 Hàm MixColumns

Hàm MixColumns thực hiện biến đổi trộn các cột của ma trận đầu vào, được thực hiện trên mỗi cột một cách riêng biệt, như chỉ ra trong hình 2.19.



Hình 2.19: Hàm MixColumns

Mỗi byte của một cột được ánh xạ thành một giá trị mới là một hàm của bốn byte trong cột đó. Phép biến đổi này có thể được định nghĩa bởi phép nhân ma trận sau:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Như vậy mỗi cột đầu ra được biến đổi theo cột đầu vào tương ứng như sau:

$$\begin{aligned} s'_{0,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j}) \\ s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j}) \end{aligned}$$

Dưới đây là ví dụ về hoạt động biến đổi của hàm MixColumns.

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

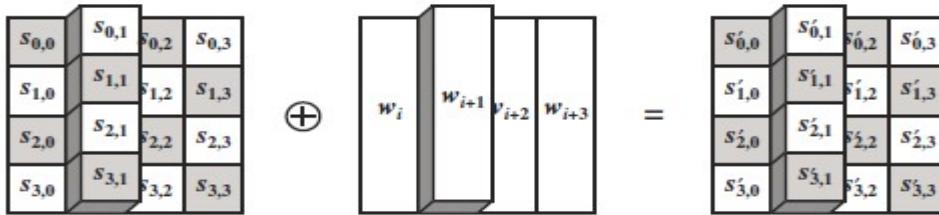
47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

Hàm biến đổi ngược của hàm MixColumns là hàm InvMixColumns, được định nghĩa bởi phép nhân ma trận dưới đây:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}$$

### 2.3.2.4 Hàm AddRoundKey

Hàm AddRoundKey thực hiện phép XOR giữa 128 bit đầu vào và 128 bit khóa của vòng đó, được mô tả trong hình 2.20.



Hình 2.20: Hàm AddRoundKey

Dưới đây là ví dụ hoạt động của hàm AddRoundKey. Ma trận đầu tiên là ma trận đầu vào, ma trận tiếp theo là ma trận khóa, và ma trận cuối cùng là ma trận đầu ra.

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

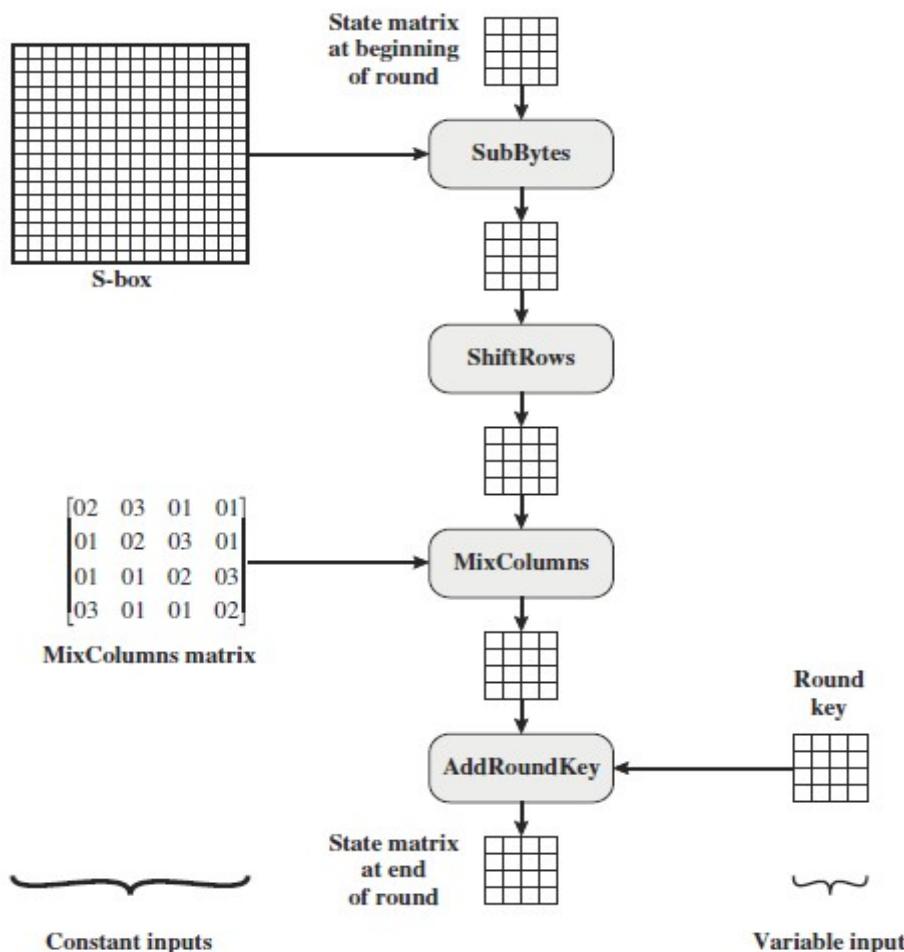
⊕

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

Như vậy, mỗi vòng mật mã của mật mã AES được thực hiện như trong hình 2.21 dưới đây.



Hình 2.21: Các đầu vào cho một vòng mật mã của AES

### 2.3.3 Tạo khóa AES

Thuật toán mở rộng khóa AES với đầu vào gồm 4 từ hay 16 byte (hình 2.11) tạo ra đầu ra gồm một mảng tuyến tính 44 từ (176 byte). Đầu ra của bộ mở rộng khóa đủ để cung cấp khóa cho các vòng mật mã.

Khóa được đưa ra bốn từ đầu tiên của khóa mở rộng. Phần còn lại của khóa mở rộng được điền vào 4 từ tại mỗi thời điểm. Mỗi từ được thêm vào  $w[i]$  phụ thuộc vào từ ngay trước nó,  $w[i-1]$ , và  $w[i-4]$ . Thuật toán tạo khóa mở rộng như sau:

```

KeyExpansion (byte key[16], word w[44])
{word temp
For (i = 0; i < 4; i++) w[i] = (key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]);
For (i = 4; i < 44; i++)
{temp = w[i-1];
If (i mod 4 = 0) temp = SubWord (RotWord (temp)) ⊕ Rcon[i / 4];
w[i] = w[i-4] ⊕ temp
}
}

```

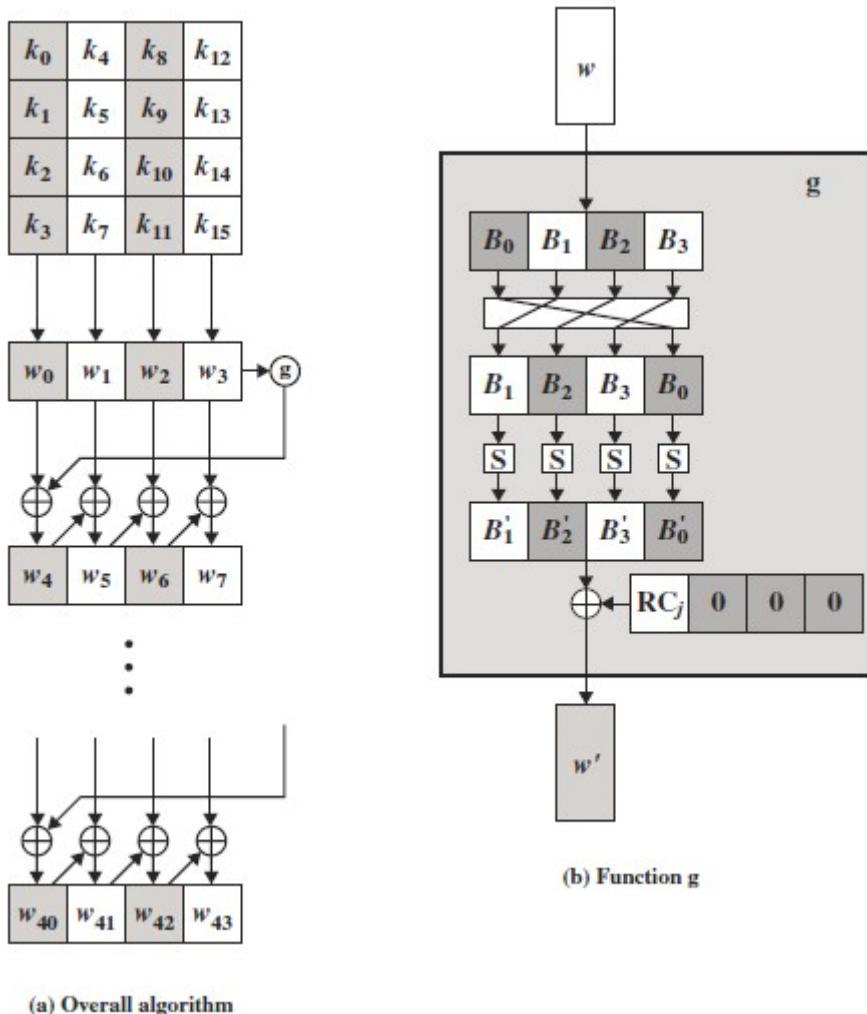
Hình 2.21 dưới đây mô tả thuật toán tạo khóa AES, và hàm phức g. Hàm phức g gồm các hàm con dưới đây:

- + RotWord: dịch vòng trái một byte. Giả sử từ đầu vào có 4 byte là [B0, B1, B2, B3] thì kết quả của RotWord là [B1, B2, B3, B0].
- + SubWord: thay thế mỗi byte trong từ đầu vào bằng cách tra cứu bảng S-box trong thao tác SubBytes.2
- + Kết quả của việc thực hiện hai hàm trên sẽ được thực hiện XOR với hàm Rcon[j].

Hàm Rcon[j] cho mỗi vòng là khác nhau, và được định nghĩa như sau

$Rcon[j] = (RC[j], 0, 0, 0)$ , với  $RC[j]$  được xác định như bảng dưới đây:

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



Hình 2.22: Thuật toán tạo khóa AES

### 2.3.4 Thực hiện AES

Như đã đề cập ở trên, việc giải mật mã AES không hoàn toàn giống như quá trình mật mã hóa. Nghĩa là, trình tự các phép biến đổi cho việc giải mật mã khác với trình tự các phép biến đổi cho việc mật mã hóa, mặc dù các sơ đồ tạo khóa cho quá trình mật mã hóa và giải mật mã là giống nhau. Đây là một nhược điểm mà do đó hai module phần mềm và phần firmwave riêng biệt cần được yêu cầu cho các ứng dụng mật mã hóa và giải mật mã. Tuy nhiên, có một phiên bản tương đương của thuật toán giải mật mã mà có cấu trúc giống như cấu trúc của thuật toán mật mã hóa. Phiên bản tương đương này có trình tự các phép biến đổi như nhau cho cả quá trình mật mã hóa và giải mật mã. Để đạt được điều này, sơ đồ tạo khóa cần có sự thay đổi.

Như chỉ ra trong hình 2.11, mỗi vòng mật mã hóa có các hàm SubBytes, ShiftRows, MixColumns, AddRoundKey. Mỗi vòng giải mật mã chuẩn có các hàm InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns. Do đó, hai giai đoạn đầu tiên của vòng giải mật mã cần được thay đổi, và hai giai đoạn sau cũng cần được thay đổi.

**Thay đổi hàm InvShiftRows và InvSubBytes:** hàm InvShiftRows ảnh hưởng đến thứ tự byte trong ma trận nhưng không làm thay đổi nội dung các byte và không phụ thuộc vào nội dung để thực hiện phép biến đổi. Hàm InvSubBytes ảnh hưởng đến nội dung của các byte trong ma trận nhưng không làm thay đổi thứ tự byte và không phụ thuộc vào thứ tự byte để thực hiện phép biến đổi. Do đó, hai phép toán này có thể được thay thế lẫn nhau như sau:

$$\text{InvShiftRows} [\text{InvSubBytes}(S_i)] = \text{InvSubBytes} [\text{InvShiftRows}(S_i)]$$

**Thay đổi hàm AddRoundKey, InvMixColumns:** các phép biến đổi AddRoundKey, InvMixColumns không làm thay đổi thứ tự các byte trong ma trận. Nếu coi khóa là một chuỗi các từ, thì cả hai hàm AddRoundKey và InvMixColumns hoạt động trên một cột của ma trận tại một thời điểm. Hai hoạt động đó là tuyến tính đối với đầu vào là cột. Nghĩa là, với ma trận cho trước  $S_i$  và khóa cho trước  $w_j$ , thì:

$$\text{InvMixColumns}(S_i \oplus w_j) = [\text{InvMixColumns}(S_i)] \oplus [\text{InvMixColumns}(w_j)]$$

Ví dụ, giải sử cột đầu tiên của ma trận  $S_i$  có trình tự là  $(y_0, y_1, y_2, y_3)$  và cột đầu tiên của khóa  $w_j$  là  $(k_0, k_1, k_2, k_3)$ , ta có:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \oplus k_0 \\ y_1 \oplus k_1 \\ y_2 \oplus k_2 \\ y_3 \oplus k_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \oplus \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

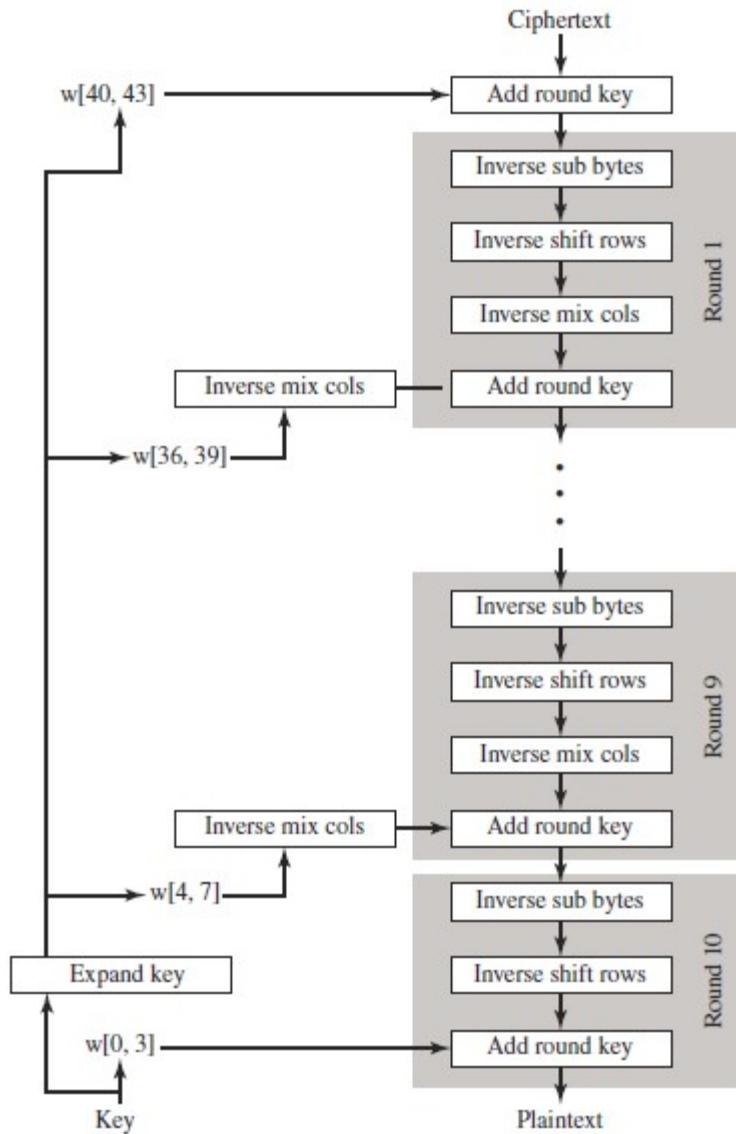
Cột đầu tiên được tính như sau:

$$\begin{aligned} & [\{0E\} \cdot (y_0 \oplus k_0)] \oplus [\{0B\} \cdot (y_1 \oplus k_1)] \oplus [\{0D\} \cdot (y_2 \oplus k_2)] \oplus [\{09\} \cdot (y_3 \oplus k_3)] \\ &= [\{0E\} \cdot y_0] \oplus [\{0B\} \cdot y_1] \oplus [\{0D\} \cdot y_2] \oplus [\{09\} \cdot y_3] \oplus \\ & \quad [\{0E\} \cdot k_0] \oplus [\{0B\} \cdot k_1] \oplus [\{0D\} \cdot k_2] \oplus [\{09\} \cdot k_3] \end{aligned}$$

Từ đó, ta thấy rằng có thể tráo đổi vị trí các hàm AddRoundKey và InvMixColumns, miễn là đầu tiên phải thực hiện hàm InvMixColumns đối với khóa mỗi vòng. Chú ý rằng, không cần phải thực hiện hàm InvMixColumns với khóa cho đầu vào của hàm biến đổi AddRoundKey đầu tiên (vòng 0) hay hàm biến đổi AddRoundKey của

vòng cuối cùng (vòng 10). Bởi vì hai phép biến đổi AddRoundKey này không cần tráo đổi với hàm InvMixColumns để tạo ta thuật toán giải mật mã tương đương.

Hình 2.23 mô tả thuật toán giải mật mã tương đương.



Hình 2.23: Mật mã nghịch đảo tương đương

#### Các khía cạnh thực thi:

Để thực hiện hiệu quả mật mã hóa AES trên các bộ xử lý 8 bit của các card thông minh hiện tại, và trên các bộ xử lý 32 bit của các máy tính, một số yêu cầu sau phải được đáp ứng.

Đối với bộ xử lý 8 bit: AES cần được thực hiện một cách rất hiệu quả trên các bộ xử lý 8 bit. Hàm AddRoundKey là một phép XOR theo byte. Hàm ShiftRows chỉ là phép

dịch byte đơn giản. Hàm SubBytes thực hiện tại mức byte và chỉ yêu cầu bảng 256 byte. Hàm MixColumns thực hiện nhân ma trận.

Đối với bộ xử lý 32 bit: Việc thực hiện được mô tả trong các mục trước chỉ sử dụng cho các phép tính toán 8 bit. Đối với bộ xử lý 32 bit, việc thực hiện hiệu quả hơn có thể đạt được nếu các phép tính toán được định nghĩa trên các từ 32 bit. Khi đó, các phép biến đổi có thể được mô tả như sau, trong đó giả thiết ma trận đầu vào gồm các phần tử  $a_{i,j}$  và ma trận khóa gồm các phần tử  $k_{i,j}$ .

SubBytes	$b_{i,j} = S[a_{i,j}]$
ShiftRows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
MixColumns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRoundKey	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

Từ đó, có thể dùng một phương trình dưới đây để mô tả toàn bộ các hàm trên.

$$\begin{aligned}
 \begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} &= \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix} \\
 &= \left( \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot S[a_{0,j}] \right) \oplus \left( \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[a_{1,j-1}] \right) \oplus \left( \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot S[a_{2,j-2}] \right) \\
 &\quad \oplus \left( \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot S[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}
 \end{aligned}$$

Phép nhân ma trận trong phương trình trên được xem là sự kết hợp tuyến tính của các vector. Bốn bảng 256 từ (1024 byte) được định nghĩa như sau:

$$T_0[x] = \begin{pmatrix} [02] \\ [01] \\ [01] \\ [03] \end{pmatrix} \cdot S[x] \quad T_1[x] = \begin{pmatrix} [03] \\ [02] \\ [01] \\ [01] \end{pmatrix} \cdot S[x] \quad T_2[x] = \begin{pmatrix} [01] \\ [03] \\ [02] \\ [01] \end{pmatrix} \cdot S[x] \quad T_3[x] = \begin{pmatrix} [01] \\ [01] \\ [03] \\ [02] \end{pmatrix} \cdot S[x]$$

Đầu vào mỗi bảng là 1 byte và cho ra một vector cột (một từ 32 bit) là một hàm của S-box.

Như vậy, hoạt động của mỗi vòng trên mỗi cột của ma trận đầu vào được xác định như sau:

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = T_0[s_{0,j}] \oplus T_1[s_{1,j-1}] \oplus T_2[s_{2,j-2}] \oplus T_3[s_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

Nhận thấy rằng, việc thực hiện AES chỉ dựa vào việc tra 4 bảng và thực hiện các 4 phép XOR trên mỗi cột cho mỗi vòng, cộng với 4 Kbyte để lưu trữ bảng.

## 2.4 Các ứng dụng của mật mã khối

### 2.4.1 Mật mã hóa nhiều lần

Đối mặt với nguy cơ tấn công brute-force của mật mã DES, đã có rất nhiều hướng quan tâm trong việc tìm ra phương pháp thay thế. Một cách tiếp cận là thiết kế một thuật toán mới hoàn toàn, AES là một ví dụ. Một cách khác có thể bảo tồn được các đầu tư trước đó về phần cứng cũng như phần mềm đó là sử dụng mật mã hóa nhiều lần với DES và sử dụng nhiều khóa.

#### DES hai lần (Double DES)

Dạng đơn giản nhất của mật mã hóa nhiều lần có hai giai đoạn mật mã hóa và sử dụng hai khóa như hình 2.24. Với bản rõ P và hai khóa mật mã K1 và K2, bản mã C được tạo ra như sau:

$$C = E(K_2, E(K_1, P))$$

Giải mật mã yêu cầu biết các khóa đó và được thực hiện ngược lại như sau:

$$P = D(K_1, D(K_2, C))$$

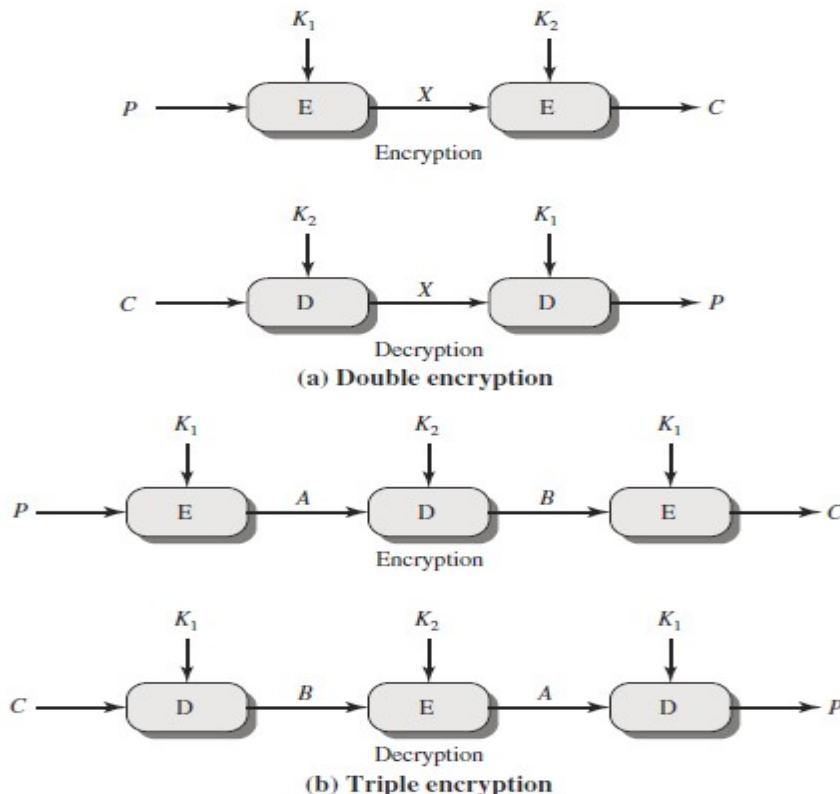
Điều này giống như là Double DES dùng một khóa có kích thước là 112 byte, chỉ có một hạn chế là tốc độ chậm hơn DES vì phải dùng DES hai lần. Tuy nhiên người ta đã tìm được một phương pháp tấn công Double DES có tên gọi là gấp-nhau-ở-giữa (meet-in-the-middle). Đây là một phương pháp tấn công chosen-plaintext.

Phương pháp tấn công này có thể được hiểu như sau. Như chỉ ra trong hình 2.24 a,

$$X = E(K_1, P)$$

$$X = D(K_2, C)$$

Nếu biết trước một cặp  $(P, C)$ , kẻ tấn công sẽ thực hiện như sau. Đầu tiên mật mã hóa  $P$  với  $2^{56}$  giá trị có thể của khóa  $K_1$ . Lưu các kết quả đó trong 1 bảng. Tiếp theo, giải mã  $C$  sử dụng  $2^{56}$  giá trị có thể của khóa  $K_2$ . Với mỗi kết quả giải mã, so sánh với kết quả trong bảng. Nếu có giá trị nào giống nhau, thử lại hai khóa được tìm ra đó với cặp  $(P, C)$  đã biết mới. Nếu hai khóa đó cho ra bản mã đúng, hai khóa đó được coi là hai khóa đúng.



Hình 2.24: Mật mã hóa nhiều lần

- Mật mã hóa hai lần
- Mật mã hóa ba lần

### Triple DES

Để khắc phục được sự tấn công của Double DES, DES ba lần với ba khóa khác nhau được lựa chọn. Chiều dài khóa là 168 bít sẽ gây phức tạp hơn nhiều cho việc phá mã bằng phương pháp tấn công gấp-nhau-ở-giữa. Triple DES được mô tả như hình 2.24b. Bản mã được xác định như sau:

$$C = E(K_3, D(K_2, E(K_1, P)))$$

Trong thực tế người ta chỉ dùng Triple DES với hai khóa K1, K2 mà vẫn đảm bảo độ an toàn cần thiết. Khi đó, bản mã và bản rõ được xác định bởi.

$$C = E(K_1, D(K_2, E(K_1, P)))$$

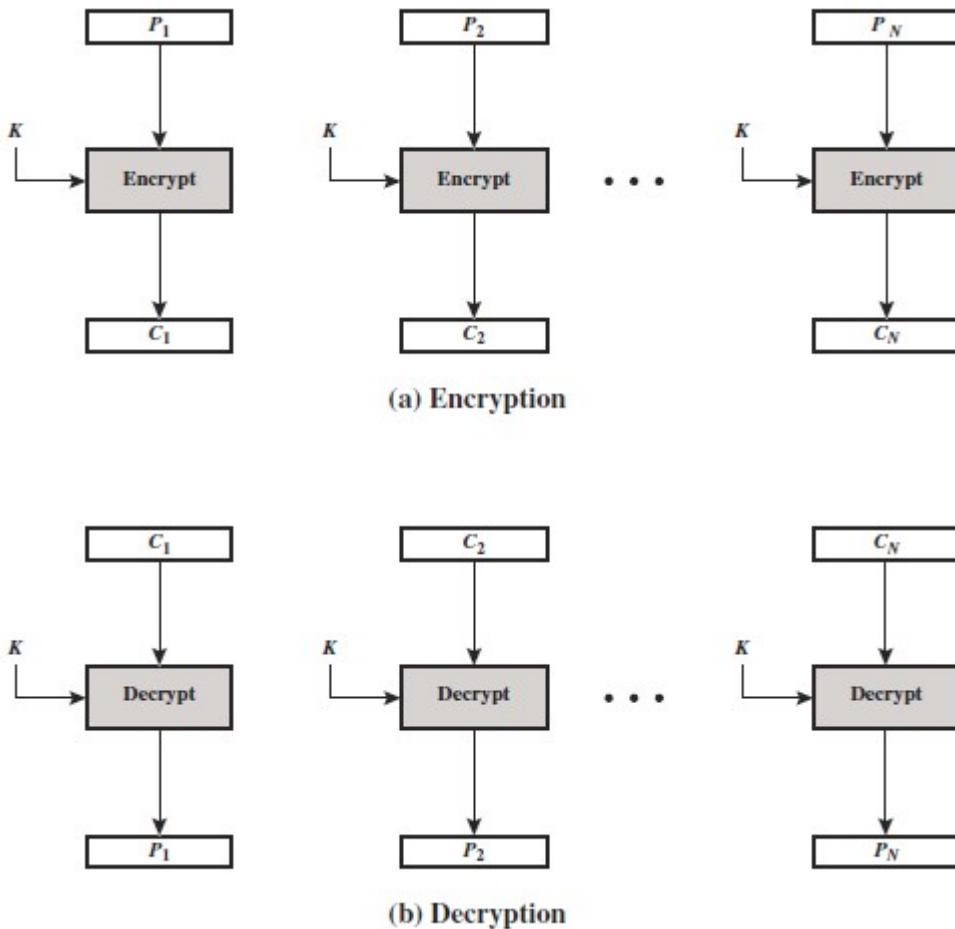
$$P = D(K_1, E(K_2, D(K_1, C)))$$

#### 2.4.2 Các chế độ và ứng dụng mật mã khối

Mật mã khối được áp dụng để mật mã hóa một khối dữ liệu có kích thước xác định. Để mật mã hóa một bản tin dài, bản tin được chia ra thành nhiều khối và áp dụng mật mã khối cho từng khối một. Có nhiều mô hình ứng dụng mật mã khối như ECB, CBC, CTR, OFB và CFB.

##### Chế độ ECB (Electronic Codebook)

Trong mô hình ECB, mỗi khối được mật mã hóa một cách riêng rẽ, dùng chung một khóa K. Mô hình mật mã hóa và giải mật mã của ECB được mô tả trong hình 2.25.

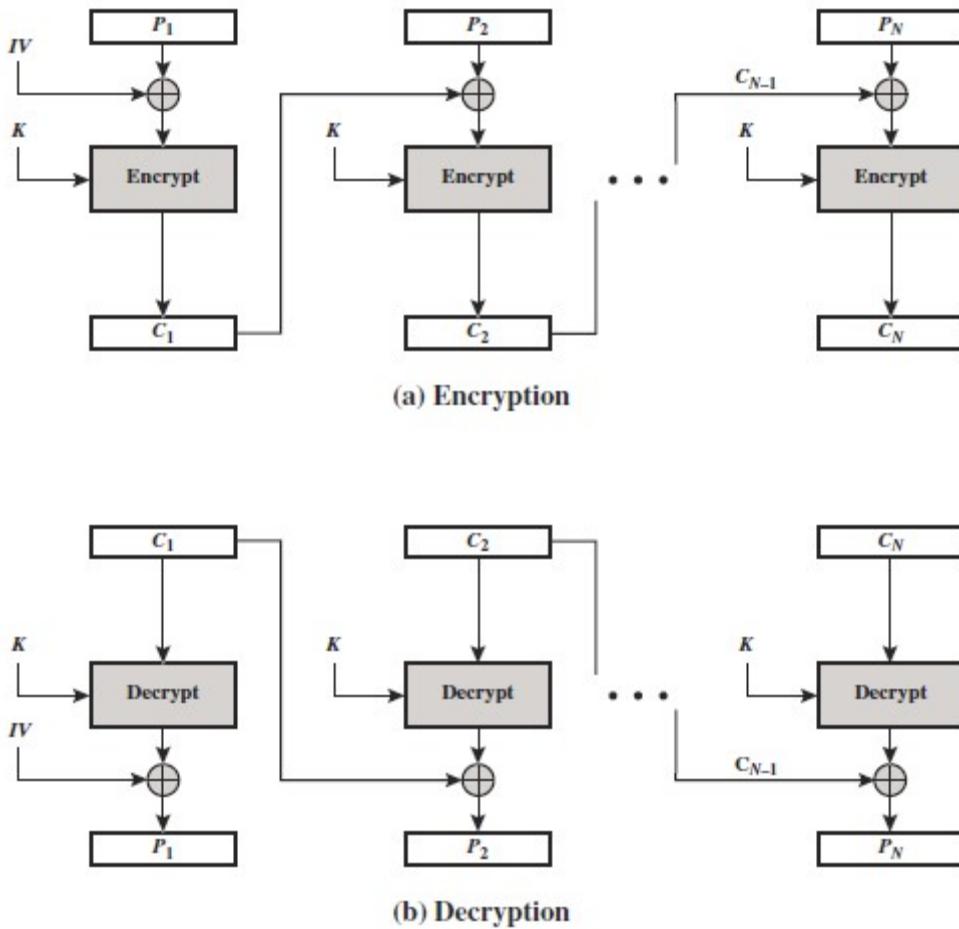


Hình 2.25: Mô hình mật mã hóa và giải mật mã của ECB

Đặc điểm nổi bật của mô hình ECB là nếu như bản rõ giống nhau thì bản mã giống nhau. Với mô hình đơn giản của ECB, kẻ tấn công có thể dựa vào một số đặc tính thống kê của dữ liệu để tiến hành phá mã. ECB chỉ thích hợp để mật mã hóa cho các bản tin có kích thước ngắn, ví dụ như mật mã hóa cho khóa. Do đó, nếu muốn truyền khóa DES hoặc AES một cách an toàn, ECB là một phương pháp thích hợp để truyền khóa đó. Với các bản tin dài hơn, ECB có thể không an toàn.

### Chế độ CBC (Cipher Block Chaining)

Để khắc phục được nhược điểm của chế độ ECB, chế độ CBC được đưa ra với mục đích là tạo ra các khối bản mã khác nhau khi đầu vào là các bản rõ giống nhau. Hình 2.26 mô tả mô hình CBC đó.



Hình 2.26: Mô hình mật mã hóa và giải mật mã CBC

Trong sơ đồ mật mã hóa CBC, đầu vào của mỗi khối mật mã hóa là kết quả của phép XOR giữa bản rõ hiện tại và bản mã trước đó, sử dụng cùng một khóa cho mỗi khối.

$$C_i = E(P_i \oplus C_{i-1}, K) \text{ với } i = 1, 2, \dots, N$$

Do đó để mã hóa khối đầu tiên, người ta dùng một khối dữ liệu giả được gọi là vector khởi tạo (initialization vector – IV) và được chọn ngẫu nhiên:

$$C_1 = E(P_1 \oplus IV, K)$$

Quá trình giải mật mã được thực hiện ngược lại.

$$P_1 = D(C_1, K) \oplus IV$$

$$P_i = D(C_i, K) \oplus C_{i-1} \text{ với } i = 1, 2, \dots, N$$

Bên mật mã hóa và bên giải mật mã phải dùng chung vector khởi tạo IV. Vector khởi tạo không cần giữ bí mật nên thường được gắn vào trước bản mã trước khi truyền

thông điệp ( $IVC_1C_2\dots C_N$ ). Có thể thấy rằng nội dung của bản mã  $C_i$  không chỉ phụ thuộc vào bản rõ  $P_i$  mà còn phụ thuộc vào tất cả các bản rõ đứng trước và IV. Do đó nếu có hai bản rõ giống nhau thì hai bản mã sẽ không giống nhau (do IV ngẫu nhiên). Điều này khắc phục được hạn chế của mô hình ECB, từ bản mã kẻ tấn công không thể phát hiện ra những đặc tính thống kê của dữ liệu. Ngược lại, đối với việc giải mật mã, bản rõ  $P_i$  không chỉ phụ thuộc vào bản mã  $C_i$  mà còn phụ thuộc vào bản mã  $C_{i-1}$  đứng trước. Do đó nếu xảy lỗi trên đường truyền, chỉ cần một bít bị hỏng thì dẫn đến không thể giải mật mã được bản mã đó và bản mã tiếp theo sau.

### Chế độ CFB (Cipher Feedback)

Đối với AES, DES, hay bất cứ mật mã khối nào, việc mật mã hóa được thực hiện trên mỗi khối  $b$  bit. Trong trường hợp DES,  $b = 64$  bit và trong AES,  $b = 128$  bit. Tuy nhiên, có thể biến đổi từ mật mã khối thành mật mã dòng, sử dụng một trong ba chế độ gồm CFB, OFB và CTR. Mật mã dòng loại bỏ được các bit đệm vào bản tin để được số nguyên các khối. Nó có thể hoạt động theo thời gian thực. Do đó, nếu dòng ký tự đang được truyền đi, mỗi ký tự có thể được mật mã hóa và truyền đi ngay lập tức nhờ sử dụng mật mã dòng hướng ký tự.

Một đặc tính mong muốn của mật mã dòng là bản mã có cùng độ dài với bản rõ. Do đó, nếu các ký tự 8 bit được truyền đi, mỗi ký tự sẽ được mật mã hóa để tạo ra bản mã 8 bit đầu ra. Nếu nhiều hơn 8 bit được tạo ra, dung lượng truyền dẫn sẽ bị lãng phí.

Hình 2.27 mô tả sơ đồ CFB. Giả thiết đơn vị truyền dẫn là  $s$  bit, thường  $s = 8$ . Trong sơ đồ này, bản rõ được chia thành các đoạn  $s$  bit.

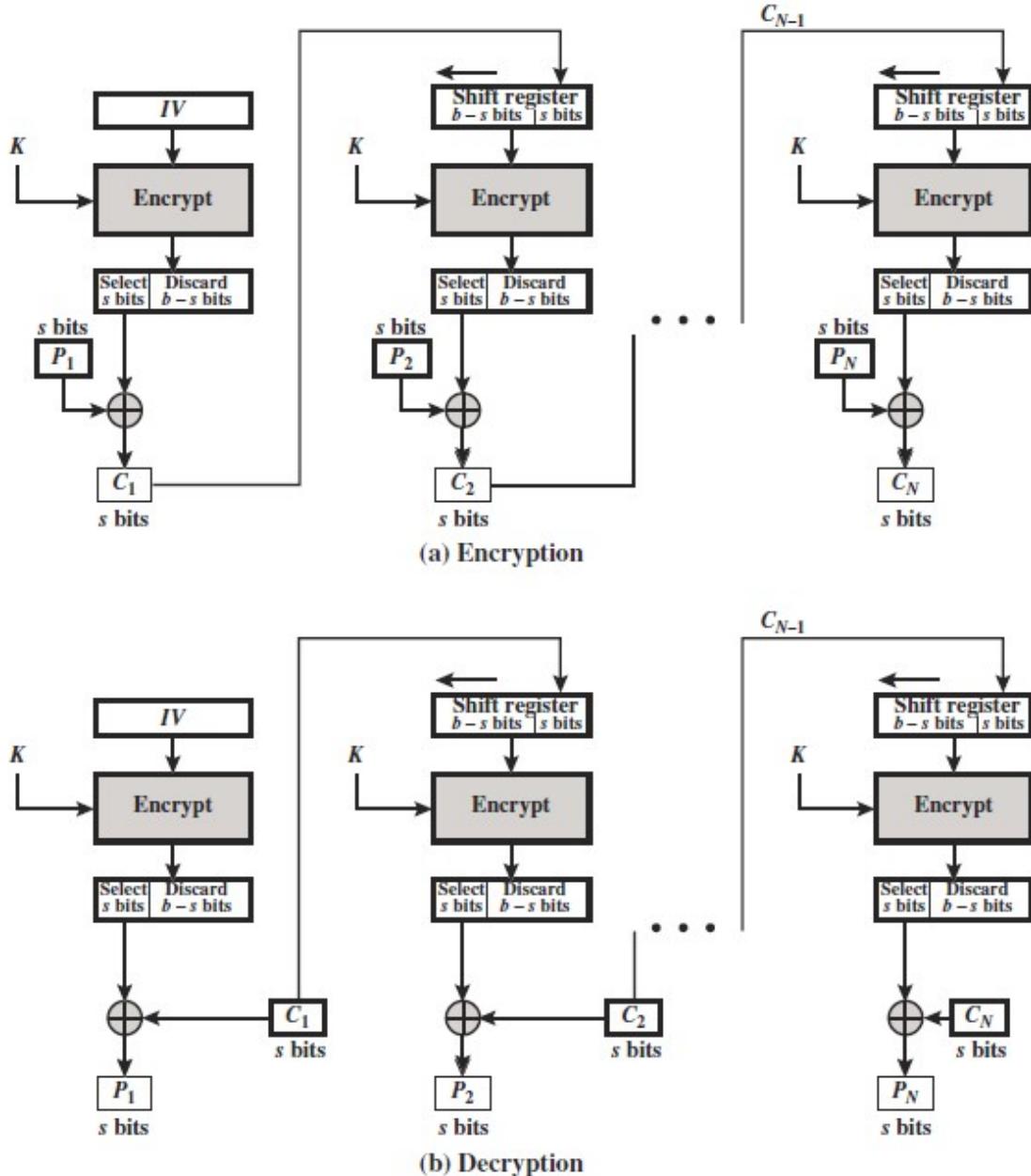
Trong quá trình mật mã hóa, đầu vào của khối chức năng mật mã hóa là thanh ghi dịch  $b$  bit được thiết lập khởi tạo bởi một vector khởi tạo (IV).  $S$  bit bên trái của khối mật mã hóa được thực hiện XOR với phần bản rõ đầu tiên  $P_1$  để tạo ra đoạn bản mã  $C_1$  đầu tiên. Nội dung của thanh ghi dịch được dịch trái  $s$  bit, và  $C_1$  được đặt vào  $s$  bit bên phải của thanh ghi dịch. Quá trình tiếp tục cho đến khi tất cả các đoạn bản rõ được mật mã hóa.

Đối với quá trình giải mật mã, sơ đồ thực hiện cũng tương tự, ngoại trừ đoạn bản mã nhận được được thực hiện XOR với đầu ra của khối mật mã hóa để tạo ra đoạn bản rõ. Chú ý rằng, ở đây khối mật mã hóa được sử dụng chứ không phải khối giải mật mã được sử dụng. Điều này được giải thích cụ thể như sau. Gọi  $MSB_s(X)$  là  $s$  bit có ý nghĩa nhất (các bit bên trái) của  $X$ . Ta có,

$$C_1 = P_1 \oplus \text{MSB}[E(K, IV)]$$

Do đó,

$$P_1 = C_1 \oplus \text{MSB}[E(K, IV)]$$



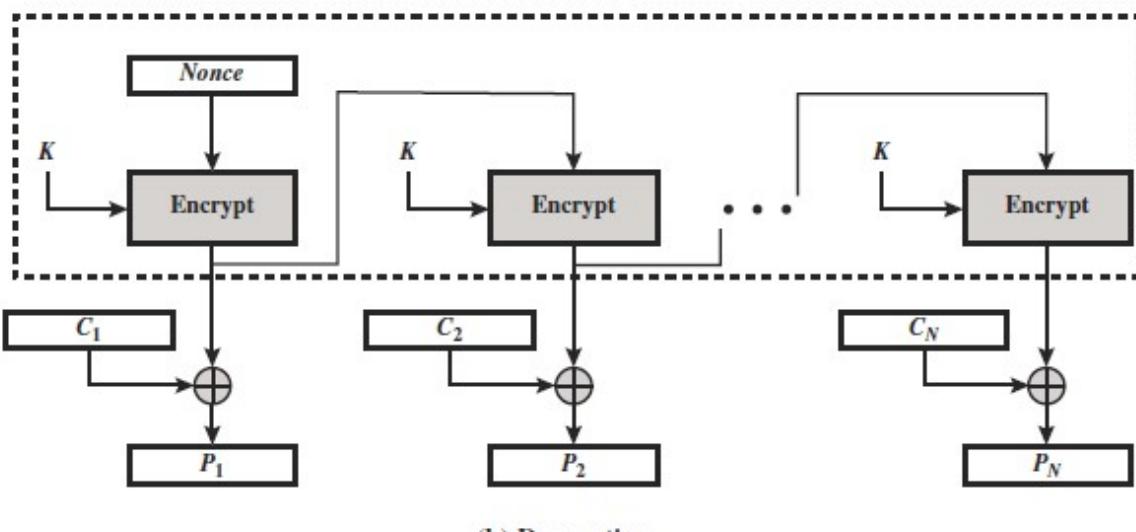
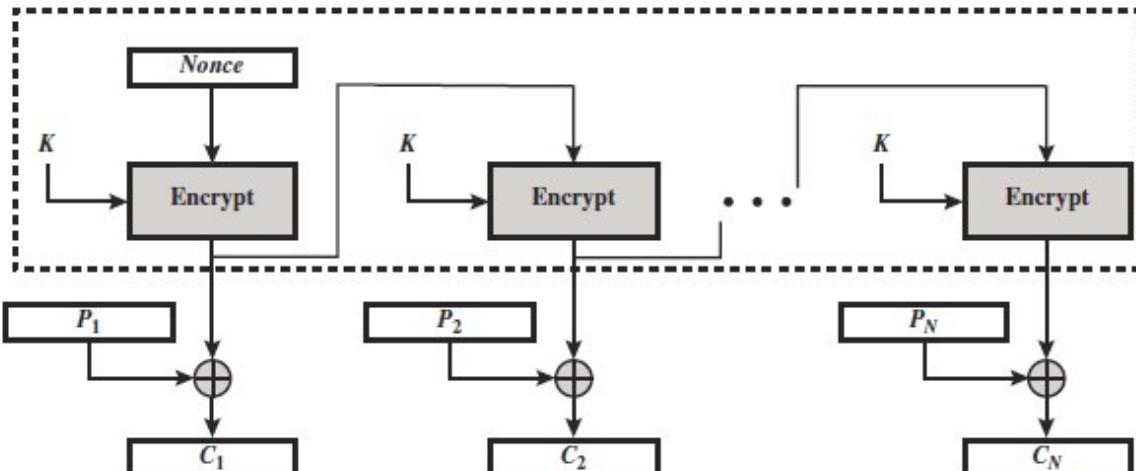
Hình 2.27: Chế độ CFB

### Chế độ OFB (Output Feedback)

Chế độ OFB tương tự như CFB. Đầu ra của khối mật mã hóa được đưa ngược lại thành đầu vào để mật mã hóa khôi bản rõ tiếp theo (hình 2.28). Trong chế độ OFB, đơn vị khôi bản rõ và bản mã có kích thước bất kỳ. Mật mã hóa OFB được mô tả như sau:

$$C_j = P_j \oplus E(K, O_{j-1})$$

Trong đó,  $O_{j-1} = E(K, O_{j-2})$ .



Hình 2.28: Chế độ OFB

Một ưu điểm của chế độ OFB là các lỗi bit trong quá trình truyền dẫn không bị truyền đi. Ví dụ, nếu lỗi bit xảy ra tại  $C_1$ , chỉ có giá trị bản rõ khôi phục  $P_1$  bị ảnh hưởng;

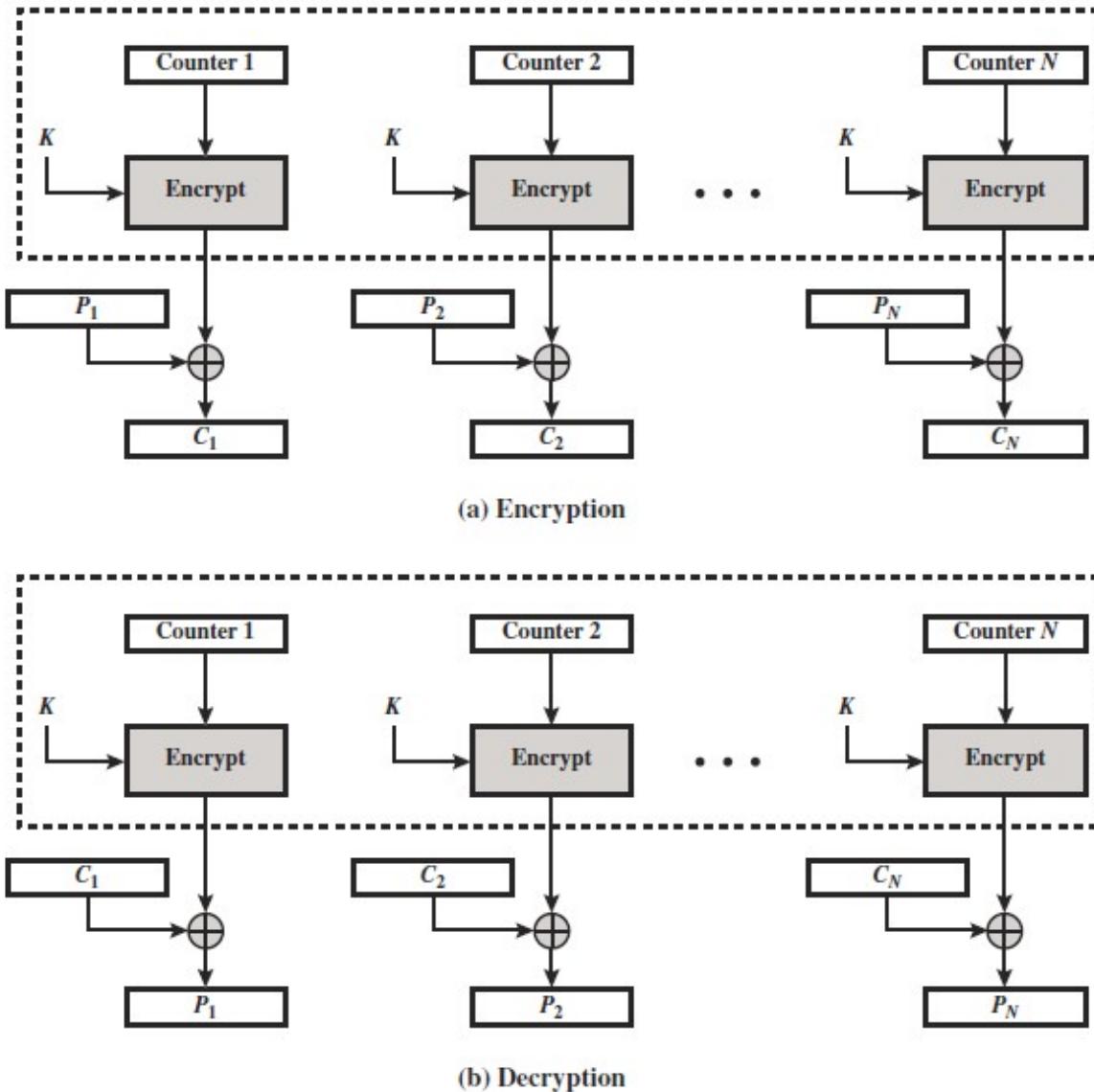
các đoạn bản rõ tiếp theo không bị ảnh hưởng. Tuy nhiên, nó có nhược điểm là dễ bị tấn công hơn CFB.

### Chế độ CTR (counter)

Hình 2.29 mô tả sơ đồ khối của CTR. Một Counter có kích thước bằng kích thước của khối bản rõ được sử dụng. Giá trị của counter phải khác với mỗi khối bản rõ được mật mã hóa. Cụ thể, counter được khởi tạo tại một giá trị nào đó, sau đó được tăng lên 1 cho các khối tiếp theo. Đối với quá trình mật mã hóa, counter được mật mã hóa và được thực hiện XOR với bản rõ để tạo ra khối bản mã. Đối với quá trình giải mật mã, mỗi counter được mật mã hóa được thực hiện XOR với bản mã để khôi phục khối bản rõ tương ứng. Với các counter lần lượt là  $T_1, T_2, \dots, T_N$ , chế độ CTR được định nghĩa như sau:

CTR	$C_j = P_j \oplus E(K, T_j) \quad j = 1, \dots, N - 1$ $C_N^* = P_N^* \oplus \text{MSB}_u[E(K, T_N)]$	$P_j = C_j \oplus E(K, T_j) \quad j = 1, \dots, N - 1$ $P_N^* = C_N^* \oplus \text{MSB}_u[E(K, T_N)]$
-----	--	--

Đối với khối bản rõ cuối cùng (có thể chỉ có  $u$  bit), chỉ  $u$  bit có ý nghĩa nhất (bên trái) của khối mật mã cuối cùng được sử dụng cho phép XOR cùng khối bản rõ đó.



Hình 2.29: Chế độ CTR

## 2.5 Tạo số giả ngẫu nhiên và mật mã dòng

### 2.5.1 Nguyên lí tạo số giả ngẫu nhiên

Số ngẫu nhiên đóng vai trò quan trọng trong việc sử dụng mật mã hóa cho các ứng dụng, các giao thức, và các thuật toán an toàn mạng khác nhau như trong các sơ đồ phân phối khóa và nhận thực lẫn nhau, tạo khóa phiên, tạo các khóa cho thuật toán mật mã khóa công khai RSA, hay tạo luồng bit cho mật mã dòng đối xứng.

Có hai tiêu chí được sử dụng để đánh giá tính ngẫu nhiên của chuỗi ngẫu nhiên, đó là:

- + Phân phối đồng nhất: phân phối các bit trong chuỗi phải là đồng nhất; nghĩa là tần suất xuất hiện của các bit 0 và 1 phải là như nhau.
- + Độc lập: không chuỗi con nào trong chuỗi ngẫu nhiên đó có thể được suy ra từ các chuỗi con khác.

Trong các ứng dụng như nhận thực lẫn nhau, tạo khóa phiên, và mật mã dòng, tính không thể dự đoán trước được yêu cầu. Yêu cầu này không chỉ là chuỗi số đó là ngẫu nhiên thống kê mà các thành phần kế tiếp trong chuỗi đó còn phải là không dự đoán trước được. Với các chuỗi ngẫu nhiên “đúng”, mỗi số là độc lập thống kê với các số khác trong chuỗi đó và vì vậy là không thể dự đoán trước được. Mặc dù, các số ngẫu nhiên đúng được sử dụng trong một số ứng dụng, chúng có một số hạn chế như tính không hiệu quả. Do vậy, việc sử dụng các thuật toán để tạo các chuỗi số ngẫu nhiên là hiệu quả hơn.

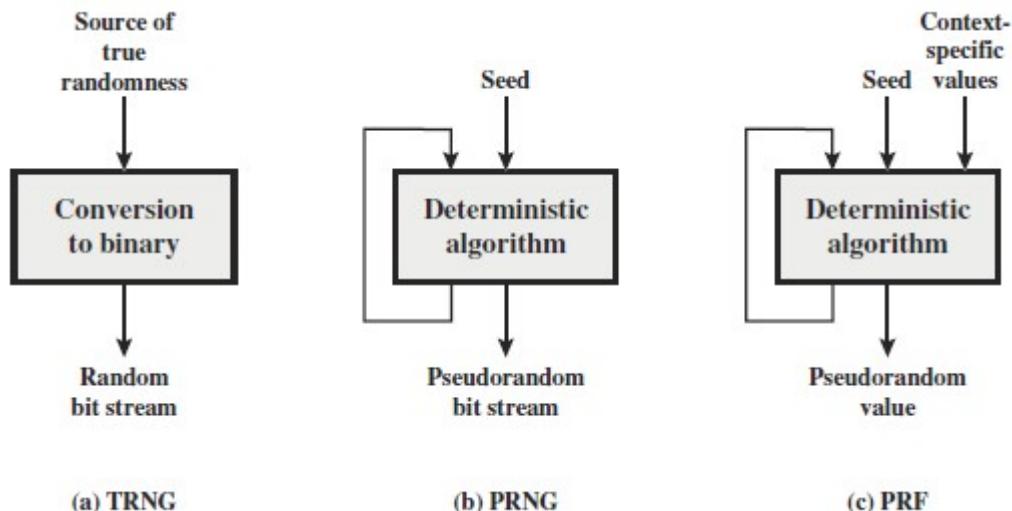
Các ứng dụng mật mã hóa thường tận dụng các kỹ thuật thuật toán cho việc tạo số ngẫu nhiên. Các thuật toán này là xác định và vì vậy tạo ra các chuỗi số không phải là ngẫu nhiên thống kê. Tuy nhiên, nếu thuật toán tốt, chuỗi số kết quả sẽ qua được nhiều bài kiểm tra về tính ngẫu nhiên. Các số như vậy được gọi là các số giả ngẫu nhiên.

Hình 2.30 mô tả bộ tạo số ngẫu nhiên đúng (TRNG) với hai bộ tạo số giả ngẫu nhiên. TRNG có đầu vào là một nguồn thực sự ngẫu nhiên; nguồn này thường được đề cập đến là nguồn entropy (entropy source). Nguồn entropy được lấy ra từ môi trường vật lý của máy tính và có thể tính đến cả các mẫu định thời bấm phím, hoạt động của ổ đĩa, sự di chuyển chuột, và các giá trị tức thời của đồng hồ hệ thống. Một nguồn, hoặc sự kết hợp của các nguồn, được coi là đầu vào của thuật toán để tạo ra đầu ra là chuỗi nhị phân ngẫu nhiên. TRNG có thể chỉ đơn giản là phép biến đổi các nguồn tương tự thành đầu ra nhị phân.

Ngược lại, PRNG có đầu vào là một giá trị cố định, được gọi là hạt giống (seed), và tạo ra chuỗi bit đầu ra sử dụng một thuật toán xác định. Thường thì seed được tạo ra bởi một bộ TRNG. Như chỉ ra trong hình 2.30, một số kết quả của thuật toán được phản hồi lại như là đầu vào của thuật toán bằng một đường hồi tiếp. Chú ý rằng, dòng bit đầu ra được xác định chỉ bởi một hoặc nhiều giá trị đầu vào, do đó kẻ tấn công mà biết thuật toán và seed thì có thể tạo lại toàn bộ dòng bit.

Hình 2.30 đưa ra hai bộ tạo chuỗi giả ngẫu nhiên khác nhau:

- + Bộ tạo số giả ngẫu nhiên: thuật toán được sử dụng để tạo ra chuỗi bit kết thúc mở được gọi là PRNG. Ứng dụng phổ biến cho chuỗi bit này là đầu vào của mật mã dòng đối xứng.
- + Hàm giả ngẫu nhiên (PRF): PRF được sử dụng để tạo ra chuỗi bit giả ngẫu nhiên có độ dài cố định. PRF cũng có đầu vào là seed và một thông tin khác như là ID người sử dụng hay ID ứng dụng.



Hình 2.30: Nguyên lý tạo số ngẫu nhiên và giả ngẫu nhiên

### 2.5.2 Bộ tạo số giả ngẫu nhiên

Trong phần này, hai kiểu thuật toán cho bộ PRNG được trình bày.

#### Các bộ tạo đồng dạng tuyến tính (Linear Congruential)

Kỹ thuật được sử dụng rộng rãi cho việc tạo số giả ngẫu nhiên là thuật toán đồng dạng. Thuật toán này có 4 tham số như sau:

$m$	the modulus	$m > 0$
$a$	the multiplier	$0 < a < m$
$c$	the increment	$0 \leq c < m$
$X_0$	the starting value, or seed	$0 \leq X_0 < m$

Chuỗi số ngẫu nhiên  $\{X_n\}$  được tính như sau:

$$X_{n+1} = (aX_n + c) \bmod m$$

Nếu  $m$ ,  $a$ ,  $c$ , và  $X_0$  là số nguyên, kỹ thuật này sẽ tạo ra chuỗi số nguyên với mỗi số nguyên nằm trong dải  $0 \leq X_n < m$ .

Việc lựa chọn các giá trị cho  $a$ ,  $c$ , và  $m$  là vấn đề then chốt trong việc phát triển một bộ tạo số ngẫu nhiên tốt. Ví dụ, với  $a = c = 1$ . Chuỗi được tạo ra rõ ràng là không thỏa mãn. Với các giá trị  $a = 7$ ,  $c = 0$ ,  $m = 32$ , và  $X_0 = 1$ . Bộ tạo ngẫu nhiên tạo ra chuỗi  $\{7, 17, 23, 1, 7, \dots\}$ , chuỗi này cũng không thỏa mãn. Trong số 32 giá trị có thể, chỉ bốn giá trị được sử dụng; do đó, chuỗi đó được coi là có chu kỳ là 4. Nếu thay  $a$  bằng 5, thì chuỗi mới là  $\{5, 25, 29, 17, 21, 9, 13, 1, 5, \dots\}$ , khi đó chu kỳ tăng lên thành 8.

Nếu  $m$  là một số rất lớn, có khả năng tạo ra một chuỗi dài các số ngẫu nhiên khác nhau. Tiêu chí chung đó là  $m$  gần với số nguyên không âm lớn nhất có thể đối với mỗi máy tính xác định trước. Do đó, giá trị của  $m$  gần hoặc bằng  $2^{31}$  sẽ được lựa chọn.

Có ba tiêu chí được sử dụng để đánh giá bộ tạo số ngẫu nhiên như sau:

- + Hàm tạo sẽ là hàm tạo toàn chu kỳ. Nghĩa là, hàm tạo sẽ tạo ra tất cả các số từ 0 đến  $m-1$  trước khi lặp lại.
- + Chuỗi được tạo ra phải là ngẫu nhiên
- + Hàm sẽ thực hiện một cách hiệu quả với số 32 bit.

Với các giá trị thích hợp của  $a$ ,  $c$ , và  $m$ , chuỗi số tạo ra có thể đáp ứng được ba tiêu chí đánh giá trên. Với tiêu chí đánh giá đầu tiên, nếu  $m$  là số nguyên tố và  $c = 0$ , thì đối với giá trị nào đó của  $a$ , chu kỳ của hàm tạo số ngẫu nhiên sẽ là  $m-1$ . Đối với số 32 bit, giá trị nguyên tố của  $m$  là  $2^{31}-1$ . Do đó, hàm tạo số ngẫu nhiên trở thành:

$$X_{n+1} = (aX_n) \bmod (2^{31} - 1)$$

Độ mạnh của thuật toán đồng dạng tuyến tính phụ thuộc vào số nhân  $a$  và  $m$  được lựa chọn. Tuy nhiên, không có sự ngẫu nhiên nào trong thuật toán, ngoại trừ việc lựa chọn giá trị khởi tạo  $X_0$ . Khi giá trị đó được lựa chọn, các số còn lại là chuỗi theo sau được xác định. Điều này là lợi thế cho các kẻ tấn công. Nếu kẻ tấn công biết rằng thuật toán đồng dạng tuyến tính được sử dụng và nếu các tham số đã biết (ví dụ  $a = 7^5$ ,  $c = 0$ ,  $m = 2^{31}-1$ ), thì khi một số được phát hiện ra, tất cả các số tiếp theo sẽ biết được, chỉ cần biết một phần nhỏ của chuỗi số là đủ để xác định các tham số của thuật toán. Giả sử rằng kẻ tấn công có khả năng xác định các giá trị  $X_0, X_1, X_2$ , và  $X_3$ , thì

$$X_1 = (aX_0 + c) \bmod m$$

$$X_2 = (aX_1 + c) \bmod m$$

$$X_3 = (aX_2 + c) \bmod m$$

Từ các phương trình này, các giá trị của  $a$ ,  $c$ , và  $m$  có thể tìm được.

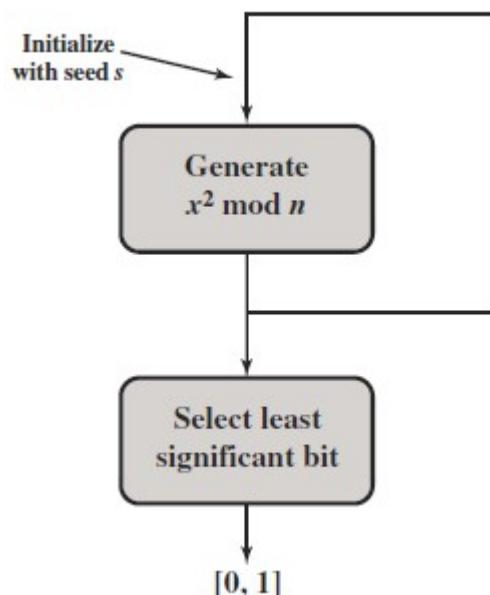
Do đó, mục tiêu của bộ PRNG là làm sao để phần chuỗi mà kẻ tấn công có thể khám phá ra là không đủ để xác định các thành phần tiếp theo của chuỗi. Mục tiêu đó có thể đạt được bằng một số cách, ví dụ như sử dụng đồng hồ hệ thống nội để thay đổi dòng số ngẫu nhiên. Một cách sử dụng đồng hồ là khởi tạo lại chuỗi đó sau mỗi  $N$  số sử dụng giá trị đồng hồ hiện tại ( $\text{mod } m$ ) làm seed mới. Cách khác đơn giản hơn là thêm giá trị đồng hồ hiện tại vào mỗi số ngẫu nhiên ( $\text{mod } m$ ).

### Bộ tạo BBS (Blum Blum Shub)

Một phương pháp phổ biến để tạo các số giả ngẫu nhiên an toàn được biết như là bộ tạo BBS (hình 2.31). Hoạt động của bộ tạo BBS như sau. Đầu tiên, lựa chọn hai số nguyên tố lớn,  $p$  và  $q$ . Hai số đó khi chia cho 4 đều có số dư là 3, nghĩa là

$$p \pmod{4} = q \pmod{4} = 3$$

Ví dụ, các số nguyên tố 7 và 11 đều thỏa mãn điều kiện trên. Đặt  $n = p \times q$ .



Hình 2.31: Sơ đồ khối bộ tạo BBS

Tiếp theo, chọn số ngẫu nhiên  $s$ , sao cho cả  $p$  và  $q$  đều không phải là thừa số của  $s$ . Sau đó, bộ tạo BBS tạo ra chuỗi bit  $B_i$  theo thuật toán sau:

$$X_0 = s^2 \bmod n$$

for  $i = 1$  to  $\infty$

$$X_i = (X_{i-1})^2 \bmod n$$

$$B_i = X_i \bmod 2$$

Do đó, bit có ý nghĩa thấp nhất được lấy ra tại mỗi vòng lặp. Bảng sau chỉ ra ví dụ hoạt động của bộ tạo BBS với  $n = 192649 = 383 \times 503$  và seed  $s = 101355$ .

Bảng 2.5: Ví dụ hoạt động của bộ tạo BBS

$i$	$X_i$	$B_i$
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

BBS còn được gọi là bộ tạo bit giả ngẫu nhiên an toàn bảo mật (CSPRNG). Tính an toàn của BBS phụ thuộc vào hai thừa số nguyên tố p và q của n.

### 2.5.3 Mật mã dòng

Mật mã dòng thực hiện mật mã bắn rỗng theo từng byte tại một thời điểm, mặc dù mật mã dòng có thể được thiết kế để hoạt động trên từng bit tại một thời điểm hoặc trên đơn vị lớn hơn 1 byte tại một thời điểm. Hình 2.32 đưa ra sơ đồ cấu trúc của mật mã dòng. Trong cấu trúc này, khóa là đầu vào của bộ tạo bit giả ngẫu nhiên. Bộ tạo bit giả ngẫu nhiên này tạo ra dòng số 8 bit ngẫu nhiên. Đầu ra của bộ tạo, được gọi là dòng khóa, được kết hợp một byte tại một thời điểm với dòng bắn rỗng sử dụng phép XOR. Ví dụ, nếu byte tiếp theo được tạo ra bởi bộ tạo này là 01101100 và byte bắn rỗng tiếp theo là 11001100, thì byte bắn mã sẽ là:

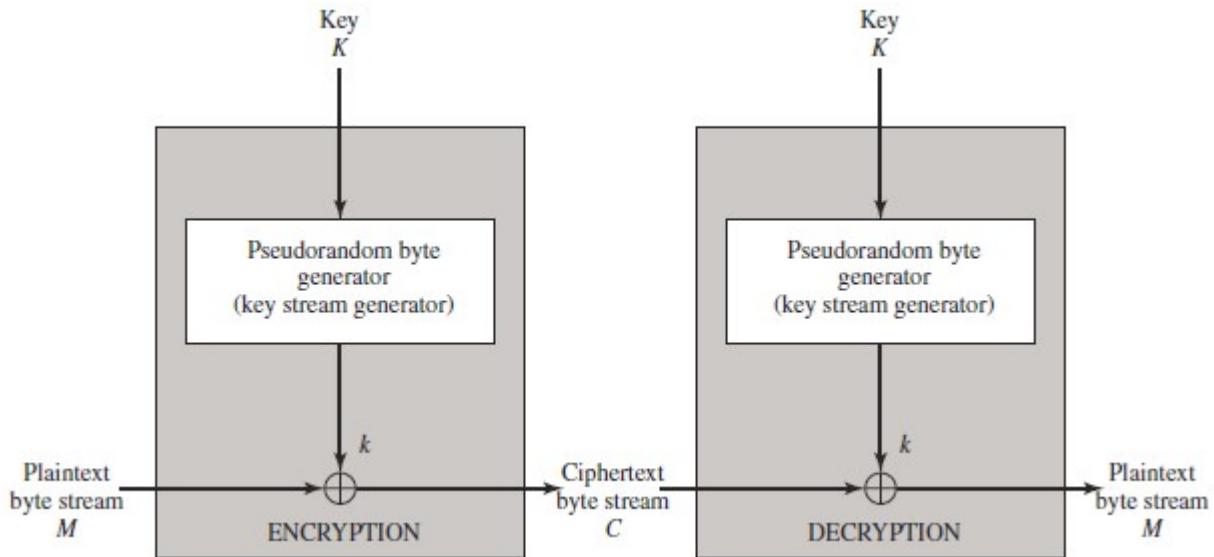
```

11001100 plaintext
⊕ 01101100 key stream
10100000 ciphertext

```

Giải mật mã yêu cầu sử dụng cùng chuỗi giả ngẫu nhiên:

$\oplus$     10100000 ciphertext  
 01101100 key stream  
 11001100 plaintext



Hình 2.32: Sơ đồ mật mã dòng

Như vậy, có thể thấy mật mã dòng tương tự như mật mã hóa One-Time Pad. Điểm quan trọng nhất của các mật mã dòng là bộ tạo số ngẫu nhiên. Nếu chọn khóa có chiều dài ngắn thì không bảo đảm an toàn, còn nếu chọn khóa có chiều dài bằng chiều dài tin như One-Time Pad thì lại không thực tế. Bộ tạo số của mật mã dòng cân bằng giữa hai điểm này, cho phép dùng một khóa ngắn nhưng dãy số tạo ra bảo đảm một độ ngẫu nhiên cần thiết như khóa của One-time Pad, dùng rằng không hoàn toàn thực sự ngẫu nhiên. Phần tiếp theo trình bày phương pháp mật mã hóa dòng tiêu biểu là RC4.

#### 2.5.4 RC4

RC4 là một phương pháp mật mã dòng được thiết kế vào năm 1987 bởi Ron Rivest cho RSA. Nó là một mật mã dòng có kích thước khóa thay đổi với các hoạt động hướng byte. Thuật toán được dựa trên việc sử dụng phép hoán vị ngẫu nhiên. RC4 được dùng trong giao thức SSL để bảo mật dữ liệu trong quá trình truyền dữ liệu giữa Web Server và trình duyệt Web. Ngoài ra RC4 còn được sử dụng trong giao thức mã hóa WEP và giao thức WPA (Wifi Protected Access) của mạng Wireless LAN.

Thuật toán RC4 là khá đơn giản. Khóa có độ dài thay đổi từ 1 tới 128 byte (8 đến 2048 bit) được sử dụng để khởi tạo vector S gồm 256 byte, với các phần tử là  $S[0]$ ,  $S[1], \dots, S[255]$ . Tại tất cả các thời điểm S gồm một hoán vị của tất cả các số 8 bit từ 0

đến 255. Đối với quá trình mật mã hóa và giải mật mã, byte k (hình 2.32) được tạo ra từ S bằng cách lựa chọn một trong 255 phần tử theo một cách có hệ thống. Khi mỗi giá trị của k được tạo ra, các phần tử trong S lại được hoán vị một lần nữa.

### **Khởi tạo S**

Để bắt đầu, các phần tử của S được thiết lập với các giá trị từ 0 đến 255 theo thứ tự tăng dần; nghĩa là  $S[0]=0, S[1]=1, \dots, S[255]=255$ . Một vector tạm thời T cũng được tạo ra. Nếu độ dài khóa K là 256 byte, thì K được chuyển tới T. Nếu không, đối với khóa có độ dài keylen byte, keylen phần tử đầu tiên của T được sao chép từ K, và sau đó K được lặp lại nhiều lần để điền đầy vào T. Hoạt động đó có thể tóm tắt như sau:

```
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod keylen];
```

Tiếp theo, T được sử dụng để tạo ra hoán vị đầu tiên của S, cụ thể như sau:

```
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```

Bởi vì phép toán thực hiện trên S chỉ là tráo đổi các phần tử của S, ảnh hưởng duy nhất là hoán vị. S vẫn chứa tất cả các số từ 0 đến 255.

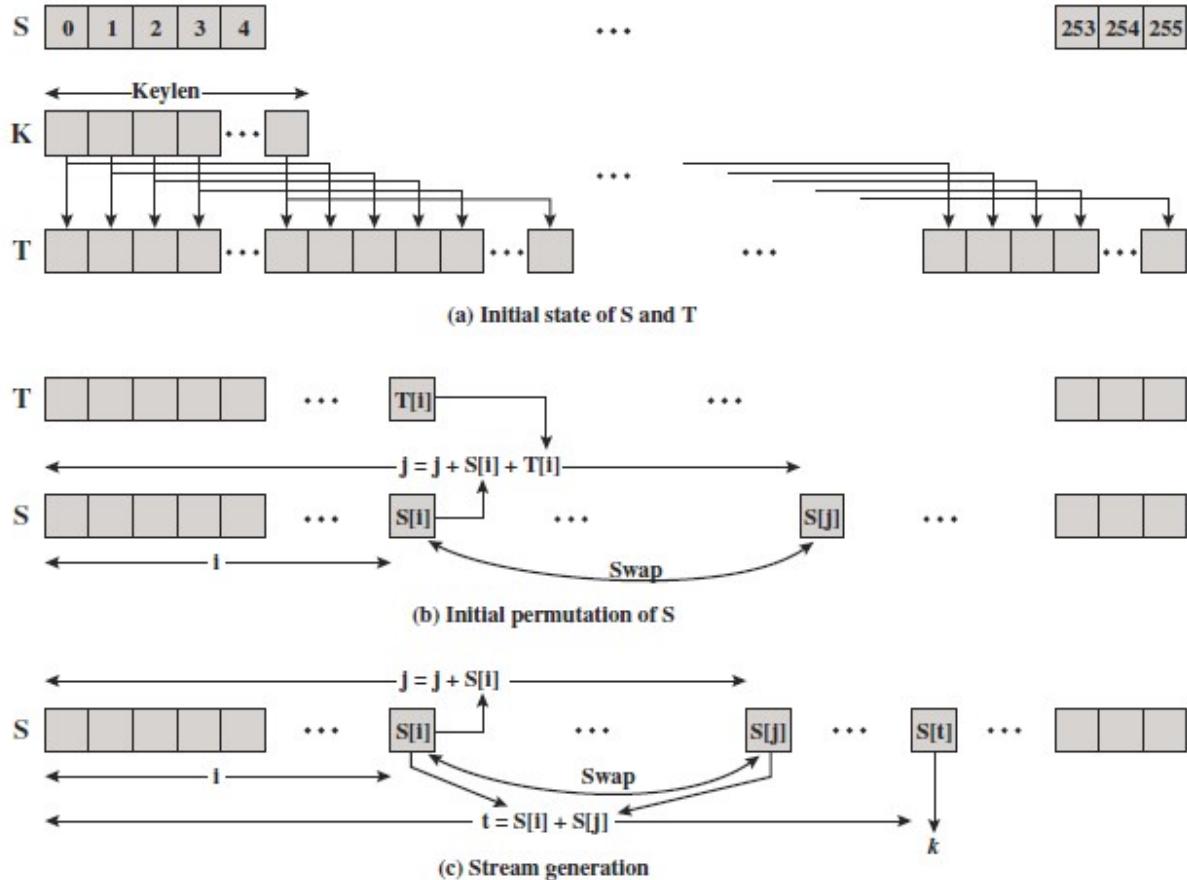
### **Tạo dòng**

Khi vector S đã được khởi tạo, khóa đầu vào không còn được sử dụng nữa. Việc tạo dòng được thực hiện như sau:

```
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
```

Để mật mã hóa, thực hiện phép XOR giá trị k với byte tiếp theo của bản rõ. Để giải mật mã, thực hiện XOR giá trị k với byte tiếp theo của bản mã.

Hình 2.33 minh họa hoạt động của RC4.



Hình 2.33: Mật mã hóa RC4

Quá trình tạo số của RC4 cũng tạo ra dãy số ngẫu nhiên, khó đoán trước, vì vậy RC4 đạt được mức độ an toàn cao hơn mật mã hóa One-Time Pad. Mật mã hóa RC4 hoàn toàn được thực hiện trên các số nguyên một byte do đó tối ưu cho việc thiết lập bằng phần mềm và tốc độ thực hiện nhanh hơn so với mật mã khôi.

## 2.6 Kết luận chương 2

Trong chương này, các khái niệm cơ bản về mật mã hóa, mật mã khóa đối xứng, mật mã khôi, và mật mã dòng được trình bày một cách chi tiết. Ngoài ra, các kiến trúc, nguyên lý thiết kế, và các ứng dụng của mật mã khôi cũng được đưa ra. Cũng trong chương này, các giải thuật phổ biến như RC4, DES, và AES đã được mô tả và giải thích một cách đầy đủ và dễ hiểu.

## Câu hỏi ôn tập chương 2

1. Trình bày các thành phần cơ bản của mật mã khóa đối xứng
2. Trình bày cấu trúc chung của mật mã khối
3. Trình bày sự khác nhau giữa mật mã khối và mật mã dòng
4. Trình bày cấu trúc mật mã hóa và giải mật mã Feistel
5. Trình bày cấu trúc DES
6. Trình bày thuật toán sinh khóa con của DES
7. Giải thích hiệu ứng lan truyền
8. Trình bày nguyên lý thiết kế mật mã khối
9. Trình bày cấu trúc AES
10. Giải thích ý nghĩa và mục đích của các hàm biến đổi trong AES
11. Phân biệt hai hàm SubBytes và SubWord
12. Phân biệt hai hàm ShiftRows và RotWord
13. Trình bày quá trình tạo khóa AES
14. Nêu và phân tích các kiểu mật mã hóa nhiều lần
15. Trình bày các chế độ và ứng dụng mật mã khối
16. Trình bày nguyên lý tạo số giả ngẫu nhiên
17. Vẽ và phân tích sơ đồ mật mã dòng
18. Trong mô hình mật mã Feistel, với khối  $n$  bit, chứng minh rằng số các phép ánh xạ thuận nghịch cho mật mã khối lý tưởng là  $2^n!$ .
19. Tìm 8 từ đầu tiên của việc tạo khóa biệt匙 128 bit toàn bit 0
20. Cho bản rõ  $\{000102030405060708090A0B0C0D0E0F\}$  và khóa  $\{01010101010101010101010101010101\}$ :
  - a. Tìm giá trị ma trận khởi tạo
  - b. Tìm giá trị ma trận sau khối AddRoundKey khởi tạo
  - c. Tìm giá trị ma trận sau SubBytes
  - d. Tìm giá trị ma trận sau ShiftRows
  - e. Tìm giá trị ma trận sau MixColumns

## Chương 3: Mật mã khóa bất đối xứng

### 3.1. Mật mã khóa công khai và RSA

#### 3.1.1 Nguyên lý hệ thống mật mã khóa công khai

Khái niệm về mật mã khóa công khai phát triển từ hai trỏ ngại của mã hóa đối xứng. Vấn đề đầu tiên là phân phối khóa và vấn đề thứ hai là chữ ký số. Vấn đề phân phối khóa trong hệ mật đối xứng yêu cầu hoặc (1) hai thành viên cùng chia sẻ khóa và cách nào để phân phối cho chúng, hoặc (2) sử dụng một trung tâm phân phối khóa. Whitfield Diffie, người phát hiện ra mã hóa khóa công khai (cùng với Martin Hellman, Đại học Stanford), lý luận rằng yêu cầu thứ hai này phủ nhận bản chất của mật mã: khả năng duy trì bí mật tổng thể sẽ xâm phạm thông tin riêng tư. Vấn đề thứ hai không liên quan đến vấn đề đầu tiên là chữ ký kỹ thuật số. Nếu việc sử dụng mật mã đã trở nên phổ biến, không chỉ trong những tình huống quân sự mà còn trong thương mại và mục đích cá nhân, thì tin nhắn điện tử và các văn bản sẽ cần xác nhận tương đương với chữ ký được sử dụng trong các tài liệu giấy.

#### 3.1.1.1 Hệ mật khẩu công khai

Giải thuật bất đối xứng dựa trên cơ chế sử dụng một khóa để mã hóa và một khóa khác (có liên quan tới khóa mã) để giải mã. Các giải thuật này có đặc điểm quan trọng sau đây.

- + Nó là tính toán khả thi để xác định một khóa giải mã duy nhất thông hiểu về thuật toán mã hóa và khóa mã hóa.

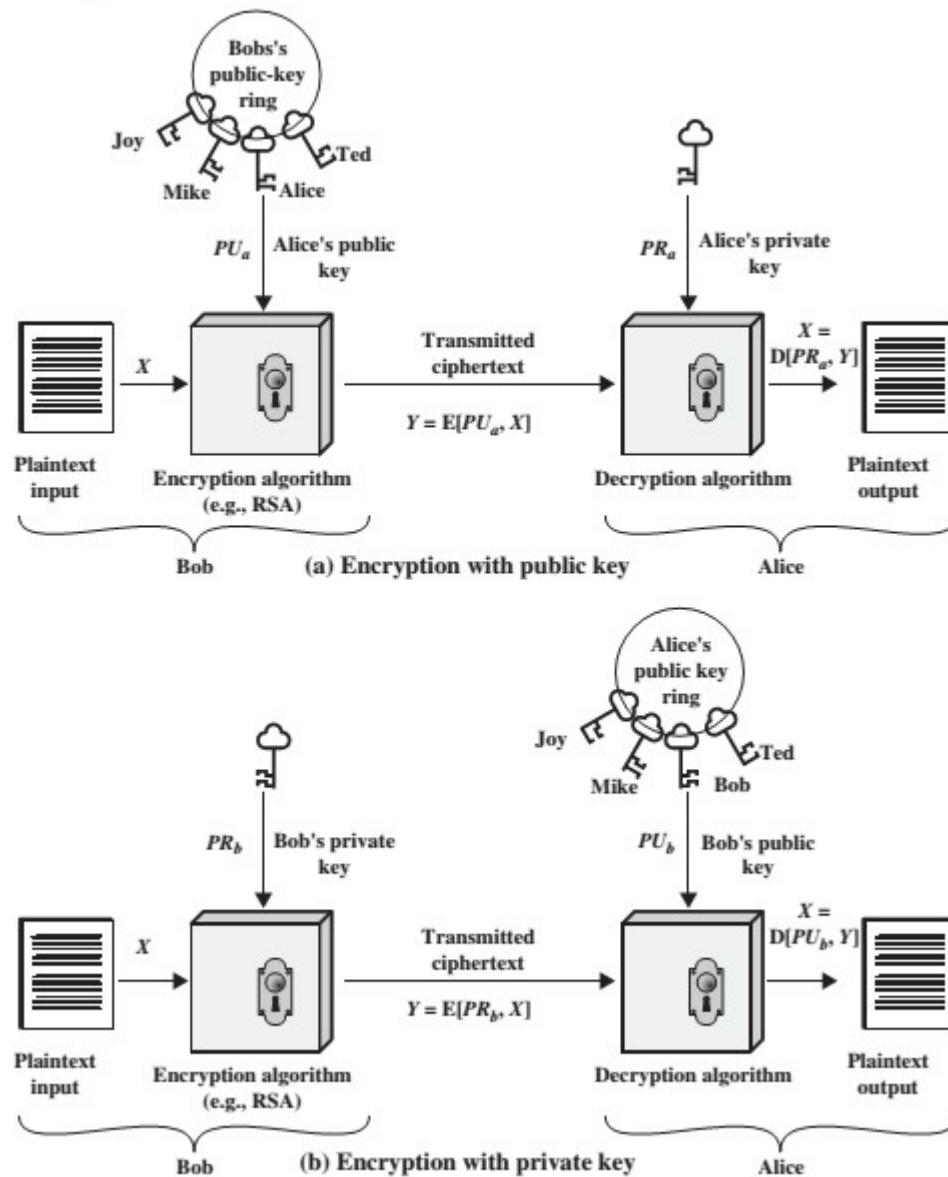
Thêm vào đó, đối với một số thuật toán như RSA còn bổ sung một số đặc tính sau:

Một trong hai khoá có liên quan có thể được sử dụng để mã hóa và khóa còn lại để giải mã.

Một lưu đồ mã hóa khóa công khai có sáu thành phần (hình 3.1a);

- + Plaintext (bản rõ): Đây là bản tin hoặc dữ liệu có thể đọc được và là đầu vào của thuật toán.
- + Thuật toán mã hóa: Các thuật toán mã hóa thực hiện các biến đổi khác nhau trên bản rõ.

- + Khóa công cộng và khóa riêng: Đây là một cặp chìa khóa đã được chọn để nếu một khóa được sử dụng để mã hóa, thì khóa kia được sử dụng để giải mã. Các phép chuyển đổi chính xác được thực hiện bởi các thuật toán phụ thuộc vào khóa công khai hoặc riêng mà được coi là các đầu vào.



Hình 3. 1: Hệ mật khẩu công khai

- + Bản mã: Đây là bản tin được tạo ra tại đầu ra thuật toán, nó phụ thuộc vào bản rõ và khóa. Đối với một bản tin, hai khóa khác nhau sẽ tạo ra hai bản mã khác nhau.

- + Thuật toán giải mã: Thuật toán này chấp nhận các bản mã và khóa phù hợp để các bản rõ ban đầu.

Một số bước cơ bản có thể gồm:

- + Mỗi người sử dụng tạo ra một cặp khóa được sử dụng để mã hóa và giải mã các bản tin.
- + Mỗi người sử dụng đặt một trong hai khóa làm khóa công cộng và khóa kia bí mật. Như hình 3.1a, mỗi người dùng duy trì một tập của các khóa công cộng thu được từ những người khác.
- + Nếu Bob muốn gửi một bản tin bí mật cho Alice, Bob mã hóa bản tin bằng khóa công khai của Alice.
- + Khi Alice nhận được bản tin, cô giải mã bằng khóa riêng của mình. Không người nhận nào khác có thể giải mã bản tin bởi vì chỉ có Alice biết khóa riêng của Alice.

Với tiếp cận này, tất cả những thành viên tham gia đều có quyền truy cập vào các khóa công khai và khóa riêng được tạo cục bộ của từng người tham gia và không cần cơ chế phân phối. Chừng nào khóa riêng của người dùng được bảo vệ và bí mật, thì truyền thông là an toàn. Tại bất kỳ thời điểm nào, một hệ thống có thể thay đổi khóa riêng của nó và tạo ra khóa công khai mới để thay thế khóa công khai cũ.

Bảng 3.1 tóm tắt một số khía cạnh quan trọng của mã hóa đối xứng và khóa công khai. Để phân biệt, ta coi các khóa được sử dụng trong mã hóa đối xứng như một khóa bí mật. **Hai khóa được sử dụng để mã hóa bất đối xứng được gọi là khóa công khai và khóa riêng.** Tất nhiên là **khóa riêng được giữ bí mật**, nhưng nó được gọi là **khóa riêng chứ không phải là một khóa bí mật để tránh nhầm lẫn với mã hóa đối xứng.**

Chúng ta hãy xem xét kỹ hơn các yếu tố thiết yếu của một lưu đồ mã hóa khóa công khai trên hình 3.2. Ở đây có một số nguồn A tạo ra một bản tin rõ,  $X = [X_1, X_2, \dots, X_M]$ .  $M$  phần tử của  $X$  là các chữ cái trong bảng chữ cái. Bản tin này được gửi đến cho các điểm đến B. B tạo ra một cặp quan hệ của các khóa: một khóa công khai  $PUb$  và một khóa riêng  $PRb$ . Khóa  $PUb$  được công khai truy nhập bởi A.

Với bản tin  $X$  và mã hóa bởi khóa  $PUb$ , A tạo các bản mã  $Y = [Y_1, Y_2, \dots, Y_N]$ :

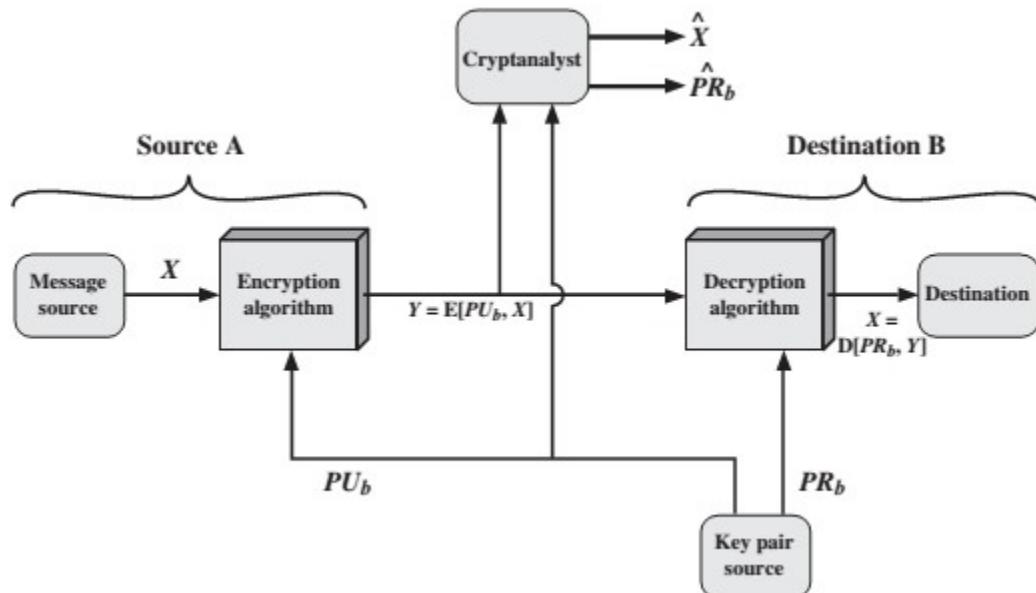
$$Y = E(PUb, X)$$

Bên phía nhận có khóa riêng phù hợp, thực hiện biến đổi ngược:

$$X = D(PR_b, Y)$$

Bảng 3.1: Mã khóa công khai và truyền thống

Mã hóa truyền thống	Mã hóa công khai
<p>Yêu cầu hoạt động</p> <ul style="list-style-type: none"> <li>- Cùng thuật toán và cùng khóa để mã hóa và giải mã</li> <li>- Bên gửi và bên nhận phải chia sẻ thuật toán và khóa</li> </ul> <p>Yêu cầu bảo mật</p> <ul style="list-style-type: none"> <li>- Khóa phải giữ bí mật</li> <li>- Phải bắt khả thi trong việc giải mã nếu không có khóa bí mật</li> <li>- Hiểu biết về thuật toán và các mẫu của bản mã không đủ để xác định khóa</li> </ul>	<p>Yêu cầu hoạt động</p> <ul style="list-style-type: none"> <li>- Một thuật toán và một khóa để mã hóa và một thuật toán khác và một khóa khác để giải mã</li> <li>- Bên gửi và bên nhận phải giữ một khóa riêng</li> </ul> <p>Yêu cầu bảo mật</p> <ul style="list-style-type: none"> <li>- Một trong hai khóa phải giữ bí mật</li> <li>- Phải bắt khả thi trong việc giải mã nếu không có khóa bí mật</li> <li>- Hiểu biết về thuật toán và các mẫu của bản mã không đủ để xác định khóa</li> </ul>



Hình 3. 2: Bảo mật của hệ mật khẩu công khai

Một kẻ tấn công quan sát  $Y$  và có quyền truy cập vào  $PU_b$ , nhưng không có quyền truy cập vào  $PR_b$  hoặc  $X$ , phải cố gắng để khôi phục lại  $X$  và / hoặc  $PR_b$ . Giả thiết rằng,

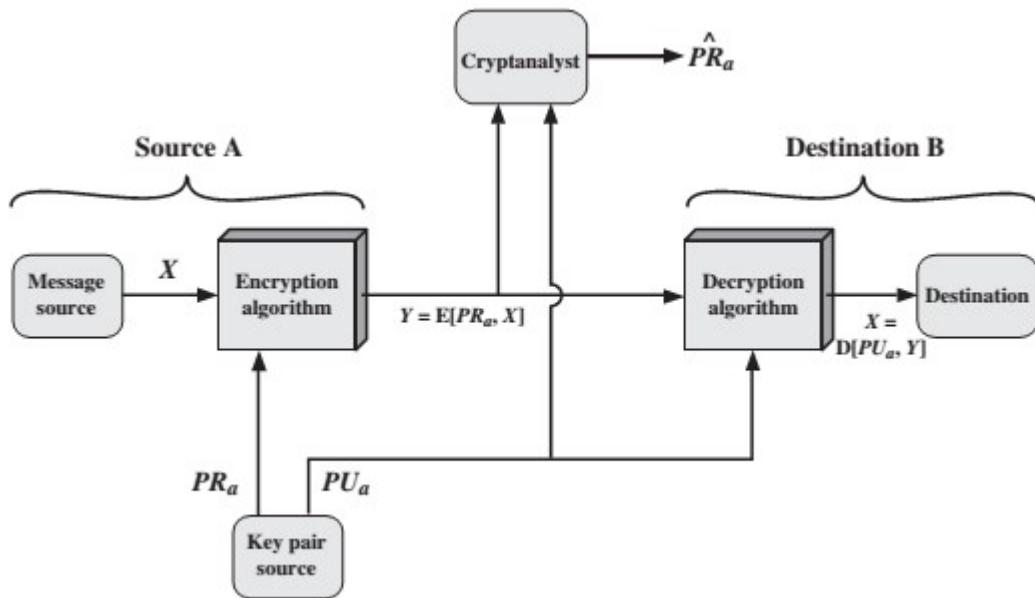
kẻ tấn công không biết về sự thuật toán mã hóa ( $E$ ) và giải mã ( $D$ ). Nếu kẻ tấn công chỉ quan tâm đến bản tin cụ thể, thì sẽ tập trung vào mục tiêu khôi phục  $X$  bằng cách tạo ra một bản rõ ước tính  $\hat{X}$ . Thông thường, kẻ tấn công thường hướng đến các bản tin tương lai và hướng đến sự khôi phục  $PR_b$  bằng cách tạo ra một ước tính  $\hat{PR}_b$ .

Ta đã biết, một trong hai khoá có liên quan có thể được sử dụng để mã hóa, và khóa khác đang được sử dụng để giải mã. Điều này cho phép nhiều hơn một lược đồ mã hóa khác nhau thực hiện. Lưu đồ trong hình 3.2 cung cấp bảo mật, và lưu đồ trong 3.1 và 3.3 sử dụng khóa công khai để cung cấp nhận thực:

$$Y = E(PU_a, X)$$

$$X = D(PU_a, Y)$$

Trong trường hợp này, A chuẩn bị một bản tin đến B và mã hóa nó bằng khóa riêng của mình trước khi truyền nó. **B có thể giải mã các bản tin bằng khóa công khai của nó.** Bởi vì các bản tin được mã hóa bằng khóa riêng của A, thì chỉ có duy nhất A tạo ra được bản tin như vậy. Vì thế, toàn bộ bản tin được mã hóa phục vụ như một chữ ký kỹ thuật số.Thêm vào đó, không thể thay đổi được bản tin mà không truy cập vào khóa riêng của A, nên bản tin được xác thực cả về nguồn và về tính toàn vẹn của dữ liệu.



Hình 3. 3: Nhận thực của hệ mật khẩu công khai

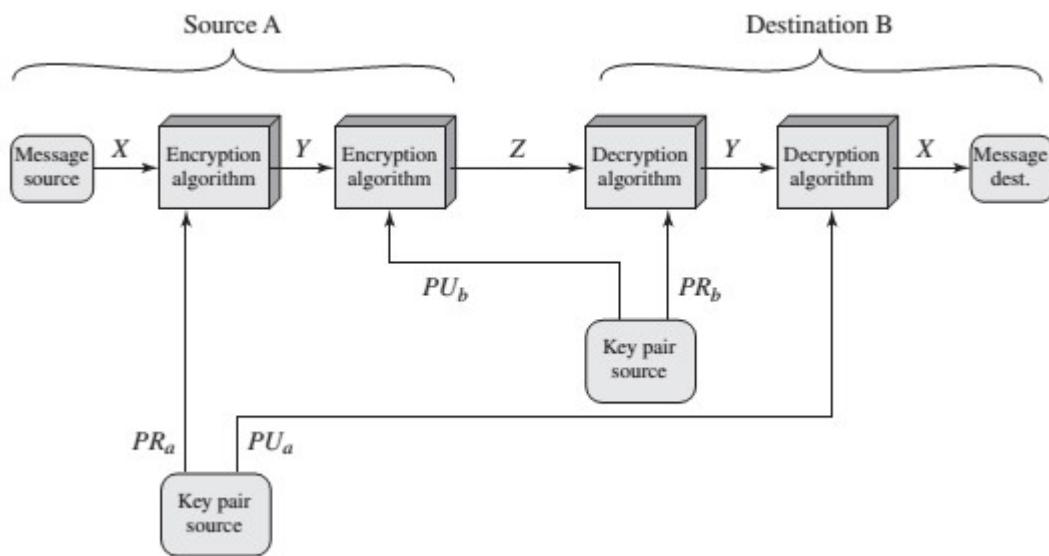
Điều quan trọng cần nhấn mạnh là quá trình mã hóa được mô tả trong Hình 3.1b và 3.3 không cung cấp bảo mật. Vì thế, bản tin được gửi an toàn khi không có sự thay đổi

nhưng không phải từ việc nghe trộm. Đây là rõ ràng trong trường hợp của một chữ ký dựa trên một phần của bản tin, vì phần còn lại của bản tin được truyền rõ ràng. Ngay cả trong trường hợp mã hóa toàn bộ như trong hình 3.3, thì bản tin cũng không bí mật bởi vì bất kỳ người khác nào đều có thể giải mã bản tin bằng cách sử dụng khóa công khai của người gửi. Tuy nhiên, có thể cung cấp cả chức năng xác thực và bảo mật bằng việc sử dụng kép hai lược đồ khóa công khai (hình 9.4):

$$Z = E(PU_b, E(PR_a, X))$$

$$X = D(PU_a, D(PR_b, Z))$$

Trong trường hợp này, ta bắt đầu như trước bằng cách mã hóa một bản tin, sử dụng khóa của người gửi tin để cung cấp chức năng ký kỹ thuật số. Sau đó, ta mã hóa một lần nữa, bằng cách sử dụng khóa công khai của người nhận, bản mã cuối cùng có thể được giải mã chỉ của người nhận có khóa riêng phù hợp và đảm bảo tính bảo mật. Điểm bất lợi của tiếp cận này là độ phức tạp của thuật toán mã hóa công khai, phải thực hiện bốn lần thay vì hai lần trong mỗi phiên truyền thông.



Hình 3. 4: Nhận thực và bảo mật của hệ mật khẩu công khai

### 3.1.1.2 Các ứng dụng cho hệ mật khẩu công khai

Một hệ mật khẩu công khai được đặc trưng bởi việc sử dụng các thuật toán mã hóa và hai khóa. Tùy thuộc vào ứng dụng, bên gửi có thể gửi khóa bí mật, khóa riêng hoặc cả hai cho bên nhận và tạo thành các kiểu chức năng của hệ mật. Hệ mật khẩu công khai có thể chia thành 3 dạng:

Mã hóa / giải mã: Người gửi sẽ mã hóa một bản tin với người nhận khóa công khai.

- + Chữ ký kỹ thuật số: Người gửi "ký" một tin nhắn với khóa riêng của họ. Ký được thực hiện bởi một thuật toán mã hóa áp dụng cho cả bản tin hoặc một phần của bản tin.
- + Trao đổi khóa: Hai bên hợp tác để trao đổi khóa phiên. Một số tiếp cận khác nhau được tạo ra liên quan đến các khóa riêng của một hoặc cả hai bên.

Một số thuật toán phù hợp cho tất cả ba ứng dụng trên, trong khi các thuật toán khác có thể chỉ được sử dụng cho một hoặc hai trong số các ứng dụng này. Bảng 3.3 chỉ ra các ứng dụng được hỗ trợ bởi một số thuật toán quen thuộc.

Bảng 3.2: Các ứng dụng cho hệ mật khẩu công khai

Thuật toán	Mã hóa/Giải mã	Chữ ký số	Trao đổi khóa
RSA	Yes	Yes	Yes
Đường cong elliptic	Yes	Yes	Yes
Diffie-hellman	No	No	Yes
DSS	No	Yes	No

### 3.1.1.3 Các yêu cầu đối với hệ mật khẩu công khai

Các hệ thống mật mã minh họa trong hình 3.2 tới 3.4 phụ thuộc vào một thuật toán mã hóa với hai khóa liên quan. Diffie và Hellman mặc nhiên công nhận hệ thống này mà không cần chứng minh rằng các thuật toán như vậy có tồn tại hay không. Tuy nhiên, họ đã đặt ra các điều kiện cho thuật toán phải được thực hiện gồm:

- + Tính toán dễ dàng cho phía bên B để tạo ra một cặp khóa (công cộng  $PU_b$ , khóa riêng  $PR_b$ ).
- + Tính toán dễ dàng cho một người gửi A, biết khóa công khai và bản tin được mã hóa, M, để tạo ra các bản mã tương ứng:

$$C = E(PU_b, M)$$

- + Tính toán dễ dàng cho người nhận B để giải mã bản mã bằng cách sử dụng khóa riêng để phục hồi bản tin gốc:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

- + Tính toán không khả thi đối với kẻ tấn công khi biết khóa công khai, nhằm xác định khóa riêng.

### 3.1.2 Giải thuật RSA

Thuật toán Rivest-Shamir-Adleman (RSA) là thuật toán mã hóa khóa công khai được các tác giả Ronald Rivest, Adi Shamir và Leonard Adleman phát triển tại Học Viện Công nghệ Massachusetts (MIT) vào năm 1977. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với điều kiện độ dài khóa đủ lớn.

#### 3.1.2.1 Mô tả thuật toán

RSA tạo một biểu thức với các hàm mũ, bản rõ được mã hóa trong các khối, với mỗi khối có một giá trị nhị phân nhỏ hơn một số  $n$ . Vì vậy, các khối kích thước phải nhỏ hơn hoặc bằng  $\log_2(n) + 1$ ; trong thực tế, kích thước khối là  $i$  bit, với  $2^i < n \leq 2^{i+1}$ . Mã hóa và giải mã được thực hiện theo dạng sau (các khối bản rõ  $M$  và bản mã  $C$ ):

$$C = M^e \text{ mod } n$$

$$M = C^d \text{ mod } n = (M^e)^d \text{ mod } n = M^{ed} \text{ mod } n$$

Cả người gửi và người nhận phải biết giá trị của  $n$ . Người gửi biết giá trị của  $e$ , và chỉ có người nhận biết được giá trị của  $d$ . Như vậy, đây là một thuật toán mã hóa khóa công khai với một khóa công khai của PU = { $e, n$ } và một khóa riêng của PR = { $d, n$ }.

Đối với thuật toán này, để thỏa mãn cho việc mã hóa khóa công khai, các yêu cầu sau đây phải được đáp ứng.

1. Có thể tìm thấy giá trị của  $e, d$ , và  $n$  sao cho  $M^{ed} \text{ mod } n = M$  cho tất cả giá trị  $M < n$ .
2. Nó là tương đối dễ dàng để tính toán  $M^e \text{ mod } n$  và  $C^d \text{ mod } n$  cho tất cả các giá trị của  $M < n$ .
3. Nó có tính khả thi để xác định  $d$  khi biết  $e$  và  $n$ .

Ta cần tìm mối quan hệ  $M^{ed} \text{ mod } n = M$ . Các mối quan hệ trước đó giữ nếu  $e$  và  $d$  là nhân nghịch đảo theo modulo  $\phi(n)$ , với  $\phi(n)$  là hàm Euler. Nó được biểu diễn cho  $p, q$  là các số nguyên tố,  $\phi(p, q) = (p-1) \times (q-1)$ . Mỗi quan hệ giữa  $e$  và  $d$  có thể được thể hiện như sau:

$$ed \bmod \phi(n) = 1$$

Điều này tương đương với

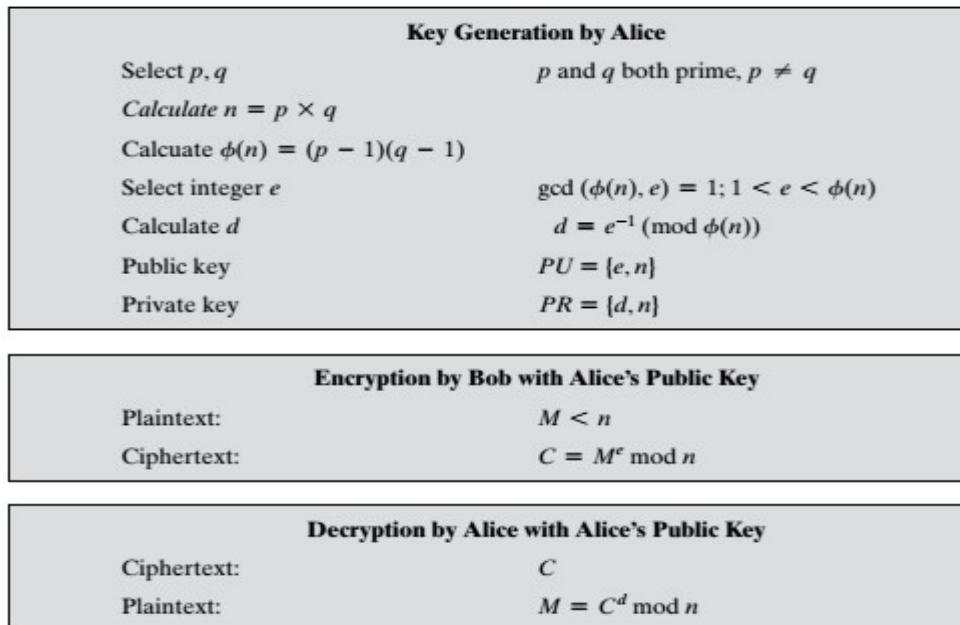
$$ed \equiv 1 \bmod \phi(n)$$

$$d \equiv e^{-1} \bmod \phi(n)$$

Vì vậy, e và d là nhân nghịch đảo của modulo  $\phi(n)$ . Lưu ý rằng, theo quy tắc của số học modul, điều này chỉ đúng nếu d (và do đó e) là nguyên tố cùng nhau với  $\phi(n)$ . Ta tính toán RSA theo các thành phần như sau:

- $p, q$ , hai số nguyên tố (riêng, lựa chọn)
- $n = pq$  (công cộng, tính toán được)
- $e$ , với  $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$  (công khai, lựa chọn)
- $d \equiv e^{-1} \bmod \phi(n)$  (riêng, tính toán được )

Các khóa riêng gồm {d, n} và khóa công khai bao gồm {e, n}. Giả sử rằng người dùng A đã đưa ra khóa công khai của mình và rằng người dùng B muốn gửi bản tin M tới A. Thì B sẽ tính toán  $C = M^e \bmod n$  và truyền C. Khi nhận được bản mã này, người dùng A giải mã bằng cách tính toán  $M = C^d \bmod n$ .



Hình 3. 5: Thuật toán RSA

Hình 3.5 tóm tắt các thuật toán RSA. Nó tương ứng với hình 3.1a: Alice tạo ra một cặp khóa công khai / riêng; Bob mã hóa bằng khóa công khai của Alice; và Alice giải mã

bằng khóa riêng của mình. Một ví dụ được thể hiện trong hình 3.6, các khóa được tạo ra như sau.

Chọn hai số nguyên tố,  $p = 17$ ,  $q = 11$ .

2. Tính  $n = pq = 17 * 11 = 187$ .

3. Tính  $\phi(n) = (p - 1)(q - 1) = 16 * 10 = 160$ .

4. Chọn  $e$  mà  $e$  là số nguyên tố cùng nhau  $\phi(n) = 160$  và nhỏ hơn  $\phi(n)$ ; ta chọn  $e = 7$ .

5. Xác định  $d$  sao cho  $de \equiv 1 \pmod{160}$  và  $d < 160$ . Giá trị đúng là  $d = 23$ , vì  $23 * 7 = 161 = (1 * 160) + 1$ ;  $d$  có thể được tính toán bằng cách sử dụng thuật toán Euclid mở rộng.

Từ đó, thu được khóa công khai PU = {7, 187} và khóa riêng PR = {23, 187}. Các ví dụ cho thấy việc sử dụng các khoá cho một đầu vào bản rõ  $M = 88$ . Đối với mã hóa, chúng ta cần phải tính toán  $C = 887 \pmod{187}$ . Khai thác các tính chất của modul số học, chúng ta có thể làm điều này như sau.

$$887 \pmod{187} = [(884 \pmod{187}) * (882 \pmod{187}) * (881 \pmod{187})] \pmod{187}$$

$$881 \pmod{187} = 88$$

$$882 \pmod{187} = 7744 \pmod{187} = 77$$

$$884 \pmod{187} = 59.969.536 \pmod{187} = 132$$

$$887 \pmod{187} = (88 * 77 * 132) \pmod{187} = 894.432 \pmod{187} = 11$$

Để giải mã, ta tính toán  $M = 1123 \pmod{187}$ :

$$1123 \pmod{187} = [(111 \pmod{187}) * (112 \pmod{187}) * (114 \pmod{187}) * (118 \pmod{187}) * (118 \pmod{187})] \pmod{187}$$

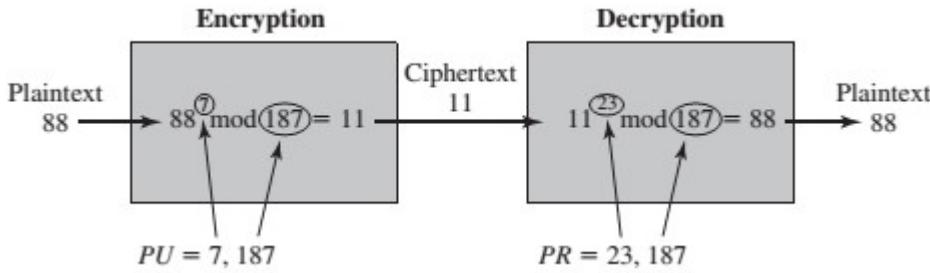
$$111 \pmod{187} = 11$$

$$112 \pmod{187} = 121$$

$$114 \pmod{187} = 14.641 \pmod{187} = 55$$

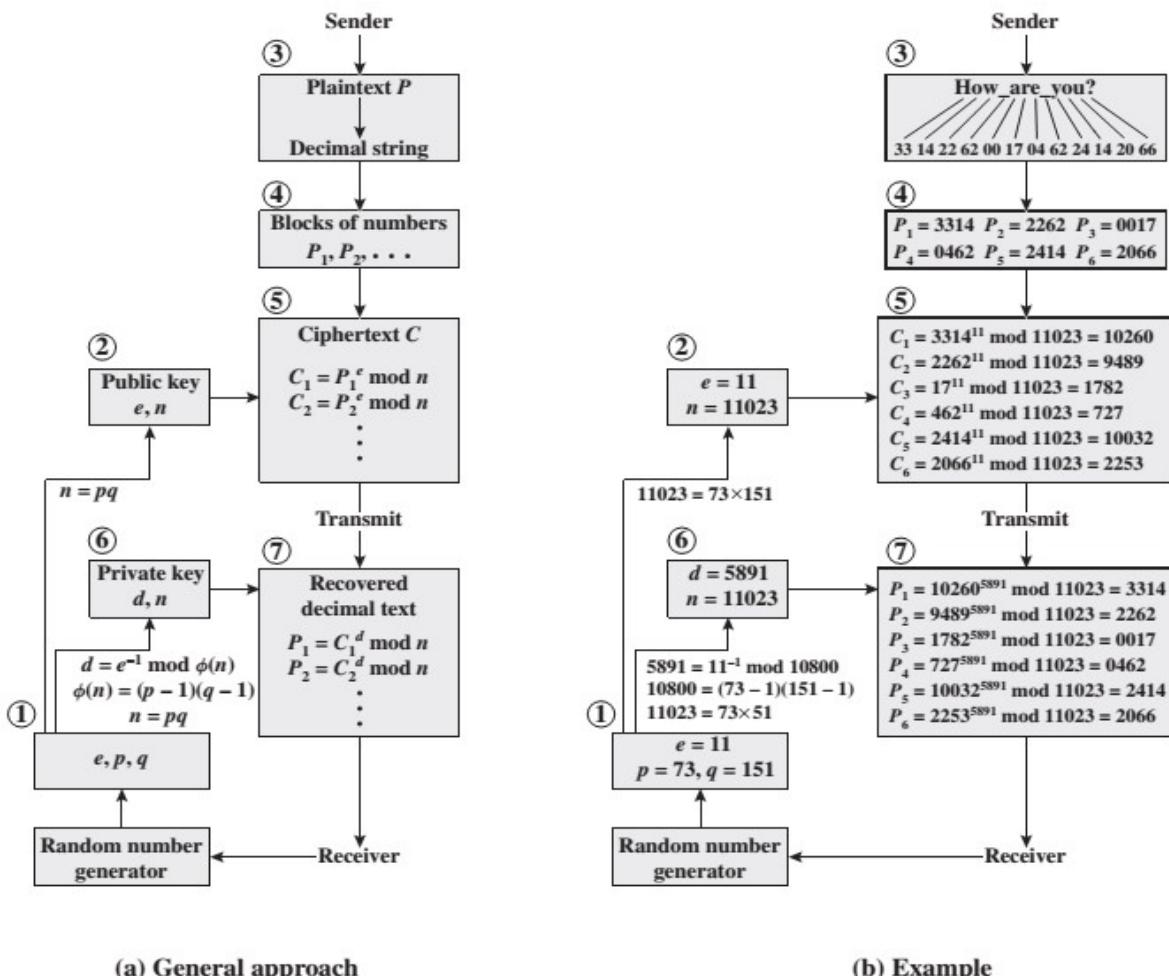
$$118 \pmod{187} = 214.358.881 \pmod{187} = 33$$

$$1123 \pmod{187} = (11 * 121 * 55 * 33 * 33) \pmod{187} = 79.720.245 \pmod{187} = 88$$



Hình 3. 6: Ví dụ tính toán của thuật toán RSA

Ta xem xét một ví dụ, trong đó RSA được sử dụng để xử lý nhiều khối dữ liệu với bản rõ là một chuỗi chữ số, mỗi ký hiệu rõ được gán một mã số duy nhất của hai số thập phân (ví dụ, A=00, A=26). Một khối bản rõ gồm 4 chữ số thập phân hoặc hai ký tự. Hình 3.7a chỉ ra dây các sự kiện cho mã hóa nhiều khối, hình 3.7b đưa ra một ví dụ đặc biệt. Các vòng tròn chỉ ra số các toán tử thực hiện.



### (i) Các khía cạnh tính toán

Vấn đề độ phức tạp tính toán cần thiết khi sử dụng RSA là một vấn đề quan trọng. Trên thực tế, có hai vấn đề cần xem xét: mã hóa / giải mã và tạo khóa.

Hàm mũ trong tính toán modulo: Cá vấn đề mã hóa và giải mã trong RSA đều liên quan từ số nguyên tới lũy thừa số nguyên, mod n. Nếu một phép tính lũy thừa được thực hiện trên số nguyên rồi giảm theo module n, thì các giá trị trung gian sẽ cực lớn. Ví dụ dưới đây sẽ sử dụng đặc tính tính toán modulo.

$$[(a \text{ mod } n) * (b \text{ mod } n)] \text{ mod } n = (a * b) \text{ mod } n$$

Như vậy, chúng ta có thể làm giảm số kết quả trung gian modulo n. Điều này làm cho việc tính toán thực tế hơn. Một khía cạnh khác là hiệu quả của hàm mũ do RSA thường đưa ra các hàm mũ lớn. Nếu xem xét việc tính  $x^{16}$  thì tiếp cận trực tiếp yêu cầu 15 phép nhân.

$$x^{16} = x * x * x * x * x * x * x * x * x * x * x * x * x * x * x * x$$

Tuy nhiên, chúng ta có thể đạt được kết quả cuối cùng cũng chỉ có bốn phép nhân nếu chúng ta nhóm các phần tạo thành ( $x^2, x^4, x^8, x^{16}$ ). Một ví dụ khác, giả sử muốn tính  $x^{11} \text{ mod } n$  cho một số các số nguyên x và n. Quan sát rằng  $x^{11} = x^{1+2+8} = (x)(x^2)(x^8)$ . Trong trường hợp này, chúng ta tính  $x \text{ mod } n, x^2 \text{ mod } n, x^4 \text{ mod } n$ , và  $x^8 \text{ mod } n$  và sau đó tính toán  $[(x \text{ mod } n) * (x^2 \text{ mod } n) * (x^8 \text{ mod } n)] \text{ mod } n$ .

Tổng quát hơn, giả sử chúng ta muốn tìm giá trị  $ab \text{ mod } n$  với  $a, b$ , và  $n$  nguyên dương. Nếu chúng ta biểu diễn  $b$  là một số nhị phân  $b_k, k-1 \dots b_0$ , sau đó có

$$b = \sum_{b_i \neq 0} 2^i$$

Vì vậy,

$$a^b = a^{\left(\sum_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^b \text{ mod } n = \left[ \prod_{b_i \neq 0} a^{(2^i)} \right] \text{ mod } n = \left( \prod_{b_i \neq 0} (a^{2^i} \text{ mod } n) \right) \text{ mod } n$$

Từ đó có thể sử dụng các thuật toán để tính toán  $ab \text{ mod } n$  như thể hiện trong hình 3.8. Bảng 3.4 đưa ra ví dụ cụ thể về việc thực hiện các thuật toán này. Lưu ý rằng biến c chỉ để dùng cho mục đích giải thích. Giá trị cuối cùng của c là giá trị của số mũ.

Bảng 3.4: Kết quả tính toán  $a^b \bmod n$  với  $a=7$ ,  $b=560=1000110000$  và  $n=561$ .

<i>i</i>	9	8	7	6	5	4	3	2	1	0
$b_i$	1	0	0	0	1	1	0	0	0	0
$c$	1	2	4	8	17	35	70	140	280	560
$f$	7	49	157	526	160	241	298	166	67	1

```

c ← 0; f ← 1
for i ← k downto 0
    do c ← 2 × c
    f ← (f × f) mod n
    if bi = 1
        then c ← c + 1
        f ← (f × a) mod n
return f

```

Hình 3.8: Thuật toán để tính  $a^b \bmod n$ 

### Hoạt động hiệu quả sử dụng khóa công khai

Để tăng tốc độ hoạt động của thuật toán RSA bằng cách sử dụng khóa công khai, ta thường lựa chọn  $e$  đặc biệt. Lựa chọn  $e$  phổ biến là 65537; hoặc  $e$  chọn là 3 và 17. Lựa chọn này chỉ cần sử dụng sử dụng hai số (1 bit) nên số lượng các phép nhân cần thiết để thực hiện các lũy thừa được giảm thiểu.

Tuy nhiên, với một khóa công khai rất nhỏ như  $e = 3$ , RSA trở nên dễ bị tấn công với một cuộc tấn công đơn giản. Giả sử ta có ba người dùng RSA khác nhau và tất cả những người sử dụng các giá trị  $e = 3$  nhưng có giá trị duy nhất của  $n$ , cụ thể là  $(n_1, n_2, n_3)$ . Nếu người dùng A gửi cùng một bản tin  $M$  mã hóa cho tất cả ba người dùng, sau đó ba bản mã là  $C_1 = M^3 \bmod n_1$ , là  $C_2 = M^3 \bmod n_2$ , là  $C_3 = M^3 \bmod n_3$ . Có khả năng là  $n_1, n_2, n_3$  và là cặp nguyên tố cùng nhau. Do đó, người ta có thể tính  $M^3 \bmod (n_1 n_2 n_3)$ . Theo các quy tắc của thuật toán RSA,  $M$  ít hơn mỗi giá trị của  $n_i$ ; do đó  $M^3 < n_1 n_2 n_3$ . Theo đó, những kẻ tấn công chỉ cần tính toán căn bậc ba của  $M^3$ . Ta lưu ý rằng định nghĩa của các thuật toán RSA yêu cầu người sử dụng chọn một giá trị của  $e$  là nguyên tố cùng nhau với  $\phi(n)$ . Như vậy, nếu một giá trị của  $e$  được lựa chọn đầu tiên và các số nguyên tố  $p$  và  $q$  được tạo ra, nó có thể chỉ ra rằng  $\gcd(\phi(n), e) \neq 1$ . Trong trường hợp đó, người sử dụng phải từ chối cặp giá trị  $p, q$  và tạo ra một cặp mới.

### **Hoạt động hiệu quả sử dụng khóa riêng**

Tương tự ta không thể lựa chọn một hằng số có giá trị nhỏ của  $d$ . Giá trị mà nhỏ sẽ dễ bị tấn công cưỡng bức và bị một số kiểu phân tích mật mã khác. Tuy nhiên có một cách để đẩy nhanh tốc độ tính toán sử dụng CRT. Ta muốn tính giá trị  $M = C^d \bmod n$ . Ta xác định các kết quả trung gian sau:

$$V_p = C^d \bmod p \quad V_q = C^d \bmod q$$

Tiếp theo CRT chỉ ra chất lượng

$$X_p = q \times (q^{-1} \bmod p) \quad X_q = p \times (p^{-1} \bmod q)$$

Sau đó CRT biểu diễn

$$M = (V_p X_p + V_q X_q) \bmod n$$

Hơn nữa ta có thể đơn giản mà tính  $V_p$  và  $V_q$  sử dụng lý thuyết Fermat với  $a^{p-1} \equiv 1 \pmod{p}$  nếu  $p$  và  $a$  quan hệ nguyên tố cùng nhau. Và ta có

$$V_p = C^d \bmod p = C^{d \bmod (p-1)} \bmod p \quad V_q = C^d \bmod q = C^{d \bmod (q-1)} \bmod q$$

Số lượng  $d \bmod (p-1)$  và  $d \bmod (q-1)$  có thể được tính toán lại. Kết quả cuối cùng việc tính toán nhanh hơn gấp xấp xỉ 4 lần so với tính  $M = C^d \bmod n$  một cách trực tiếp.

### **Tạo khóa**

Trước khi áp dụng hệ thống mật mã khóa công khai RSA, mỗi bên tham gia phải tạo ra một cặp khóa theo các bước sau:

- + Xác định 2 số nguyên tố  $p$  và  $q$
- + Lựa chọn hoặc  $e$  hoặc  $d$  và tính toán cái còn lại.

Bởi vì giá trị  $n = pq$  sẽ bị phát hiện bởi bất kỳ kẻ theo dõi tiềm năng nào, nên những số nguyên tố này phải được chọn từ tập đủ lớn tức là  $p$  và  $q$  phải là những số lớn. Một cách phương pháp sử dụng để tìm các số nguyên phải có hiệu quả hợp lý.

Hiện tại không có những kỹ thuật hữu hiệu để thu được các số nguyên tố lớn một cách tùy ý nên cần một số cách thức để xử lý vấn đề này. Thủ tục là nhặt ngẫu nhiên một số lẻ nằm trong độ lớn mong muốn và kiểm tra xem nó có phải là số nguyên tố không. Nếu không phải nhặt lại các số ngẫu nhiên khác cho đến khi nó là số nguyên tố. Thủ tục để kiểm tra xem một số nguyên cho trước  $n$  nào đó có phải là số nguyên tố là thực hiện một số tính toán liên quan đến  $n$  và một số nguyên ngẫu nhiên  $a$ . Nếu  $n$  “không vượt

qua” phép thử thì  $n$  không phải là số nguyên tố. Nếu  $n$  “vượt qua” phép thử thì  $n$  có thể là số nguyên tố hoặc là không. Nếu  $n$  vượt qua một số lượng phép thử các giá trị  $a$  khác nhau được lựa chọn ngẫu nhiên thì ta có thể tin chắc  $n$  là số nguyên tố.

Nhìn chung thủ tục lấy ra một số nguyên tố là:

1. Lấy ngẫu nhiên một số nguyên lẻ (ví dụ sử dụng bộ tạo số giả ngẫu nhiên).
2. Lấy ngẫu nhiên một số nguyên  $a < n$ .
3. Thực hiện phép thử xác suất với  $a$  là một tham số. Nếu  $n$  không vượt qua phép thử, loại bỏ  $n$  và quay về bước 1.
4. Nếu  $n$  vượt qua đủ số lượng phép thử chấp nhận  $n$ . Một khác chuyển sang bước 2.

Tuy nhiên tiến trình được thực hiện không thường xuyên mà chỉ khi cần một cặp ( $PU, PR$ ) mới.

Cũng cần phải lưu ý là sẽ có bao nhiêu số bị loại bỏ trước khi một số nguyên tố được tìm ra. Lý thuyết số nguyên tố đã chỉ ra, số các số nguyên tố ở lân cận  $N$  trung bình là một trong  $\ln(N)$  số nguyên. Như vậy ta có thể phải kiểm tra đến  $\ln(N)$  số nguyên trước khi tìm ra một số nguyên tố. Vì các số nguyên chẵn sẽ bị loại ngay nên chính xác là sẽ phải kiểm tra  $\ln(N/2)$  số. Ví dụ, nếu muốn tìm một số nguyên tố ở khoảng 2200 thì phải thử khoảng  $\ln(2200/2)=70$  lần mới tìm ra.

Sau khi có được số nguyên tố  $p$  và  $q$  thì tiếp theo là lựa chọn giá trị  $e$  và tính toán  $d$  hoặc ngược lại lựa chọn  $d$  và tính toán  $e$ . Nếu là trường hợp trước thì ta cần chọn một số  $e$  sao cho  $\text{gdc}(\phi(n), e) = 1$  và tính toán  $d \equiv e^{-1}(\text{mod } \phi(n))$ . Đây là một thuật toán đơn nên cùng lúc nó sẽ tính toán ước số chung lớn nhất của 2 số nguyên và nếu gcd là 1 thì xác định được nghịch đảo của một số nguyên modulo số còn lại. Như vậy thủ tục là tạo ra một chuỗi số ngẫu nhiên, kiểm tra chúng lần lượt với  $\phi(n)$  cho đến khi một số nguyên tố tương quan với  $\phi(n)$  được tìm ra.

## (ii) Vấn đề bảo mật của RSA

Có 5 khả năng để tấn công thuật toán RSA:

- + Tấn công cưỡng bức: Cố gắng bằng mọi khả năng tìm ra khóa riêng.
- + Tấn công toán học: Nỗ lực tìm ra tích của 2 số nguyên tố.

- + Tấn công thời gian: Phụ thuộc vào thời gian hoạt động của thuật toán giải mã.
- + Tấn công dựa trên lỗi phần cứng: Gây ra các lỗi phần cứng trong bộ xử lý tạo ra các chữ ký số.
- + Tấn công vào bản mã: Kiểu tấn công này khai thác các đặc tính của thuật toán RSA.

Việc chống lại tấn công cưỡng bức thì đối với thuật toán RSA hay các hệ thống mật mã khác đều bằng cách sử dụng khóa đủ lớn. Tức là số lượng bit trong  $d$  càng lớn càng tốt. Tuy nhiên việc tính toán trong cả việc tạo khóa và việc mã hóa/giải mã đều phức tạp nên kích thước khóa càng lớn thì hệ thống hoạt động càng chậm.

*Bài toán tìm thừa số* là tiếp cận tấn công toán học vào RSA

1. Tìm thừa số  $n$  từ 2 thừa số nguyên tố của nó. Điều này cho phép tính toán được  $\phi(n) = (p-1)(q-1)$  và từ đó xác định được  $d \equiv e^{-1}(\text{mod } \phi(n))$ .
2. Xác định một cách trực tiếp  $\phi(n)$  mà không cần xác định  $p$  và  $q$ . Một lần nữa nó cũng cho phép tính được  $d \equiv e^{-1}(\text{mod } \phi(n))$ .
3. Tìm ra  $d$  một cách trực tiếp mà không cần xác định  $\phi(n)$

Các hướng giải mật mã RSA đều tập trung vào nhiệm vụ tìm ra  $n$  từ 2 thừa số nguyên tố. Với thuật toán đã biết việc xác định với  $e$  và  $n$  đã cho ít nhất cũng tiêu tốn thời gian như bài toán phân tích thừa số. Vì vậy chúng ta có thể dùng cách này như một điểm chót trong tính bảo mật của RSA.

Nếu  $n$  lớn thì việc phân tích ra nó là rất khó khăn. Năm 1977, ba nhà phát minh ra RSA đã thách thức các độc giả của tạp chí khoa học Mỹ giải mật mã một bản mã. Họ sẽ trả 100USD tiền thưởng cho một câu trong bản rõ mà họ dự đoán rằng việc này phải làm trong 40 nghìn triệu năm. Tháng 4 năm 1994, một nhóm làm việc qua Internet đã nhận giải chỉ sau 8 tháng làm việc. Thách thức này sử dụng kích thước khóa công khai (độ dài của  $n$ ) là 129 chữ số thập phân hoặc khoảng 428 bít. Trong lúc đó chỉ cần thực hiện đối với DES, các phòng thí nghiệm RSA đã đưa ra các thách thức cho mật mã RSA với kích thước khóa là 100, 110, 120... Thách thức cuối cùng được giải là RSA-768 với độ dài khoa là 232 số thập phân hoặc 768 bit. Bảng 3.5 chỉ ra kết quả về mặt thời gian. Bộ xử lý hàng triệu phép tính/giây chạy trong 1 năm tức là khoảng  $3 \times 10^{13}$  phép tính được thực

hiện. Cho đến giữa những năm 1990 việc tấn công tìm thừa số được thực hiện với tiếp cận sàng lọc bậc hai. Việc tấn công vào RSA-130 sử dụng một thuật toán mới GNFS cho phép tìm được thừa số của số lớn hơn RSA-129 mà chỉ mất 20% nỗ lực tính toán.

Bảng 3.3: Tiến trình tìm ra thừa số trong RSA

Number of Decimal Digits	Number of Bits	Date Achieved
100	332	April 1991
110	365	April 1992
120	398	June 1993
129	428	April 1994
130	431	April 1996
140	465	February 1999
155	512	August 1999
160	530	April 2003
174	576	December 2003
200	663	May 2005
193	640	November 2005
232	768	December 2009

Việc đe dọa vào kích thước khóa lớn đã tăng lên gấp đôi bởi năng lực tính toán của máy tính đang ngày càng tăng và các thuật toán tìm thừa số tiếp tục được hoàn chỉnh. Như vậy chúng ta cần cẩn thận trong việc lựa chọn kích thước khóa cho RSA.

Ngoài việc xác định kích thước của  $n$ , một số các ràng buộc khác cũng được gợi ý bởi các nhà nghiên cứu. Để tránh một số giá trị của  $n$  mà có thể tìm ra một cách dễ dàng các nhà phát minh thuật toán đưa ra các ràng buộc của  $p$  và  $q$ .

1.  $p$  và  $q$  nên khác nhau về độ dài trong vài con số. Như vậy đối với khóa 1024-bit (309 số thập phân) cả  $p$  và  $q$  có thể ở độ lớn tầm  $10^{75}$  đến  $10^{100}$
2. Cả  $(p-1)$  và  $(q-1)$  đều là số nguyên tố lớn.
3.  $\text{Gcd}(p-1, q-1)$  có thể nhỏ.

Thêm nữa cần phải nhấn mạnh rằng nếu  $e < n$  và  $d < n^{1/4}$  thì  $d$  có thể dễ dàng được xác định.

### Tấn công thời gian

Tấn công thời gian cũng thể hiện được sự khó khăn để phá vỡ tính an toàn của thuật toán mật mã hóa. Paul Kocher, một chuyên gia mật mã hóa, đã nhấn mạnh rằng một kẻ đi rình có thể xác định được khóa riêng biệt bằng việc theo dõi thời gian một máy tính đưa ra bản giải mã. Tấn công thời gian không chỉ áp dụng cho RSA mà cho cả các hệ thống

mật mã khóa công khai khác. Việc tấn công này có thể đến từ bất kỳ hướng nào và chỉ tấn công vào bản mã.

Tấn công thời gian được giải thích sử dụng thuật toán hàm mũ modulo như trên hình 3.8 nhưng sự tấn công này có thể thực hiện ở bất kỳ giai đoạn nào chứ không phải trong thời gian cố định. Trong thuật toán này việc mũ hóa modulo được thực hiện từng bit với một tích modular được thực hiện tại mỗi vòng lặp và thêm một tích modular được thực hiện tại mỗi bit.

Gia sử mục tiêu của hệ thống là dùng một hàm tích modular hoạt động rất nhanh trong phần lớn các trường hợp nhưng trong đôi khi lại mất rất nhiều thời gian hơn là dùng hàm mũ modular trung bình toàn cục. Tiến trình tấn công tấn công theo từng bit bắt đầu với bit bên trái  $bk$ . Trước hết các bit  $j$  đã được biết (để có được toàn bộ số mũ ta bắt đầu với  $j=0$  và lặp lại việc tấn công cho đến khi toàn bộ số mũ được biết). Đôi với một bản mã cho trước kẻ tấn công có thể hoàn thành trước hết các bước lặp  $j$  của vòng lặp for. Hoạt động của bước tiếp theo phụ thuộc vào bit số mũ chưa biết. Nếu bit này được thiết lập thì phép  $d \leftarrow (d \times a) \text{ mod } n$  sẽ được thực hiện. Đôi với một giá trị mới của  $a$  và  $d$ , tích modular sẽ thực hiện chậm và kẻ tấn công biết được. Vì vậy nếu quan sát thấy thời gian thực hiện thuật toán giải mã là luôn chậm thì bước lặp đặc thù chậm là với bit 1 nên bit này được giả thiết là 1. Nếu một bước thực hiện nào đó trong thuật toán là nhanh thì bit này được giả thiết là 0.

Trong thực tế việc thực hiện mũ hóa modular không có biến động nhiều về thời gian. Tuy nhiên nó lại biến động đủ để thực hiện được việc tấn công.

Mặc dù tấn công thời gian rất nguy hiểm nhưng cũng đã có một số phương pháp đối phó được sử dụng:

- + Thời gian mũ hóa là hằng số: Đảm bảo việc mũ hóa được thực hiện hết cùng một khoảng thời gian. Việc này là đơn giản nhưng lại bị giảm hiệu suất.
- + Trễ ngẫu nhiên: Thực hiện thêm một độ trễ ngẫu nhiên vào thuật toán mũ hóa để làm rối loạn tấn công thời gian. Kocher chỉ ra rằng nếu các biện pháp phòng chống không đưa vào đủ nhiều thì kẻ tấn công vẫn thành công trong việc thu thập các mẫu đo bù vào cho các trễ ngẫu nhiên.
- + Làm đầy: Nhân bản mã với một số ngẫu nhiên trước khi thực hiện mũ hóa. Tiến trình này ngăn kẻ tấn công biết được các bit của bản mã được xử lý ở

trong máy tính và từ đó ngăn được việc thực hiện phân tích thời gian theo từng bit của tấn công thời gian.

Bảo mật dữ liệu RSA kết hợp đặc tính làm đầy này trong một số sản phẩm của mình. Khóa riêng biệt  $M=Cd \text{ mod } n$  được thực hiện như sau:

1. Tạo ra một số ngẫu nhiên bí mật  $r$  nằm giữa 0 và  $n-1$ .
2. Tính toán  $C'=C(re) \text{ mod } n$ , với  $e$  là số mũ công khai.
3. Tính toán  $M'=(C')^d \text{ mod } n$ , với các bước RSA gốc.
4. Tính toán  $M=M'r^{-1} \text{ mod } n$ . Trong công thức này  $r^{-1}$  là tích nghịch đảo của  $r$  mod  $n$ . Đây là kết quả chính xác bởi  $r^{-1} \text{ mod } n = r^{-1} \text{ mod } n$ .

**Tấn công dựa trên lỗi:** Một tiếp cận tấn công RSA không chính thống nữa là tấn công vào bộ xử lý tạo ra chữ ký số RSA. Việc tấn công này gây ra lỗi trong việc tính toán chữ ký số bằng việc giảm năng lực của bộ vi xử lý. Các lỗi gây ra bởi phần mềm tạo ra các chữ ký không đủ năng lực có thể bị phân tích bởi kẻ tấn công nhằm tìm ra khóa riêng.

Thuật toán tấn công gây ra các lỗi bit đơn và bắt lấy kết quả. Kiểu tấn công này có vẻ không gây nguy hiểm nghiêm trọng cho RSA. Nó yêu cầu kẻ tấn công phải truy nhập vật lý đến thiết bị và kẻ tấn công phải có khả năng điều khiển trực tiếp công suất đầu vào của vi xử lý.

### **Tấn công bản mã lựa chọn và đệm mật mã bất đối xứng tối ưu.**

Thuật toán RSA cơ sở có thể bị tấn công gọi là tấn công bản mã lựa chọn (CCA). CCA được định nghĩa là một kẻ tấn công chọn ra một số bản mã và sau đó tìm ra bản rõ tương ứng, giải mã với khóa riêng của đích. Như vậy kẻ quan sát có thể lựa chọn một bản rõ, mật mã hóa với khóa công khai của đích và sau đó lại có khả năng có bản rõ bằng việc giải mã với khóa riêng. Rõ ràng điều này không cung cấp cho kẻ theo dõi thông tin mới. Thay vào đó kẻ theo dõi khám phá ra đặc tính của RSA và lựa chọn các khối dữ liệu và trong tiến trình xử lý với mã riêng của đích thì sẽ thu thập được thông tin cần thiết cho việc phân tích mật mã.

CCA tấn công RSA dựa trên đặc tính của RSA:

$$E(PU, M_1) \times E(PU, M_2) = E(PU, [M_1 \times M_2]) \quad (3.2)$$

Ta mã giải mã  $C=M^e \text{ mod } n$  sử dụng CCA như sau:

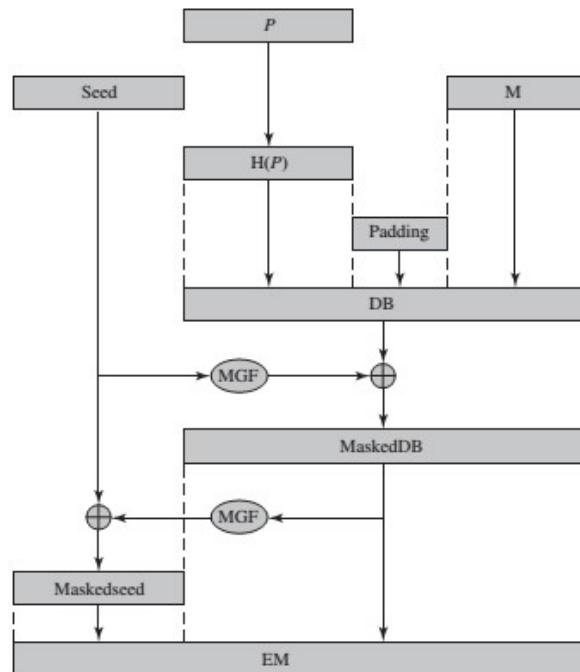
1. Tính toán  $X = (C \times 2^e) \text{ mod } n$ .

2. Để xuất X như một bản mã và nhận về  $Y = X^d \text{ mod } n$ .

Lưu ý là

$$\begin{aligned} X &= (C \text{ mod } n) \times (2^e \text{ mod } n) \\ &= (M^e \text{ mod } n) \times (2^e \text{ mod } n) \\ &= (2M)^e \text{ mod } n \end{aligned}$$

Như vậy  $Y = (2M) \text{ mod } n$ . Từ đây ta có thể suy ra M. Để giải quyết việc tấn công này trong thực tế các hệ thống dựa trên RSA ngẫu nhiên đệm thêm vào bản rõ trước khi mật mã. Việc này dẫn đến công thức 3.2 không còn giữ được nữa. Tuy nhiên các phép CCA tinh vi thì vẫn có thể thực hiện được và việc chèn thêm một giá trị ngẫu nhiên đơn giản không đủ để cung cấp tính bảo toàn như mong muốn. Để chống lại tấn công này ta thay đổi bản rõ sử dụng thủ tục đệm mật mã bất đối xứng tối ưu (OAEP). Hình vẽ 3.9 chỉ ra mã hóa OAEP.



P: Các tham số mã hóa

DB: Khối dữ liệu

M: Bản tin đưa vào mã hóa

MGF: Hàm tạo mặt nạ

H: Hàm băm

EM: Bản tin đã được mã hóa

Hình 3. 9: Mã hóa sử dụng đệm mật mã bất đối xứng tối ưu (OAEP)

Tại bước đầu tiên bản tin M đưa vào mã hóa được đệm thêm. Một tập các tham số tùy chọn P được đưa qua hàm băm H. Đầu ra được đệm thêm với các số không để có độ dài mong muốn trong toàn khối dữ liệu. Tiếp theo một mầm ngẫu nhiên được tạo ra chuyển đến một hàm băm khác gọi là hàm tạo mặt nạ MGF. Kết quả giá trị băm là theo từng bít được thực hiện phép XOR với DB để tạo ra DB đã được che mặt nạ. DB được phủ mặt nạ được gửi qua MGF tạo ra một hàm băm mà thực hiện phép toán XOR với mầm để tạo ra mầm đã được che. Sự móc nối của mầm được che mặt nạ và khối dữ liệu được che mặt nạ tạo ra bản tin đã được mã hóa EM. Lưu ý rằng EM gồm bản tin đã được đệm, mặt nạ tạo ra bởi mầm, mầm, mặt nạ bởi dữ liệu đã được che. EM này sau đó được mật mã hóa sử dụng RSA.

### 3.2 Trao đổi khóa Diffie-Hellman

Thuật toán khóa công khai đầu tiên được xuất hiện trong báo cáo của Diffie và Hellman được gọi là mật mã khóa công khai và thường được nhắc đến là trao đổi khóa Diffie-Hellman. Một loạt các sản phẩm thương mại đã sử dụng kỹ thuật trao đổi khóa này.

Mục đích của thuật toán là cho phép 2 người dùng trao đổi một cách an toàn một khóa mà sau đó có thể được dùng để mã hóa đối xứng các bản tin. Bản thân thuật toán đơn giản hóa là sự trao đổi những giá trị bí mật.

Hiệu quả của thuật toán Diffie-Hellman là dựa vào những trở ngại trong việc tính toán của các thuật toán rời rạc. Nói một cách ngắn gọn, chúng ta có thể định nghĩa thuật toán rời rạc theo cách sau đây. Gốc ban đầu của một số nguyên tố  $p$  là một số mà lũy thừa của nó modulo  $p$  tạo ra tất cả các số nguyên từ 1 đến  $p-1$ . Có nghĩa là nếu  $a$  là một gốc ban đầu của một số nguyên tố  $p$  thì các số

$a \text{ mod } p, a^2 \text{ mod } p, \dots, a^{p-1} \text{ mod } p$  là các số nguyên phân biệt từ 1 đến  $p-1$ .

Với bất kỳ số nguyên  $b$  và một gốc ban đầu  $a$  của số nguyên tố  $p$ , ta có thể tìm ra một số mũ duy nhất  $i$  sao cho:

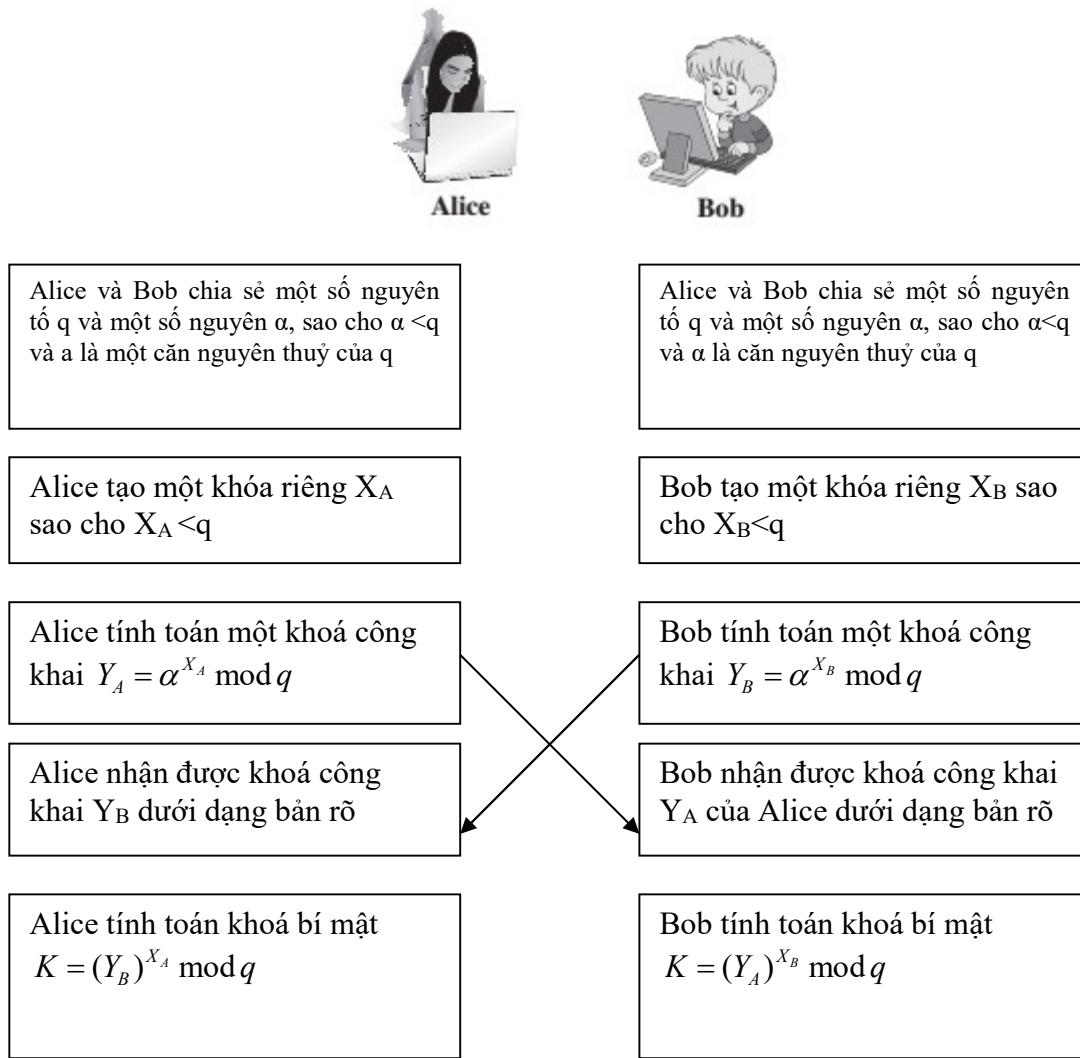
$$b \equiv a^i \pmod{p} \text{ với } 0 \leq i \leq (p-1)$$

Số mũ  $i$  được gọi là thuật toán rời rạc của  $b$  với căn nguyên thủy  $a \text{ mod } p$ .

Chúng ta thể hiện giá trị này là  $\text{dlog}_a p(b)$ .

**Thuật toán:**

Hình 3.10 tóm lược thuật toán trao đổi khóa Diffie-Hellman. Trong trường hợp này có 2 số đã biết: một số nguyên tố  $q$  và một số nguyên  $\alpha$  là căn nguyên thuỷ của  $q$ . Gia sú người dùng A và B muốn tạo ra một khóa chia sẻ.



Hình 3. 10: Trao đổi khóa Diffie-Hellman

Người dùng A chọn một số nguyên ngẫu nhiên  $X_A < q$  và tính toán  $Y_A = \alpha^{X_A} \text{ mod } q$ . Tương tự người dùng B độc lập lựa chọn một số nguyên  $X_B < q$  và tính toán  $Y_B = \alpha^{X_B} \text{ mod } q$ . Mỗi bên giữ giá trị X của mình và tạo ra giá trị Y công khai với bên kia. Như vậy  $X_A$  là khóa riêng của A và  $Y_A$  tương ứng là khóa công khai. Tương tự như thế đối với B. Người dùng A tính toán khóa là  $K = (Y_B)^{X_A} \text{ mod } q$  và người dùng B tính toán khóa là  $K = (Y_A)^{X_B} \text{ mod } q$ . Cả hai tính toán này tạo ra các kết quả độc lập:

$$\begin{aligned}
K &= (Y_B)^{X_A} \bmod q \\
&= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\
&= (\alpha^{X_B})^{X_A} \bmod q \\
&= \alpha^{X_B X_A} \bmod q && \text{theo các luật của thuật toán modulo} \\
&= (\alpha^{X_A})^{X_B} \bmod q \\
&= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\
&= (Y_A)^{X_B} \bmod q
\end{aligned}$$

Kết quả là hai bên đã trao đổi một giá trị bí mật. Giá trị bí mật này được dùng như là một khóa bí mật đối xứng được chia sẻ. Nếu một ai nào đó bắt được quá trình trao đổi khóa và muốn tìm khóa bí mật K thì vì  $X_A$  và  $X_B$  được giữ riêng, kể theo dõi chỉ có thể biết được các thành phần  $q$ ,  $\alpha$ ,  $Y_A$  và  $Y_B$ . Như vậy kể theo dõi buộc phải thực hiện một thuật toán rắc rối để xác định khóa. Ví dụ, để xác định khóa giữ riêng của người dùng B kể theo dõi phải tính toán

$$X_B = d \log_{\alpha,q}(Y_B)$$

Người theo dõi có thể tính toán khóa K theo như cách người dùng B tính toán. Như vậy kể theo dõi sẽ tính được K là

$$K = (Y_A)^{X_B} \bmod q$$

Tính an toàn của trao đổi khóa Diffie-Hellman dựa vào thực tế là nó có vẻ dễ dàng tính được modulo số mũ của một số nguyên nhưng lại rất khó để tính được các logarit rắc rối. Đối với các số nguyên lớn thì nhiệm vụ này có thể coi như bất khả thi.

Ví dụ: Khóa trao đổi dựa trên số nguyên 353 và căn nguyên thủy của nó trong trường hợp này  $\alpha = 3$ . Avà B lựa chọn những khóa riêng là  $X_A=97$  và  $X_B=233$ . Việc tính ra khóa công khai sẽ là:

A tính được  $Y_A = 3^{97} \bmod 353 = 40$ .

B tính được  $Y_B = 3^{233} \bmod 353 = 248$ .

Sau khi chúng trao đổi khóa công khai, mỗi bên sẽ tính được khóa bí mật chung là:

A tính được  $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$

B tính được  $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$

Chúng ta giả sử một kẻ tấn công có thể có được các thông tin:

$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$

Trong ví dụ đơn giản này có thể xác định được khóa bí mật là 160. Cụ thể là kẻ tấn công E có thể xác định được khóa chung bằng việc khám phá được công thức  $3^a \bmod 353 = 40$  hoặc  $3^b \bmod 353 = 248$ . Việc cưỡng chế ở đây là tính toán ra được số mũ của 3 modulo 353 ra 40 hoặc 248. Câu trả lời mong muốn là giá trị mũ bằng 97 bởi  $3^{97} \bmod 353 = 40$ . Với các số lớn hơn, vấn đề sẽ trở nên không thực tế.

### Các giao thức trao đổi khóa:

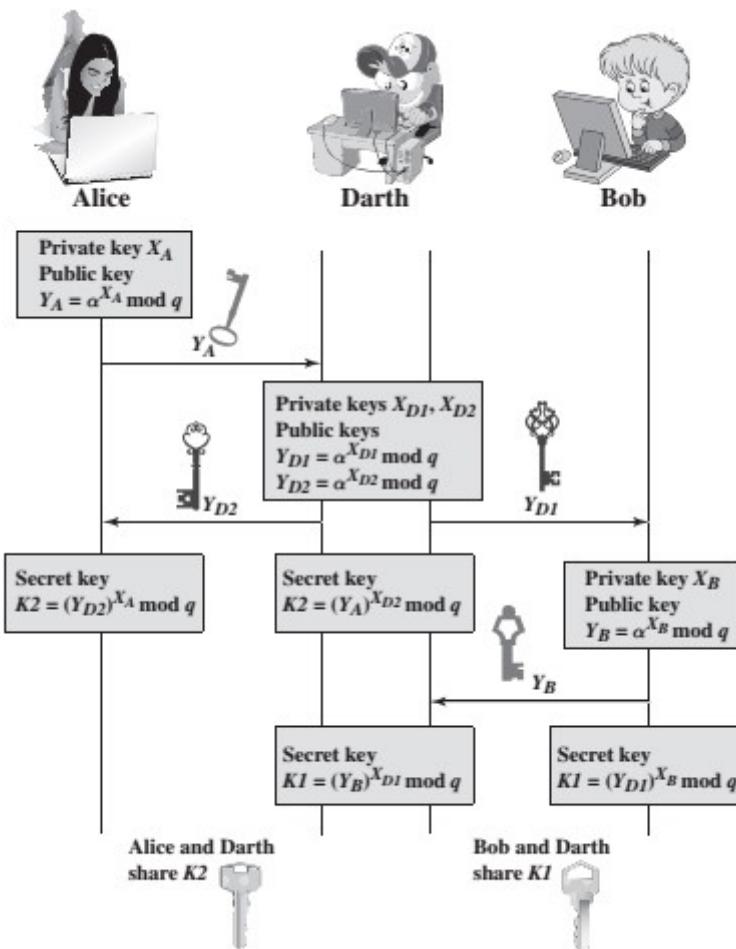
Hình 3.10 chỉ ra một giao thức đơn giản sử dụng tính toán Diffie-Hellman. Giả sử rằng người dùng A muốn thiết lập một kết nối với người dùng B và sử dụng một khóa bí mật để mã hóa các bản tin trên kết nối đó. Người dùng A có thể tạo ra khóa riêng sử dụng một lần  $X_A$ , tính toán  $Y_A$  và gửi nó cho người dùng B. Người dùng B đáp lại bằng cách tạo ra một giá trị riêng  $X_B$ , tính toán ra  $Y_B$  và gửi  $Y_B$  tới A. Lúc này cả 2 người dùng đều có thể tính toán ra khóa. Các giá trị công khai  $q$  và  $\alpha$  cần được biết trước đó có thể ngay trong bản tin đầu tiên.

Một ví dụ khác trong việc sử dụng thuật toán Diffie-Hellman là mỗi người trong một nhóm các người dùng (người dùng trong một mạng LAN) tạo ra một giá trị riêng lâu dài  $X_i$  (cho người dùng  $i$ ) và tính toán ra giá trị công khai  $Y_i$ . Các khóa công khai này kết hợp với giá trị công khai toàn cục  $q$  và  $\alpha$  được lưu trữ ở trung tâm. Bất kỳ lúc nào người dùng  $j$  cũng có thể truy nhập để có được khóa công khai của  $i$ , tính toán ra một khóa bí mật và sử dụng nó để mật mã các bản tin gửi đến người dùng A. Nếu như nơi lưu trữ trung tâm đáng tin cậy thì kiểu trao đổi thông tin này cung cấp cả tính bảo mật và tính xác thực. Bởi vì chỉ có  $i$  và  $j$  mới có thể xác định được khóa, không người dùng nào khác có thể đọc được bản tin (tính bảo mật). Người nhận  $i$  biết rằng chỉ người dùng  $j$  mới có thể tạo ra một bản tin sử dụng khóa này (tính xác thực). Tuy nhiên kỹ thuật này không chống lại được kẻ tấn công phát lại.

### Kẻ tấn công ở giữa

Giao thức chỉ ra trên hình 3.10 không an toàn để chống lại kẻ tấn công ở giữa. Giả sử Alice và Bob muốn trao đổi khóa và Darth là kẻ theo dõi. Việc tấn công diễn ra như sau: (Hình 3.11)

1. Darth chuẩn bị cho quá trình tấn công bằng việc tạo ra 2 khóa ngẫu nhiên  $X_{D1}$  và  $X_{D2}$ . Sau đó tính toán ra khóa công khai tương ứng là  $Y_{D1}$  và  $Y_{D2}$ .
2. Alice gửi  $Y_A$  sang cho Bob.
3. Darth xen vào nhận  $Y_A$  và gửi  $Y_{D1}$  cho Bob. Darth cũng tính toán  $K2 = (Y_A)^{X_{D2}} \text{ mod } q$ .
4. Bob nhận được  $Y_{D1}$  và tính toán  $K1 = (Y_{D1})^{X_B} \text{ mod } q$ .
5. Bob gửi  $Y_B$  cho Alice.
6. Darth xen vào nhận  $Y_B$  và gửi  $Y_{D2}$  cho Alice. Darth tính toán  $K1 = (Y_B)^{X_{D1}} \text{ mod } q$ .
7. Alice nhận  $Y_{D2}$  và tính toán  $K2 = (Y_{D2})^{X_A} \text{ mod } q$



Hình 3. 11: Tấn công ở giữa.

Ở đây, Bob và Alice đều nghĩ rằng họ chia sẻ một khóa bí mật nhưng thay vào đó là Bob và Darth chia sẻ khóa bí mật  $K_1$ , Darth và Alice chia sẻ khóa bí mật  $K_2$ . Tất cả những thông tin trao đổi giữa Bob và Alice đều được thỏa hiệp theo cách sau:

1. Alice gửi bản tin đã được mật mã hóa  $M:E(K_2, M)$ .
2. Darth xen vào nhận bản tin đã được mật mã và giải mã nó thành  $M$
3. Darth gửi cho Bob bản  $E(K_1, M)$  hoặc  $E(K_1, M')$  mà  $M'$  là bất kỳ bản tin nào. Trong trường hợp đầu tiên Darth chỉ đơn giản muốn nghe trộm thông tin mà không thay đổi chúng. Trường hợp sau Darth muốn làm sai lệch thông tin gửi đến Bob.

Giao thức trao đổi khóa có thể bị tấn công bởi vì nó không có sự nhận thực các bên tham gia. Ta có thể giải quyết vấn đề này bằng việc sử dụng chữ ký số và chứng nhận khóa công khai. Những vấn đề đó sẽ được trình bày ở những chương sau.

### 3.3 Hệ thống mật mã Elgamal

Vào năm 1984, T.Elgamal đã công bố một hệ thống mã công khai dựa trên logarit rời rạc, rất gần với kĩ thuật Diffie-Hellman. Hệ mật mã Elgamal được dùng dưới nhiều dạng trong một số các chuẩn, bao gồm chuẩn chữ ký số (DSS) và chuẩn S/MIME e-mail.

Giống như mã Diffie-Hellman, các yếu tố cơ bản của mật mã Elgamal là số nguyên tố  $q$  và căn nguyên thuỷ  $\alpha$  của  $q$ . Người dùng A tạo một khóa cá nhân/công khai như sau:

1. Tạo một số nguyên  $X_A$  bất kì, sao cho  $1 < X_A < q - 1$ .
2. Tính toán  $Y^A = \alpha^{X_A} \text{ mod } q$
3. Khóa cá nhân của A là  $X_A$  và khóa công khai của A là  $\{q, \alpha, Y_A\}$ .

Bất kì người dùng B nào có quyền truy cập vào khóa công khai của A có thể mã hoá một bản tin như sau:

1. Biểu diễn bản tin dưới dạng một số nguyên  $M$  thuộc khoảng  $0 \leq M \leq q - 1$ . Các bản tin dài hơn được gửi như một chuỗi các khối mà trong đó mỗi khối là một số nguyên nhỏ hơn  $q$ .
2. Chọn một số nguyên bất kì  $k$  sao cho  $1 \leq k \leq q - 1$ .
3. Tính khoá dùng một lần  $K = (Y_A)k \text{ mod } q$ .
4. Mã hoá  $M$  như một cặp số nguyên  $(C_1, C_2)$  sao cho  $C_1 = \alpha k \text{ mod } q$ ;  $C_2 = K_M \text{ mod } q$

Người dùng A khôi phục bản rõ như sau

1. Khôi phục khoá bằng cách tính  $K = (C_1)^{X_A} \text{ mod } q$
2. Tính  $M = (C_2 K^{-1}) \text{ mod } q$ .

Những bước này được tổng quát trong Hình 3.12.

Chúng ta sẽ giải thích hoạt động của hệ thống mật mã Elgamal. Đầu tiên chúng ta sẽ trình bày cách mà K được khôi phục bởi tiến trình giải mã:

$K = (Y_A)^k \text{ mod } q$	K được xác định trong suốt tiến trình mã hoá
$K = (\alpha^{X_A} \text{ mod } q)^k \text{ mod } q$	thay thế bằng cách dùng $Y_A = \alpha^{X_A} \text{ mod } q$
$K = \alpha^{kX_A} \text{ mod } q$	bằng các luật của phép số học modulo
$K = (C_1)^{X_A} \text{ mod } q$	thay thế bằng cách dùng $C_1 = \alpha^k \text{ mod } q$

Tiếp theo, bằng cách dùng K, ta khôi phục bản rõ:

$$C_2 = KM \text{ mod } q$$

$$(C_2 K^{-1}) \text{ mod } q = KMK^{-1} \text{ mod } q = M \text{ mod } q = M$$

### Các yếu tố công khai toàn cục

q	Số nguyên tố
$\alpha$	$\alpha < q$ và $\alpha$ là một căn nguyên thuỷ của q

### Khóa tạo ra bởi Alice

Lựa chọn khóa $X_A$ cá nhân	$X_A < q - 1$
-----------------------------	---------------

Tính toán $Y_A$	$Y_A = \alpha^{X_A} \text{ mod } q$
-----------------	-------------------------------------

Khóa công khai	$\{q, \alpha, Y_A\}$
----------------	----------------------

Khóa cá nhân	$X_A$
--------------	-------

### Việc mật mã hoá thực hiện bởi Bob với mã công khai của Alice

Bản rõ	$M < q$
Chọn số nguyên k bất kì	$k < q$
Tính K	$K = (Y_A)^k \text{ mod } q$
Tính $C_1$	$C_1 = \alpha^k \text{ mod } q$
Tính $C_2$	$C_2 = KM \text{ mod } q$
Bản mã	$(C_1, C_2)$

### Giải mã thực hiện bởi Alice với mã cá nhân của Alice

Bản mã	$(C_1, C_2)$
Tính K	$K = (C_1)^{X_A} \text{ mod } q$
Bản rõ	$M = (C_2 K^{-1}) \text{ mod } q$

Hình 3.12: Hệ thống mật mã Elgamal

Chúng ta có thể giải thích lại hệ thống Elgamal như sau, sử dụng Hình 3.12

1. Bob lựa chọn một số nguyên bất kì  $k$
2. Bob tạo một khoá dùng một lần  $K$  bằng cách sử dụng các yếu tố của mã công khai của Alice là  $Y_A, q$  và  $k$
3. Bob mã hoá k sử dụng yếu tố của mã công khai là  $\alpha$ , tạo ra  $C_1$ .  $C_1$  cung cấp lượng thông tin đầy đủ cho Alice để khôi phục  $K$
4. Bob mã hoá bản rõ  $M$  sử dụng  $K$
5. Alice khôi phục  $K$  từ  $C_1$  bằng cách dùng mã cá nhân
6. Alice sử dụng  $K^{-1}$  để phục hồi bản rõ từ  $C_2$ .

Thêm vào đó,  $K$  hoạt động như một chiếc khoá dùng một lần, được sử dụng để mã hoá và giải mã bản tin.

Ví dụ, hãy bắt đầu với trường số nguyên tố GF(19); tức là  $q = 19$ . Nó có căn nguyên thuỷ  $\{2,3,10,13,14,15\}$ . Ta chọn  $\alpha = 10$ .

Alice tạo ra cặp mã như sau:

1. Alice chọn  $X_A = 5$ .
2. Thì  $Y_A = \alpha^{X_A} \text{ mod } q = \alpha^5 \text{ mod } 19 = 3$
3. Mã cá nhân của Alice là 5 và mã công khai của Alice là  $\{q, \alpha, Y_A\} = \{19, 10, 3\}$ .

Giả sử Bob muốn gửi bản tin của mình với giá trị  $M = 17$  thì:

1. Bob chọn  $k = 6$
2. Sau đó  $K = (Y_A)k \text{ mod } q = 36 \text{ mod } 19 = 729 \text{ mod } 19 = 7$
3. Vì vậy
4.  $C_1 = \alpha k \text{ mod } q = \alpha 6 \text{ mod } 19 = 11$
5.  $C_2 = KM \text{ mod } q = 7 \times 17 \text{ mod } 19 = 119 \text{ mod } 19 = 5$
6. Bob gửi bản mã  $(11, 5)$

Giải mã:

1. Alice tính  $K = (C_1)^{X_A} \text{ mod } q = 11^5 \text{ mod } 19 = 161051 \text{ mod } 19 = 7$
2. Vậy  $K-1$  trong GF(19) là  $7-1 \text{ mod } 19 = 11$ .
3. Cuối cùng,  $M = (C_2 K-1) \text{ mod } q = 5 \times 11 \text{ mod } 19 = 55 \text{ mod } 19 = 17$ .

Nếu một bản tin phải bị chia thành các khối và gửi như một chuỗi các khối đã được mã hoá, một giá trị độc nhất  $k$  nên được dùng cho mỗi khối. Nếu  $k$  được dùng nhiều cho nhiều hơn một khối, thông tin trong một khối  $M_1$  của bản tin cho phép người dùng tính các khối khác như sau. Cho

$$\begin{aligned} C_{1,1} &= \alpha^k \text{ mod } q; C_{2,1} = KM_1 \text{ mod } q \\ C_{1,2} &= \alpha^k \text{ mod } q; C_{2,2} = KM_2 \text{ mod } q \end{aligned}$$

Thì,

$$\frac{C_{2,1}}{C_{2,2}} = \frac{KM_1 \text{ mod } q}{KM_2 \text{ mod } q} = \frac{M_1 \text{ mod } q}{M_2 \text{ mod } q}$$

Nếu  $M_1$  đã biết thì  $M_2$  có thể dễ dàng tính được là:

$$M_2 = (C_{2,1})^{-1} C_{2,2} M_1 \text{ mod } q$$

Tính bảo mật của hệ thống mật mã Elgamal dựa trên độ khó trong tính toán hàm logarit rời rạc. Để khôi phục mã cá nhân của A, một người theo dõi phải tính  $X_A = \text{dlog}_a, q(Y_A)$ . Một cách khác, để khôi phục mã dùng một lần K, một người theo dõi phải tìm được số bất kì  $k$ , và việc này yêu cầu tính toán hàm logarit rời rạc  $k = \text{dlog}_a, q(C_1)$ . Các phép tính này được coi là không thể thực hiện được nếu  $p$  có ít nhất 300 con số thập phân và  $q - 1$  có ít nhất một nhân tử nguyên tố lớn.

### 3.4 Tạo số giả ngẫu nhiên sử dụng mật mã bất đối xứng

Như chúng ta đã biết là bởi mật mã khối đối xứng tạo ra một đầu ra ngẫu nhiên nó có thể được xem như cơ sở của bộ tạo số giả ngẫu nhiên (PRNG). Tương tự như vậy một thuật toán mã hóa không đối xứng tạo đầu ra ngẫu nhiên cũng có thể sử dụng để xây dựng nên bộ PRNG. Tuy nhiên các thuật toán không đối xứng thường chậm hơn so với thuật toán đối xứng nên nó không được dùng để tạo ra các luồng bit PRNG có kết thúc mở. Hơn nữa tiếp cận không đối xứng là hữu dụng với việc tạo ra hàm giả ngẫu nhiên PRF để tạo ra chuỗi bit giả ngắn. Trong phần này chúng ta xem xét 2 thiết kế PRNG dựa trên các hàm giả ngẫu nhiên.

#### PRNG dựa trên RSA

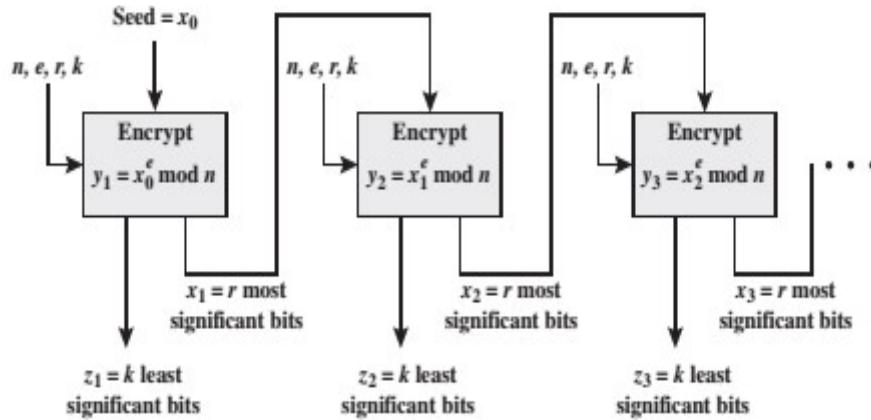
Đối với một độ dài khóa vừa đủ, thuật toán RSA được xem là an toàn và là ứng viên tốt để làm cơ sở bộ PRNG như bộ PRNG được biết đến là PRNG Micali-Schnorr được khuyến nghị trong tiêu chuẩn X9.82 (bộ tạo số ngẫu nhiên) của ANSI và chuẩn 18031 (bộ tạo bit ngẫu nhiên) của ISO.

PRNG có nhiều nét cấu trúc giống với kiểu phản hồi đầu ra OFB được sử dụng như một PRNG. Trong trường hợp này thuật toán mật mã hóa RSA được dùng nhiều hơn là mật mã khối đối xứng. Thêm nữa phần đầu ra được đưa quay trở lại ở vòng lặp tiếp theo của thuật toán mã hóa và phần còn lại của đầu ra được sử dụng là các bit giả ngẫu nhiên. Việc phân chia đầu ra thành 2 phần riêng biệt là để các bit giả ngẫu nhiên của tầng này không phải là đầu vào của tầng kế tiếp. Sự phân chia này có thể đóng góp vào việc chuyển tiếp không thể đoán trước.

Chúng ta có thể định rõ PRNG như sau:

Thiết lập	Lựa chọn số nguyên tố $p, q; n = pq; \phi(n) = (p-1)(q-1)$ . Lựa chọn e sao cho $\gcd(e, \phi(n)) = 1$ . Đây là những lựa chọn thiết lập RSA tiêu chuẩn. Thêm nữa, cho $N = \lceil \log_2 n \rceil + 1$ (độ dài bit của n). Chọn r, k sao cho $r+k=N$ .
Mầm	Lựa chọn một mầm $x_0$ của độ dài bit r

Sự tạo thành	Tạo một chuỗi giả ngẫu nhiên có độ dài kxm sử dụng vòng lặp <b>for</b> $i$ from 1 to $m$ <b>do</b> $y_i = x_{i-1}^e \text{ mod } n$ $x_i = r$ các bit có nghĩa cao nhất của $y_i$ $z_i = k$ các bit có nghĩa thấp nhất của $y_i$
Đầu ra	Chuỗi đầu ra là $z_1 \square z_2 \square \dots \square z_m$



Hình 3. 13: Tạo bit giả ngẫu nhiên Macallli-Schnorr.

Các tham số  $n, r, e$  và  $k$  được lựa chọn để thoả mãn 6 yêu cầu sau.

1.  $N = pq$ ,  $n$  được chọn là tích của 2 số nguyên tố để có độ mã hoá cao phù hợp với yêu cầu của RSA.
2.  $1 < e < \phi(n)$ ;  $\gcd(e, \phi(n)) = 1$  Đảm bảo rằng con đường từ  $s \rightarrow se \text{ mod } n$  là 1 đến 1.
3.  $re \geq 2N$  Đảm bảo rằng hàm mũ yêu cầu một sự giảm module hoàn chỉnh.
4.  $r \geq 2$  mű Bảo vệ khỏi một cuộc tấn công mã hoá.
5.  $k, r$  là cấp số của 8 Để tiện tính toán
6.  $k \geq 8$ ;  $r + k = N$  Tất cả bits được sử dụng

Biến số mũ trong yêu cầu 4 được xác định trong NIST SP 800-90 như sau: Một số liên quan đến lượng công việc (là số lượng các chu trình) cần thực hiện để phá vỡ một thuật toán hoặc hệ thống mã hoá; sức mạnh bảo vệ được cụ thể hoá bằng bits và là một giá trị đặc biệt trong tập (112, 128, 192, 256).

Rõ ràng có một sự trao đổi giữa  $r$  và  $k$ . Vì RSA là tính toán sâu hơn so với mã khối, ta sẽ muốn tạo ra nhiều bit giả ngẫu nhiên nhất trong một vòng lặp, và vì vậy sẽ thích hợp với một giá trị  $k$  lớn. Tuy nhiên, để mã hóa đủ mạnh ta cần  $r$  lớn nhất có thể.

Ví dụ, nếu  $e = 3$  và  $N = 1024$ , thì chúng ta sẽ có bất phương trình  $3^r > 1024$ , cho ta một giá trị  $r$  nhỏ nhất là 683 bit. Nếu  $r$  được đặt thành kích thước đó,  $k = 341$  bit được tạo ra với mỗi hàm mũ (Mỗi mã hoá của RSA). Trong trường hợp này, mỗi hàm mũ cần duy nhất một module bình phương của một số 683 bit và một module tích. Đó là trong trường hợp ta chỉ tính ( $x_1 \times (x_{12} \bmod n)$ ) mod n.

### PRNG dựa trên mã hoá theo đường cong elip

Trong phần này, chúng ta sẽ tóm lược ngắn gọn một kĩ thuật được phát triển bởi Bộ An Ninh Quốc Gia Hoa Kì (NSA) được biết đến là đường cong elip đôi PRNG (DEC PRNG). Kĩ thuật này được đề xuất trong NIST SP 800- 90, chuẩn X9.82 của ANSI và chuẩn ISO 18031. Tóm tắt thuật toán được trình bày như sau: Cho P và Q là 2 điểm đã biết trên đường cong elip. Giá trị khởi tạo của DEC PRNG là một số nguyên bất kì  $s_0 \in \{0, 1, \dots, \#E(GF(p))\}$  mà  $\#E(GF(p))$  biểu diễn số điểm trên đường cong. Cho  $lsbi(s)$  biểu diễn  $i$  là bit nhỏ nhất của số nguyên  $s$ . DEC PRNG biến giá trị khởi tạo thành một chuỗi giả ngẫu nhiên dài  $240^k$ ,  $k > 0$ , như sau:

```

For i = 1 to k do
    Set s1 ← x(Si-1 P)
    Set r1 ← lsb240 ( x(si Q))
    end for
    Return r1, ..., rx
  
```

Về vấn đề an ninh PRNG, lí do duy nhất để nó được sử dụng là nó dùng trong một hệ thống đã cài đặt ECC nhưng không cài đặt bất kì thuật toán bảo mật đối xứng, bất đối xứng, hoặc cứng nào mà có thể sử dụng để dựng nên PRNG.

### Câu hỏi ôn tập chương 3

1. Các đặc trưng cơ bản và ưu nhược điểm của hệ mật khẩu công khai.
2. Khía cạnh bảo mật và nhận thức của hệ thống mã hóa công khai.
3. Mô tả giải thuật và các đặc tính cơ bản của RSA.
4. Khả năng an toàn của RSA.
5. Nguyên lý trao đổi khóa Diffie-Hellman.
6. Khai quát đặc trưng của thông tin mật mã Elgamal

## Chương 4: Các giải thuật toàn vẹn dữ liệu

### 4.1 Hàm băm

#### 4.1.1 Ứng dụng của hàm băm

Hàm băm được xem là thuật toán mã hoá linh hoạt nhất trong các loại thuật toán mã hoá và được sử dụng rộng rãi trong nhiều ứng dụng bảo mật và giao thức Internet. Để hiểu rõ về các yêu cầu và tác động bảo mật của các hàm băm mã hóa, ta sẽ xem xét các miền ứng dụng được triển khai hàm băm.

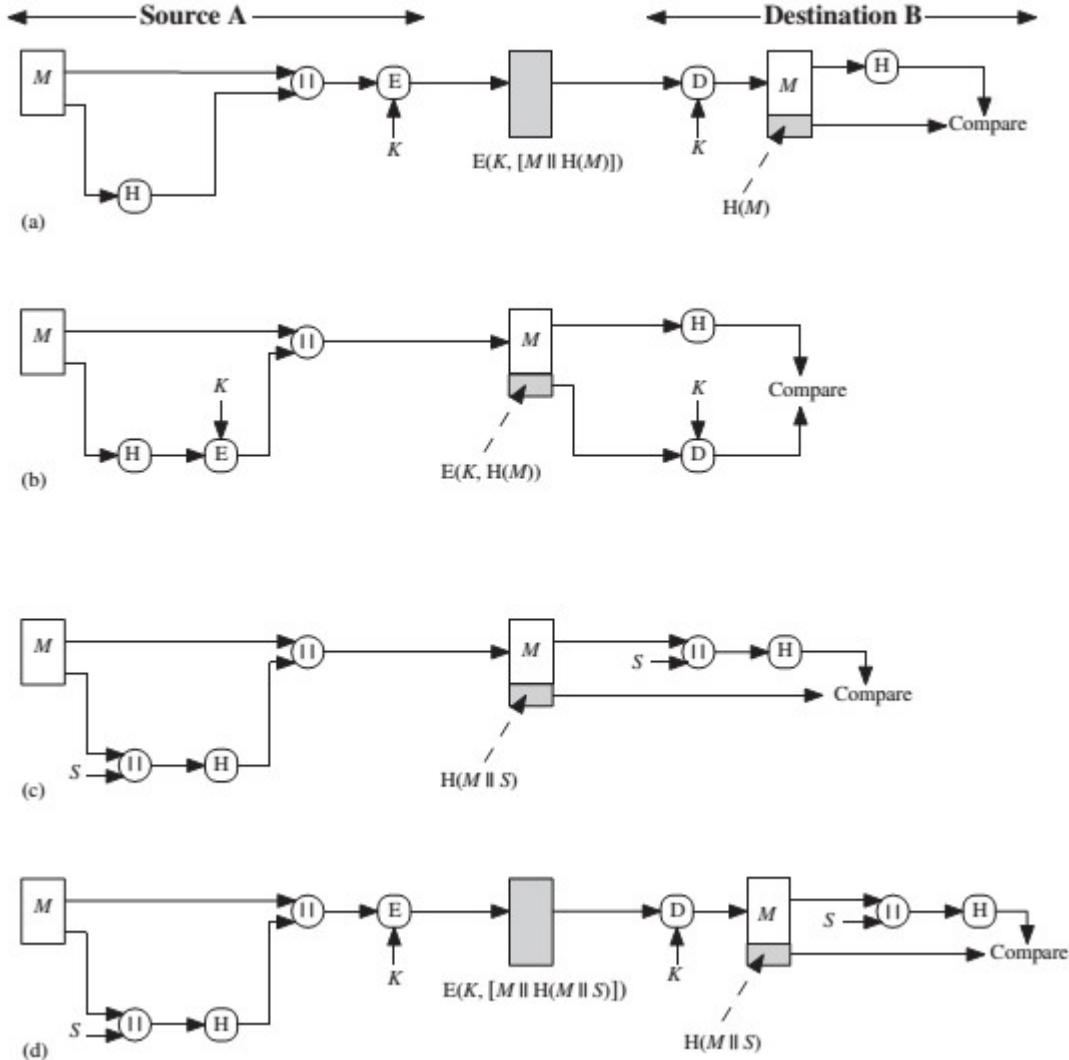
##### Xác thực bản tin

Xác thực bản tin là một cơ chế hay dịch vụ được dùng để xác minh tính toàn vẹn của bản tin; nghĩa là đảm bảo nội dung bản tin nhận được chính xác như bản tin gửi đi (không có bất kỳ sự sửa đổi, chèn, xoá nội dung, lặp lại nào của bản tin). Trong nhiều trường hợp thì cơ chế xác thực đưa ra yêu cầu xác thực đối với người gửi. Hàm băm được sử dụng trong cơ chế xác thực bản tin để cung cấp một giá trị băm, thường được gọi là bản tin rút gọn.

Bản chất của việc áp dụng hàm băm trong cơ chế xác thực như sau: Người gửi tính toán một giá trị băm từ các bít trong bản tin và truyền đi đồng thời bản tin và giá trị băm đó. Người nhận cũng thực hiện việc tính giá trị băm tương tự từ các bít trong bản tin nhận được và so sánh với giá trị băm được gửi kèm. Nếu như các giá trị băm được so sánh không trùng với nhau thì có nghĩa là bản tin nhận được (hoặc các giá trị băm) đã bị thay đổi. Các hàm băm phải được truyền đi một cách an toàn. Điều này nghĩa là ngay cả khi bản tin bị sửa đổi hoặc thay thế thì kẻ xấu không thể thay đổi được giá trị băm để đánh lừa người nhận. Hình 4.1 minh họa các cách áp dụng hàm băm trong việc xác thực bản tin.

- i) Mã băm được nối vào bản tin, sau đó được mã hoá bởi mã hoá đối xứng. Vì chỉ có A và B biết khoá bí mật nên bản tin được đảm bảo truyền từ A và không bị sửa đổi. Do cả mã băm và bản tin đều được mã hoá nên tính bảo mật cũng được cung cấp trong trường hợp này.
- ii) Chỉ có các mã băm được mã hoá bằng mã hoá đối xứng. Điều này giúp giảm gánh nặng xử lý cho các ứng dụng không yêu cầu bảo mật.

- iii) Giả sử bên gửi và nhận chia sẻ một giá trị bí mật  $S$ . Giá trị này được nối vào bản tin  $M$  và được sử dụng để tính toán giá trị băm. Sau đó, giá trị băm này được cộng với bản tin và truyền đi. Tại đầu nhận, B cũng có khả năng tính toán giá trị băm vì nó cũng biết  $S$ . Vì chỉ có A và B biết  $S$ , nên kẻ xâm nhập không thể sửa đổi hoặc làm giả bản tin. Trong phương pháp này, tính bảo mật không được cung cấp.
- iv) Phương pháp này khác phương pháp (c) ở việc tính bảo mật được thêm vào để mã hoá toàn bộ bản tin và mã băm trước khi truyền đi.



Hình 4.1: Ví dụ về sử dụng hàm băm trong nhận thực bản tin

Khi tính bảo mật không cần thiết thì phương pháp (b) ưu việt hơn hai phương pháp (a) và (d) do phải tính toán ít hơn. Thậm chí việc tránh mã hoá như trong phương pháp (c) ngày càng nhận được nhiều sự quan tâm hơn bởi một vài nguyên do sau:

- + Mã hoá bằng phần mềm tương đối chậm. Nên dù lượng dữ liệu cần mã hoá trong mỗi bản tin là nhỏ nhưng mỗi hệ thống đều chứa rất nhiều bản tin cần trao đổi.
- + Chi phí cho việc mã hoá bằng phần cứng là không nhỏ. Mặc dù, những chip giá rẻ có khả năng chạy thuật toán DES nhưng đối với một hệ thống có rất nhiều nút mạng thì chi phí cũng tăng lên đáng kể.
- + Mã hoá bằng phần cứng phù hợp hơn trong việc mã hoá dữ liệu lớn. Đối với các dữ liệu nhỏ thì phần lớn thời gian bị chiếm bởi các thủ tục khởi tạo và gọi hàm.
- + Các thuật toán mã hoá thường bị giới hạn bởi các bảng sáng chế. Do đó, cần thêm một khoản chi phí để nhận được sự cho phép sử dụng các thuật toán đó.

Xác thực bản tin sử dụng mã xác thực bản tin (MAC: Message Authentication Code), hay được biết đến như hàm băm có khoá thường được sử dụng phổ biến. Thông thường, MACs được sử dụng bởi hai bên cùng chia sẻ khoá bí mật để xác định thông tin trao đổi giữa chúng. Hàm MAC sử dụng đầu vào là một khoá bí mật và bản tin để tạo ra một mã băm được gọi là MAC. Giả sử, một kẻ tấn công muốn sửa đổi bản tin thì không thể thay đổi được giá trị MAC nếu không có khoá bí mật. Chú ý, bên nhận biết bên gửi là bởi vì không có ai khác biết được khoá bí mật.

Chú ý rằng, MAC là hàm kết hợp của kết quả băm và mã hoá (hình 4.3b).  $E(K, H(M))$  là hàm của bản tin có độ dài bất kỳ và khoá bí mật  $K$ , nó tạo ra đầu ra có độ dài cố định. Trong thực tế, các thuật toán MAC thường hiệu quả hơn các thuật toán mã hoá.

#### 4.1.2 Các yêu cầu và độ an toàn hàm băm

Để hiểu rõ yêu cầu và an toàn của hàm băm, ta cần định nghĩa hai điều kiện. Đối với giá trị băm  $h = H(x)$ ,  $x$  được gọi là nghịch ảnh của  $h$ . Nghĩa là  $x$  là một khối dữ liệu có hàm băm là  $h$  sử dụng hàm  $H$ . Vì hàm  $H$  là phép ánh xạ từ nhiều phần tử thành một, nên với một giá trị  $h$  cho trước, có thể tạo ra nhiều nghịch ảnh. Một xung đột xảy ra nếu  $x \neq y$  và  $H(x) = H(y)$ . Do chúng ta sử dụng các hàm băm để đảm bảo tính toàn vẹn của dữ liệu nên các xung đột là điều không mong muốn.

Chúng ta sẽ phân tích xem có bao nhiêu nghịch ảnh có thể được tạo ra với một giá trị băm cho trước như là một phép đo số xung đột có thể xảy ra với một giá trị băm cho trước. Giả sử rằng, độ dài của giá trị băm là  $n$  bit và hàm  $H$  có các bản tin đầu vào hay các khối dữ liệu có độ dài  $b$  bit ( $b > n$ ). Khi đó, tổng số bản tin có thể là  $2^b$  và tổng số các giá trị băm là  $2^n$ . Trung bình, mỗi giá trị băm tương ứng với  $(2b - n)$  nghịch ảnh. Trên thực tế, nếu  $H$  phân bố đều các giá trị băm thì mỗi giá trị băm sẽ có xấp xỉ  $(2b - n)$  nghịch ảnh. Nếu bây giờ, chúng ta xét đầu vào có độ dài bất kỳ, không phải chỉ là một độ dài cố định như trước thì số nghịch ảnh tạo ra từ mỗi giá trị băm là một giá trị lớn bất kỳ. Tuy nhiên, rủi ro an ninh trong việc sử dụng hàm băm không nghiêm trọng như theo phân tích này. Để hiểu hơn về vấn đề bảo mật của hàm băm mã hóa, chúng ta cần định nghĩa chính xác các yêu cầu về bảo mật của chúng.

### Các yêu cầu bảo mật cho các hàm băm bảo mật

Bảng 4.1 liệt kê các yêu cầu được chấp nhận chung cho một hàm băm mã hóa. Ba yêu cầu đầu tiên được áp dụng cho các ứng dụng thực tế của hàm băm. Yêu cầu thứ tư, chống nghịch ảnh, là một đặc tính một chiều, nghĩa là: một mã băm có thể dễ dàng tạo ra bởi một bản tin cho trước nhưng hầu như không có khả năng tái tạo lại bản tin thông qua một mã băm cho trước. Đặc tính này vô cùng quan trọng nếu các kỹ thuật xác thực sử dụng các giá trị bí mật. Giá trị bí mật không được gửi đi nhưng nếu hàm băm không phải là hàm một chiều thì kẻ tấn công có thể dễ dàng phát hiện ra được giá trị bí mật. Nếu kẻ tấn công có thể quan sát hoặc chặn một đường truyền, kẻ tấn công có thể thu được bản tin  $M$  và mã băm  $h = H(S \parallel M)$ . Sau đó, kẻ tấn công có thể triển khai ngược hàm băm để lấy được  $S \parallel M = H^{-1}(MDM)$ . Do bây giờ, kẻ tấn công có cả  $M$  và  $SAB \parallel M$ , nên có thể dễ dàng tìm ra được  $SAB$ .

Bảng 4.1: Các yêu cầu hàm băm bảo mật

Yêu cầu	Mô tả
Kích thước biến đầu vào	$H$ có thể ứng dụng cho một khối dữ liệu có kích thước.
Kích thước đầu ra cố định	$H$ tạo ra đầu ra có độ dài cố định.
Hiệu quả	$H(x)$ dễ dàng tính toán cho một $x$ bất kỳ cho trước, có thể triển khai trên cả phần cứng và phần mềm.
Tính chất một chiều	Với bất kỳ giá trị băm $h$ , tính toán $y$ để $H(y) = h$ là bất khả thi.

Kháng xung đột yếu	Với bất kỳ một khối $x$ , tính toán để tìm $y \neq x$ với $H(y)=H(x)$ là bất khả thi.
Kháng xung đột mạnh	Bất khả thi trong tính toán tìm kiếm một cặp bất kỳ $(x,y)$ để $H(y)=H(x)$ .
Giả ngẫu nhiên	Đầu ra của $H$ đảm bảo tính giả ngẫu nhiên

Đặc tính thứ năm, chống nghịch ảnh bậc 2, đảm bảo việc không thể tìm được một bản tin thay thế với giá trị băm của bản tin cho trước. Điều này ngăn chặn sự giả mạo khi mã băm mã hóa được sử dụng. Nếu đặc tính này không thỏa mãn, kẻ tấn công có khả năng thực hiện các hành động sau đây: Đầu tiên, quan sát hoặc chặn một bản tin cùng với mã băm mã hóa của nó; tiếp theo, tạo một mã băm không mã hóa từ bản tin; cuối cùng, tạo ra bản tin thay thế với mã băm tương tự.

Hàm băm thỏa mãn năm đặc tính đầu tiên được xem như hàm băm yếu. Nếu đặc tính thứ sáu, chống xung đột, thỏa mãn thì hàm băm đó được gọi là hàm băm mạnh. Một hàm băm mạnh có khả năng chống lại tấn công trong đó một bên tạo ra bản tin cho một bên khác kỵ.

### Các tấn công đoán thử đúng sai

Đối với các thuật toán mã hóa thường có hai loại tấn công nhắm vào hàm băm là: Tấn công đoán thử đúng sai và giải mã. Tấn công đoán thử đúng sai không phụ thuộc vào một thuật toán cụ thể mà chỉ phụ thuộc vào độ dài bit. Trong trường hợp của hàm băm, tấn công đoán thử đúng sai chỉ phụ thuộc vào độ dài của giá trị băm. Ngược lại, giải mã là tấn công dựa trên các điểm yếu trong một thuật toán mã hóa cụ thể nào đó.

### Tấn công vào nghịch ảnh và nghịch ảnh bậc hai

Đối với tấn công vào nghịch ảnh hoặc nghịch ảnh bậc hai, kẻ tấn công muốn tìm một giá trị  $y$  theo hàm  $H(y)$  có giá trị bằng giá trị băm  $h$  cho trước. Phương pháp tấn công đoán thử đúng sai sẽ chọn giá trị  $y$  ngẫu nhiên và thử cho đến khi xung đột xảy ra. Đối với một giá trị băm  $m$ -bit, số phép thử tỉ lệ với  $2^m$ . Nghĩa là, kẻ tấn công cần phải thử  $2^{m-1}$  giá trị  $y$  để tìm ra một giá trị chính xác.

### Tấn công vào chống xung đột

Đối với tấn công vào đặc tính chống xung đột, kẻ xấu muốn tìm hai bản tin hoặc các khối dữ liệu  $x$  và  $y$ , có cùng một hàm băm  $H(x) = H(y)$ . Tấn công này cần ít phép đoán thử hơn so với tấn công vào nghịch ảnh và nghịch ảnh bậc hai. Về cơ bản, nếu các biến

ngẫu nhiên được phân bố đều trong dải từ 0 đến  $N-1$  thì xác suất để một phần tử lặp lại vượt quá 0.5 sau  $\sqrt{N}$  lựa chọn được thực hiện. Do đó, đối với giá trị băm  $m$ -bit, nếu các khối dữ liệu được chọn một cách ngẫu nhiên thì có thể tìm thấy hai khối dữ liệu có cùng giá trị băm sau  $\sqrt{2^m} = 2^{m/2}$  phép thử.

## 4.2 Mã xác thực bản tin MAC

### 4.2.1 Các yêu cầu xác thực bản tin

Trong ngữ cảnh bản tin chuyển xuyên suốt qua mạng truyền thông, có thể tồn tại các loại tấn công sau đây:

1. Tấn công tiết lộ (disclosure): Phân phối nội dung bản tin tới bất kỳ cá nhân hoặc đối tượng không sở hữu khóa mật mã phù hợp.
2. Phân tích lưu lượng: Tìm hiểu mẫu lưu lượng giữa hai bên truyền thông. Trong một ứng dụng hướng kết nối, tần suất và khoảng thời gian kết nối có thể được xác định. Trong một môi trường hướng kết nối hoặc phi kết nối, số lượng và chiều dài bản tin giữa các bên truyền thông có thể được xác định.
3. Ngụy trang: Đưa thông tin vào các bản tin trong mạng từ nguồn giả mạo. Kiểu tấn công này bao gồm việc tạo các bản tin bởi kẻ tấn công chủ ý đến từ một thực thể được ủy quyền. Nó cũng bao gồm các bản tin báo nhận giả mạo của các bản tin được nhận hoặc không được nhận bởi một người nào đó khác với người nhận đích thực.
4. Tấn công sửa đổi nội dung: Các thay đổi đối với nội dung của bản tin, bao gồm sự thêm vào, xóa bớt, đổi chỗ và sửa đổi.
5. Sửa đổi chuỗi: Bất kỳ sự sửa đổi nào đối với chuỗi các bản tin giữa các bên truyền thông, bao gồm việc thêm, bớt và sắp xếp lại thứ tự.
6. Sửa đổi định thời: Làm trễ hoặc phát lại các bản tin. Trong một ứng dụng hướng kết nối, toàn bộ một phiên hoặc chuỗi các bản tin có thể là một bản phát lại của một vài phiên hợp lệ trước đó hoặc các bản tin riêng biệt trong chuỗi các bản tin có thể bị trễ hoặc bị phát lại. Trong một ứng dụng phi kết nối, một bản tin riêng biệt (ví dụ như một datagram) có thể bị trễ hoặc phát lại.
7. Chối bỏ nguồn: Chối bỏ sự truyền dẫn của bản tin do nhầm lẫn nguồn.
8. Chối bỏ đích: Chối bỏ sự truyền dẫn của bản tin do sai đích.

Các phương pháp đối phó với hai loại tấn công đầu nằm trong khuôn khổ của bảo mật bản tin. Các giải pháp cho các loại tấn công từ số (3) đến (6) liên quan tới nhận thực bản tin. Các cơ chế đối phó với tấn công số (7) đi theo hướng chữ ký số. Nói chung, một kỹ thuật chữ ký số cũng sẽ giúp đối phó với một vài hoặc tất cả các loại tấn công từ (3) đến (6). Để giải quyết vấn đề số (8) có thể cần tới một sự kết hợp giữa chữ ký số và một giao thức bảo mật. Như vậy, nhận thực bản tin là một thủ tục để xác thực rằng các bản tin được nhận đến từ đúng nguồn và không bị sửa đổi hoặc thay thế. Chữ ký số là một kỹ thuật nhận thực bao gồm cả các giải pháp đối phó với sự chối bỏ nguồn.

#### 4.2.2 Chức năng xác thực bản tin

Bất kỳ một cơ chế nhận thực bản tin hoặc chữ ký số nào đều có hai lớp chức năng. Tại lớp chức năng thấp, cần phải có một vài loại chức năng cung cấp một ký hiệu nhận: một giá trị được sử dụng để nhận thực. Chức năng lớp thấp này sau đó được sử dụng như là một yếu tố cơ bản cho một giao thức nhận thức lớp cao để cho phép bên nhận xác minh tính xác thực của một bản tin.

Mục này đưa các loại hàm có thể được sử dụng để đưa ra một ký hiệu nhận thực. Các hàm này có thể được phân làm ba loại:

- + **Hàm băm:** Một hàm ánh xạ một bản tin với độ dài bất kỳ vào một giá trị băm có chiều dài cố định, giá trị này là ký hiệu nhận thực.
- + **Mã hóa bản tin:** Từ mã của toàn bộ bản tin là ký hiệu nhận thực của bản tin đó.
- + **Mã xác thực bản tin (MAC):** Một hàm của bản tin và khóa bí mật tạo ra giá trị có chiều dài cố định có chức năng như một ký hiệu xác thực.

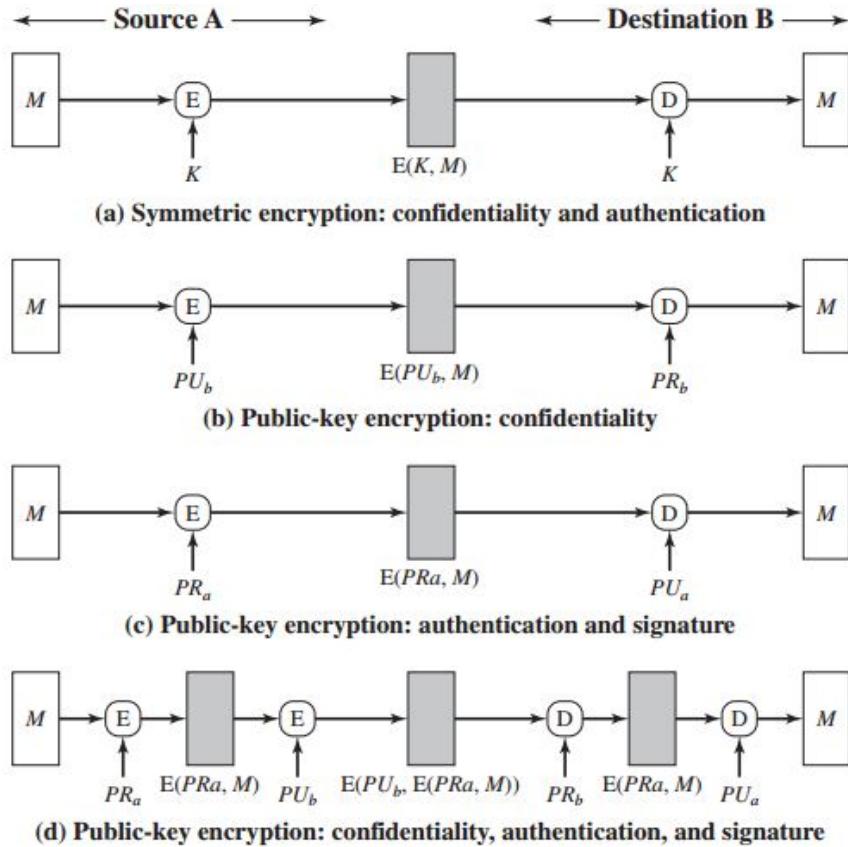
##### *Mã hóa bản tin*

Bằng chính mã hóa của bản tin có thể cung cấp một giải pháp cho vấn đề nhận thực và khác nhau đối với các cơ chế mã hóa khóa đối xứng và khóa công khai.

**Mã hóa khóa đối xứng:** Xét một trường hợp sử dụng khóa đối xứng đơn giản hình 4.2a. Một bản tin  $M$  được truyền từ nguồn A tới đích B được mã hóa sử dụng khóa bí mật  $K$  được chia sẻ giữa A và B. Nếu không có bên nào khác biết khóa này, tính bảo mật được đảm bảo: Không có một bên nào khác có thể thấy được nội dung bản tin.

Thêm vào đó, B được đảm bảo rằng bản tin được tạo ra bởi A. Tại sao? Bản tin phải được gửi từ A, bởi vì A là bên duy nhất sở hữu khóa  $K$ , và do đó là bên duy nhất có thông

tin cần thiết để tạo ra bản mã mà có thể được giải mã bằng  $K$ . Hơn nữa, nếu  $M$  được khôi phục lại, B biết rằng không có bit nào của  $M$  bị thay thế, do một kẻ tấn công không biết khóa  $K$  có thể biết được làm cách nào để thay thế các bit trong bản mã để tạo ra sự thay đổi như mong muốn trong bản rõ.



Hình 4.2: Các cách sử dụng cơ bản của mã hóa bản tin

Do vậy chúng ta nói rằng mã hóa đối xứng cung cấp sự nhận thực cũng như là tính bảo mật. Tuy nhiên, tuyên bố này cần có đủ điều kiện. Xem xét những gì xảy ra tại B. Cho trước một hàm giải mã  $D$  và khóa bí mật  $K$ , bên nhận sẽ chấp nhận bất kỳ đầu vào  $X$  nào và đưa ra đầu ra  $Y = D(K, X)$ . Nếu  $X$  là bản mã của một bản tin hợp pháp  $M$  được tạo ra bởi hàm mã hóa tương ứng, khi đó  $Y$  là một bản tin mã hóa nào đó. Ngược lại,  $Y$  có thể là một chuỗi bit vô nghĩa. Có thể cần một vài cách kiểm tra tự động để xác định xem  $Y$  có phải là bản rõ hợp pháp không và do đó có đến từ A hay không. Ngụ ý sâu xa của đoạn trên là từ gốc nhìn của nhận thực. Trong trường hợp đó, không có cách nào để xác định một cách tự động, tại đích, để đánh giá xem một bản tin đến là bản mã của một bản tin hợp pháp hay không. Điều kết luận sau đây là không thể chối cãi: Nếu  $M$  có thể là bất

kỳ mẫu bit nào, khi đó bất kỳ giá trị nào của  $X$ , giá trị  $Y = D(K, X)$  là một mẫu bit nào đó và do đó phải được chấp nhận là một bản rõ xác thực.

Do vậy, một cách tổng quát, chúng ta yêu cầu chỉ có một tập con của tất cả các trường hợp của mẫu bit được xem là bản rõ hợp pháp. Khi đó, bất kỳ bản mã giả mạo đều khó có thể tạo ra bản rõ hợp pháp. Ví dụ, giả sử chỉ có một trong 106 mẫu bit là bản rõ hợp pháp. Khi đó xác suất để một mẫu bit ngẫu nhiên bất kỳ mà ta xét là bản mã, có thể tạo ra một bản rõ hợp lệ là  $10^{-6}$ .

Với một số các ứng dụng mã hóa, các trường hợp mong muốn chiếm đa số là điều đương nhiên. Ví dụ, giả sử ta truyền một bản tin tiếng Anh sử dụng mã hóa Caesar với độ dịch bằng 1 ( $K=1$ ). A gửi bản mã hợp lệ sau:

```
Nbsftfbupbutboeepftfbupbutboemjuumfmbnctfbujwz
```

B giải mã để tạo bản rõ sau:

```
Mareseatoatsanddoeseatoatsandlittlelambseativy
```

Một phân tích tần suất đơn giản xác nhận bản tin này có dạng của ngôn ngữ tiếng Anh thông thường. Mặt khác, nếu một kẻ tấn công tạo chuỗi ký tự ngẫu nhiên sau:

```
Zuvrsoevgqqlzwigamdvnmhpccxiuireosfbcebtqxsxq
```

nó được giải mã thành:

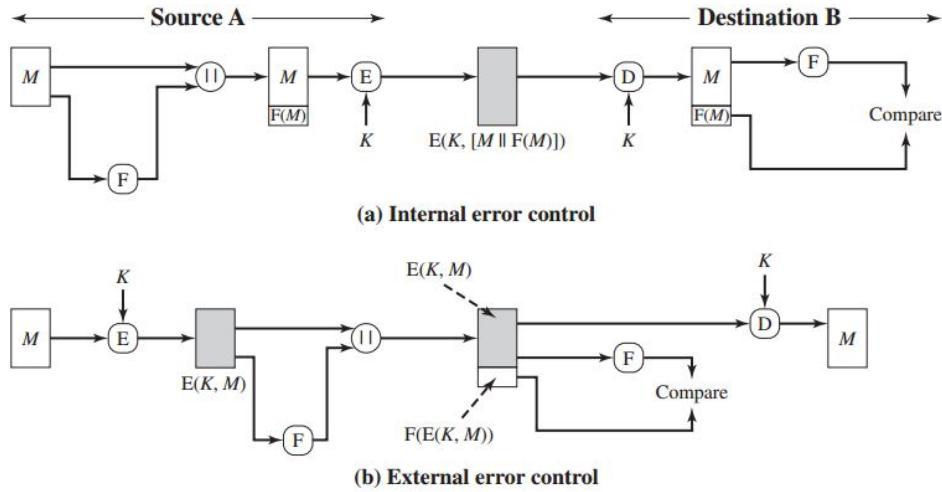
```
ytuqrndufpwkyvhfzlcumlgolbbwhqqdnreabdaspwrrwp
```

là một chuỗi không có dạng ngôn ngữ tiếng Anh.

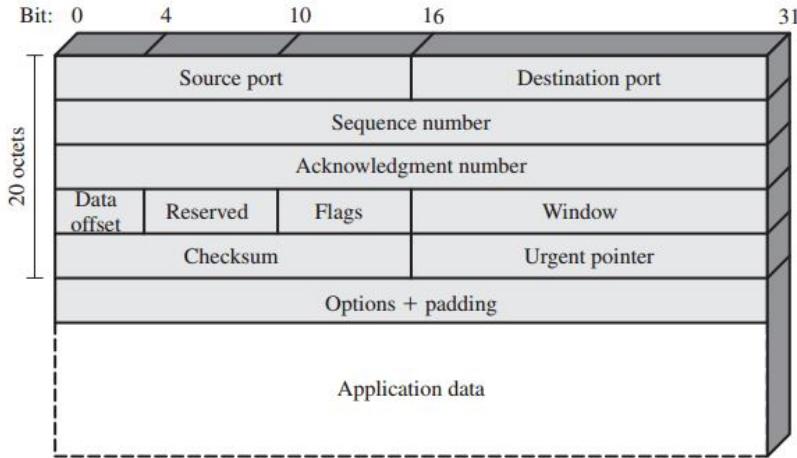
Tuy nhiên, giả sử bản rõ là một file nhị phân hoặc bản chụp X quang được số hóa, sự quyết định đúng đắn của bản rõ sẽ rất khó khăn. Do vậy, một kẻ tấn công có thể gây tổn hại ở một mức độ nào đó bằng cách đưa vào các bản tin có nội dung ngẫu nhiên với mục đích mạo danh người dùng hợp pháp.

Một giải pháp cho vấn đề này là buộc bản rõ có một cấu trúc nhất định nào đó mà có thể dễ dàng ghi nhận nhưng không thể sao chép mà không dựa vào hàm mã hóa. Ví dụ, chúng ta có thể thêm vào một mã phát hiện lỗi, còn được gọi là một chuỗi kiểm tra khung (FCS – Frame Check Sequence) hoặc tổng kiểm tra, đối với mỗi bản tin  $M$  trước khi mã hóa, như minh họa trong hình 4.2a. A chuẩn bị một bản rõ  $M$  sau đó sử dụng nó làm đầu vào của một hàm  $F$  có để tạo đầu ra là FCS. FCS được thêm vào  $M$  thành một khối và toàn bộ khối này được mã hóa. Tại phía nhận, B giải mã khối tin nhận được và xử lý đầu ra như là một bản tin với phần FCS được thêm vào. B sử dụng cùng một hàm  $F$

để tạo lại FCS của bản tin sau giải mã. Nếu giá trị FCS này giống với giá trị FCS được gửi đến thì bản tin được xem là xác thực. Một chuỗi bất kỳ khó có thể có mối quan hệ này.



Hình 4.3: Điều khiển lỗi trong và ngoài



Hình 4.4: Phân đoạn TCP

Cần lưu ý rằng thứ tự thực hiện của FCS và các hàm mã hóa là rất quan trọng. Chuỗi được minh họa trong hình 4.3a liên quan tới điều khiển lỗi bên trong (Hình 4.3b). Với điều khiển lỗi bên trong, sự nhận thực được cung cấp do một kẻ tấn công có thể (rất hiếm khi) tạo ra được bản mã mà khi giải mã, có các bit điều khiển lỗi hợp lệ. Thay vì FCS là mã ngoài (outer code), một kẻ địch có thể cấu trúc nên các bản tin với các mã lỗi hợp lệ. Mặc dù kẻ địch không thể biết được bản rõ là gì, anh ta/cô ta có thể vẫn hy vọng tạo ra sự gián đoạn và xáo trộn trong hệ thống.

Một mã điều khiển lỗi chỉ là một ví dụ, sự thật là bất kỳ loại cấu trúc nào được thêm vào bản tin truyền đều có chức năng tăng cường khả năng nhận thực. Những cấu trúc này được cung cấp bằng việc sử dụng kiến trúc truyền thông bao gồm các giao thức được xếp theo lớp. Ví dụ, xét cấu trúc của bản tin được truyền đi sử dụng kiến trúc giao thức TCP/IP. Hình 4.4 cho thấy định dạng của một phân đoạn TCP với minh họa tiêu đề TCP. Giả sử mỗi cặp máy chủ sử dụng cùng một khóa và không phụ thuộc vào ứng dụng thì ta có thể chỉ mã hóa toàn bộ gói tin trừ phần mào đầu IP. Nếu một kẻ tấn công thay thế mẫu bit ngẫu nhiên nào đó đối với phân đoạn TCP được mã hóa, kết quả là bản rõ có thể không chứa tiêu đề có ý nghĩa. Trong trường hợp này, tiêu đề không chỉ gồm một tổng kiểm tra (được tính trên toàn bộ phần mào đầu) mà còn cả những thông tin hữu ích khác. Các phân đoạn TCP của một kết nối được đánh số tuần tự, sự mã hóa đảm bảo kẻ tấn công không làm trễ, làm thay đổi thứ tự hóa xóa các bất kỳ phân đoạn nào.

### *Mã hóa khóa công khai*

Một trường hợp đơn giản sử dụng mã hóa khóa công khai (hình 4.1b) **cung cấp tính bảo mật nhưng không cung cấp nhận thực**. Nguồn (A) sử dụng khóa công khai PUb của phía nhận (B) để mã hóa bản tin  $M$ . Do chỉ có B sở hữu khóa bí mật PRb tương ứng, chỉ có B có thể giải mã bản tin. Cơ chế này không cung cấp nhận thực, vì bất kỳ kẻ tấn công nào cũng có thể sử dụng khóa công khai của B để mã hóa một bản tin và mạo danh A.

Để cung cấp sự nhận thực, A sử dụng khóa bí mật của nó để mã hóa bản tin và B sử dụng khóa công khai của A để giải mã (Hình 4.1c). Điều này cung cấp tính nhận thực sử dụng cùng một cách suy luận như trong trường hợp mã hóa khóa đối xứng: Bản tin phải đến từ A do chỉ có A sở hữu PRa và do đó là bên duy nhất có thông tin cần thiết để tạo nên bản mã mà có thể được giải mã bằng PUa.Thêm nữa, với cùng cách suy luận: Phải có một cấu trúc bên trong nào đó đối với bản rõ nhờ đó bên nhận có thể phân biệt giữa bản rõ và các bit ngẫu nhiên.

Giả sử rằng tồn tại một cấu trúc như vậy, khi đó cơ chế trong hình 4.1c không cung cấp nhận thực cũng như chức năng tương tự như chữ ký số. Chỉ A có thể cấu trúc nén bản mã bởi vì chỉ có A sở hữu PRa. Thậm chí B, phía nhận, cũng không thể cấu trúc nén bản mã. Do đó, nếu B sở hữu bản mã, B có các phương tiện để chứng minh bản tin phải đến từ A. Thực vậy, A đã “ký” vào bản tin bằng cách sử dụng khóa bí mật của nó để mã hóa. Lưu ý rằng cơ chế này không cung cấp tính bảo mật. Bất kỳ ai sở hữu khóa công khai của A đều có thể giải mã bản mã.

Để cung cấp cả tính bảo mật và nhận thực, đầu tiên A mã hóa  $M$  sử dụng khóa bí mật của nó để tạo chữ ký số, sau đó sử dụng khóa công khai của B để cung cấp tính bảo mật (hình 4.1d). Nhược điểm của tiếp cận này là thuật toán khóa công khai là thuật toán phức tạp cần được thực hiện bốn lần chứ không phải hai lần trong mỗi (phiên) truyền thông.

#### *Mã xác thực bản tin*

Một kỹ thuật nhận thực khác liên quan đến việc sử dụng khóa bí mật để tạo khối dữ liệu nhỏ có kích thước cố định, được biết đến như là tổng kiểm tra mã hóa hoặc MAC, được thêm vào bản tin. Kỹ thuật này giả thiết hai bên tham gia truyền thông, giả sử là A và B, chia sẻ một khóa bí mật chung là  $K$ . Khi A có bản tin cần gửi tới B, nó tính MAC là hàm của bản tin và khóa:

$$\text{MAC} = \text{C}(K, M)$$

Trong đó:

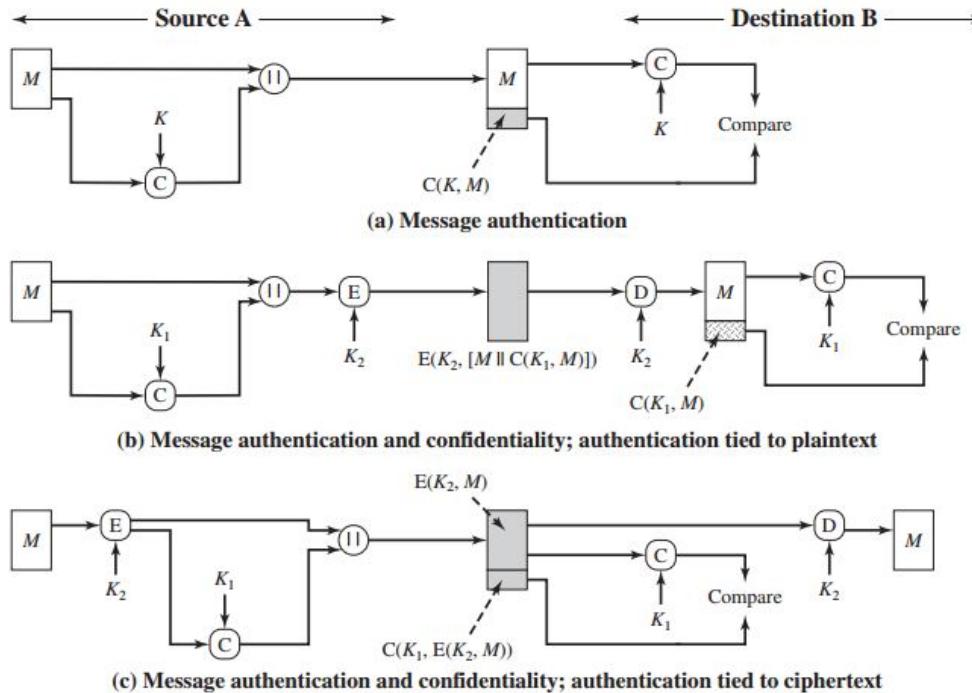
$M$  là bản tin đầu vào

$C$  là hàm MAC

$K$  là khóa bí mật chia sẻ

MAC là mã xác thực bản tin

Bản tin cùng với MAC được truyền tới người nhận mong muốn. Người nhận thực hiện các thao tác tương tự đối với bản tin đến, sử dụng cùng một khóa bí mật, để tạo một giá trị MAC mới. Giá trị MAC nhận được được so sánh với MAC được tính lại (hình 4.4a).



Hình 4.5: Các cách dùng cơ bản của mã xác thực bản tin MAC

Nếu ta giả sử chỉ có duy nhất bên nhận và bên gửi biết khóa chia sẻ, và MAC nhận được khớp với MAC được tính, khi đó:

1. Bên nhận được đảm bảo rằng bản tin đã không bị thay đổi. Nếu một kẻ tấn công thay đổi bản tin nhưng không thay đổi MAC, khi đó giá trị MAC được bên nhận tính lại sẽ không khớp với MAC nhận được. Bởi vì kẻ tấn công được giả thiết rằng không biết khóa bí mật, kẻ tấn công không thể thay đổi giá trị MAC tương ứng với sự thay đổi trong bản tin.
2. Bên nhận được đảm bảo rằng bản tin đến từ đúng người gửi. Bởi vì không ai khác biết được khóa chia sẻ, không ai khác có thể tạo một bản tin với MAC đúng.
3. Nếu bản tin bao gồm một số thứ tự (như được sử dụng trong HDLC, X.25, và TCP), khi đó người nhận có thể được đảm bảo số thứ tự đó là đúng bởi vì một kẻ tấn công không thể thay đổi được số thứ tự.

Một hàm MAC tương tự với mã hóa. Một điểm khác biệt là thuật toán MAC cần phải không thể nghịch đảo, vì nó phải dùng cho giải mã. Tổng quát, hàm MAC là hàm nhiều-tới-một. Miền của hàm bao gồm các bản tin có độ dài tùy ý, trong khi khoảng giá trị bao gồm tất cả các MAC có thể và tất cả các khóa có thể. Nếu một giá trị MAC  $n$ -bit

được sử dụng, khi đó có  $2^n$  giá trị MAC có thể xảy ra, trong khi có  $N$  bản tin có thể có với  $N >> 2n$ . Hơn nữa, với một khóa  $k$  bit có thể có  $2^k$  khóa khác nhau. .

Ví dụ, giả sử chúng ta sử dụng các bản tin 100 bit và khối MAC 10 bit. Khi đó, có tổng cộng 2100 bản tin khác nhau nhưng chỉ có 210 giá trị MAC khác nhau. Như vậy, tính trung bình, mỗi giá trị MAC được tạo bởi tổng cộng  $2100/210 = 290$  bản tin khác nhau. Nếu khóa 5 bit được sử dụng, khi đó có  $25 = 32$  phép ánh xạ khác nhau từ tập các bản tin tới tập các giá trị MAC. Điều này chỉ ra rằng, do các đặc tính toán học của hàm nhận thực, nó ít có khả năng bị phá vỡ hơn mã hóa.

Quá trình được chỉ ra trong hình 4.4a cung cấp tính nhận thực mà không có tính bảo mật, do toàn bộ bản tin được truyền một cách rõ ràng. Tính bảo mật có thể được cung cấp bằng cách sử dụng mã hóa bản tin hoặc sau (hình 4.4b) hoặc trước (hình 4.4c) thuật toán MAC. Trong cả hai trường hợp đó, cần hai khóa riêng biệt, mỗi một khóa được chia sẻ bởi bên gửi và bên nhận. Trong trường hợp thứ nhất, MAC được tính với đầu vào là bản tin rồi sau đó được gắn vào bản tin thành một khối. Toàn bộ khối này sau đó được mã hóa. Trong trường hợp thứ hai, bản tin được mã hóa trước. Sau đó MAC được tính với đầu vào là bản tin đã mã hóa và được gắn vào thành một khối để truyền đi. Thông thường, cách đầu tiên được sử dụng nhiều hơn.

Do mã hóa đối xứng sẽ cung cấp tính nhận thực và bởi vì nó được sử dụng rộng rãi cùng với các sản phẩm khả dụng có sẵn, tại sao không chỉ đơn sử dụng nó thay vì một mã xác thực bản tin riêng biệt?. Có ba tình huống trong đó mã xác thực bản tin được sử dụng:

1. Có một số các ứng dụng trong đó cùng một bản tin được phát quảng bá tới một số lượng các điểm đích, ví dụ như thông báo tới các người dùng rằng mạng hiện tại không khả dụng hoặc tín hiệu cảnh báo từ một trung tâm điều khiển. Nó rẻ hơn và đáng tin cậy hơn so với việc chỉ có một đích đến có trách nhiệm giám sát tính xác thực. Do vậy, bản tin phải được quảng bá dưới dạng bản rõ với một mã xác thực bản tin đi kèm. Hệ thống chịu trách nhiệm sở hữu khóa bí mật và thực hiện nhận thực. Nếu xung đột xảy ra, các hệ thống đích khác được cảnh báo bằng một thông báo chung.
2. Kịch bản khác cần đến nhận thực là khi có một trao đổi trong đó một bên có chịu tải nặng và không đủ thời gian để giải mã tất cả các bản tin đến. Việc chứng thực được thực hiện dựa trên một số cơ sở được chọn lọc, các bản tin được chọn ngẫu nhiên để kiểm tra.

3. Nhận thực một chương trình máy tính dưới dạng bản rõ là một dịch vụ hấp dẫn. Chương trình máy tính có thể được thực hiện mà không cần phải giải mã liên tục – gây lãng phí tài nguyên bộ xử lý. Tuy nhiên, nếu một mã xác thực bản tin được đính kèm vào chương trình, nó có thể được kiểm tra khi có yêu cầu về tính toàn vẹn của chương trình.
4. Đối với một vài ứng dụng, có thể không quan tâm tới việc giữ bí mật cho các bản tin, nhưng nhận thực bản tin là quan trọng. Ví dụ là giao thực quản lý mạng đơn giản phiên bản 3 (Simple Network Management Protocol Version 3 - SNMPv3), được phân tách thành các chức năng bảo mật và nhận thực. Với ứng dụng này, đối với hệ thống được quản lý, nhận thực các bản tin SNMP đến là vấn đề quan trọng, đặc biệt nếu bản tin chứa một lệnh thay đổi các tham số ở hệ thống được quản lý. Mặt khác, có thể không cần thiết phải che giấu các lưu lượng SNMP.
5. Sự phân chia các chức năng nhận thực và bảo mật giúp tạo tính linh động trong kiến trúc. Ví dụ, có thể ta muốn thực hiện nhận thực ở lớp ứng dụng nhưng cung cấp tính bảo mật ở lớp thấp hơn, như ở lớp truyền tải chẳng hạn.
6. Một người dùng muốn kéo dài khoảng thời gian bảo vệ vượt ra ngoài thời gian nhận và chưa cho phép xử lý nội dung bản tin. Với mã hóa bản tin, sự bảo vệ bị mất khi bản tin được giải mã, do vậy bản tin chỉ được bảo vệ chống lại các sửa đổi trái phép trong khi truyền mà không được bảo vệ trong hệ thống đích.

Cuối cùng, lưu ý MAC không cung cấp chữ ký số, do cả bên gửi và bên nhận đều chia sẻ cùng một khóa.

#### 4.2.3 Các yêu cầu cho mã xác thực bản tin

Một mã xác thực bản tin (MAC) được biết đến như là một tổng kiểm tra mã hóa, được tạo bởi hàm C theo dạng:

$$T = \text{MAC}(K, M)$$

Trong đó  $M$  là bản tin có độ dài thay đổi,  $K$  là khóa bí mật được chia sẻ chỉ bởi người gửi và người nhận, và  $\text{MAC}(K, M)$  là ký hiệu nhận thực có chiều dài cố định, đôi khi được gọi là nhãn. Nhãn được thêm vào bản tin ở nguồn tại thời điểm mà bản tin được đảm bảo hoặc được xem là đúng. Bên nhận nhận thực bản tin này bằng cách tính lại nhãn.

Khi toàn bộ bản tin được mã hóa để đảm bảo tính bảo mật, sử dụng mã hóa đối xứng hoặc bất đối xứng, tính bảo mật của cơ chế nói chung phụ thuộc vào chiều dài chuỗi bit khóa. Dựa vào một vài điểm yếu trong thuật toán, kẻ địch phải phải dùng đến tấn công vét cạn để tìm tất cả các khóa có thể có. Tính trung bình, một tấn công như vậy sẽ cần đến  $2^{k-1}$  truy nhập đối với khóa  $k$ -bit. Đặc biệt, đối với tấn công chỉ với bản mã, kẻ tấn công, cho trước bản mã  $C$ , thực hiện  $P_i = D(K_i, C)$  đối với tất cả các giá trị khóa có thể  $K_i$  cho tới khi một  $P_i$  được tạo ra khớp với dạng có thể chấp nhận được của bản rõ.

Trong trường hợp của MAC, vấn đề quan tâm là hoàn toàn khác khi hàm MAC là hàm nhiều-tới-một. Sử dụng các phương pháp vét cạn, làm sao để một kẻ địch có thể khám phá ra một khóa? Nếu tính bảo mật không được sử dụng, kẻ tấn công có được bản rõ và giá trị MAC tương ứng của nó. Giả sử  $k > n$ ; nghĩa là, giả sử kích thước khóa lớn hơn kích thước MAC. Khi đó, cho trước  $M_1$  và  $T_1$  đã biết, với  $T_1 = MAC(K, M_1)$ , bộ mã hóa (cryptanalyst) có thể thực hiện  $T_i = MAC(K_i, M_i)$  đối với tất cả các giá trị khóa  $k_i$  có thể có. Ít nhất một khóa đảm bảo tạo ra được giá trị trùng khớp  $T_i = T_1$ . Lưu ý rằng tổng số  $2^k$  nhãn sẽ được tạo ra, nhưng chỉ có  $2^n < 2^k$  giá trị nhãn khác nhau. Do vậy, một số khóa sẽ tạo ra nhãn đúng và kẻ tấn công không thể biết đâu là khóa đúng. Tính trung bình, tổng cộng  $2^{(k-n)}$  khóa sẽ tạo ra một trường hợp trùng khớp. Do đó, kẻ địch phải lặp lại các tấn công nhiều lần.

- **Vòng 1**

Cho trước:  $M_1, T_1 = MAC(K, M_1)$

Tính  $T_i = MAC(K, M_1)$  đối với  $2^k$  khóa

Số trường hợp trùng khớp  $\approx 2^{(k-n)}$

- **Vòng 2**

Cho trước:  $M_2, T_2 = MAC(K, M_2)$

Tính  $T_i = MAC(K, M_2)$  đối với  $2^{(k-n)}$  khóa nhận được từ Vòng 1

Số trường hợp trùng khớp  $\approx 2^{(k-2n)}$

Và cứ như thế. Tính trung bình,  $\alpha$  vòng sẽ cần đến  $k = \alpha \times n$ . Ví dụ, nếu một khóa 80 bit được sử dụng và nhãn là 32 bit, khi đó vòng đầu tiên sẽ sinh ra  $2^{48}$  trường hợp khóa có thể đúng. Vòng thứ hai sẽ thu hẹp các khóa có thể xảy ra còn khoảng  $2^{16}$  trường hợp khóa. Vòng thứ ba sẽ tạo ra chỉ 1 khóa duy nhất, là khóa được sử dụng bởi phía gửi.

Nếu chiều dài khóa là nhỏ hơn hoặc bằng chiều dài nhãn, khi đó có khả năng vòng đầu tiên sẽ tạo ra trường hợp khớp duy nhất. Có thể có nhiều hơn một khóa sẽ tạo ra sự

trùng khớp như vậy, trong trường hợp đó kẻ địch sẽ cần thực hiện cùng một kiểm tra trên cặp (bản tin, nhãn) mới.

Do vậy, một tấn công vét cạn cố gắng để khám phá khóa nhận thực cần không ít tính toán hơn so với việc yêu cầu khám phá khóa giải mã có cùng chiều dài. Tuy nhiên, các tấn công khác mà không yêu cầu khám phá khóa có thể xảy ra.

Xét thuật toán MAC sau đây. Đặt  $M = (X_1 \parallel X_2 \parallel \dots \parallel X_m)$  là một bản tin được xử lý như là sự kết hợp của các khối 64 bit  $X_i$ . Khi đó ta định nghĩa:

$$\Delta(M) = X_1 \oplus X_2 \oplus \dots \oplus X_m$$

$$\text{MAC}(K, M) = E(K, \Delta(M))$$

trong đó  $\oplus$  là phép OR loại trừ (XOR) và thuật toán mã hóa là DES. Do vậy, chiều dài khóa là 56 bit và chiều dài nhãn là 64 bit. Nếu một kẻ tấn công quan sát  $\{M \parallel \text{MAC}(K, M)\}$ , một tấn công vét cạn để xác định K sẽ cần  $2^{56}$  mã hóa. Nhưng kẻ địch có thể tấn công hệ thống bằng cách thay thế từ  $X_1$  tới  $X_m$  bằng bất kỳ giá trị mong muốn nào từ  $Y_1$  tới  $Y_{m-1}$  và thay thế  $X_m$  bằng  $Y_m$ :

$$Y_m = Y_1 \oplus Y_2 \oplus \dots \oplus Y_{m-1} \oplus \Delta(M)$$

Kẻ địch lúc này có thể gắn bản tin mới bao gồm  $Y_1$  tới  $Y_m$  với nhãn ban đầu để tạo thành một bản tin sẽ được chấp nhận là đã được xác thực bởi bên nhận. Với cách này, bất kỳ bản tin nào có chiều dài  $64 \times (m-1)$  bit cũng có thể được thêm vào một cách gian lận.

Do vậy, trong việc đánh giá tính an ninh của một hàm MAC, chúng ta cần xét các loại tấn công có thể được thực hiện. Từ đó, ta sẽ đưa ra các yêu cầu đối với hàm. Giả sử rằng một kẻ tấn công biết hàm MAC nhưng không biết K. Khi đó hàm MAC cần thỏa mãn những yêu cầu sau:

1. Nếu một kẻ tấn công quan sát  $M$  và  $\text{MAC}(K, M)$ , sẽ là không khả thi với kẻ tấn công nếu tạo một bản tin  $M'$  thỏa mãn:

$$\text{MAC}(K, M') = \text{MAC}(K, M)$$

2.  $\text{MAC}(K, M)$  nên có phân bố chuẩn đối với các bản tin được lựa chọn ngẫu nhiên,  $M$  và  $M'$ , xác suất để  $\text{MAC}(K, M) = \text{MAC}(K, M')$  là  $2^{-n}$ , trong đó  $n$  là số bit trong nhãn.
3. Cho  $M'$  là kết quả của sự chuyển đổi nào đó của  $M$ ,  $M' = f(M)$ . Ví dụ,  $f$  có thể là một số thao tác nghịch đảo một hoặc nhiều bit. Khi đó:
4.  $\Pr[\text{MAC}(K, M) = \text{MAC}(K, M')] = 2^{-n}$

Yêu cầu thứ nhất phát biểu đối cho ví dụ trước đó, trong đó một kẻ tấn công có khả năng tạo ra một bản tin mới để khớp với một nhãn cho trước, ngay cả khi kẻ tấn công không biết và không cần tìm hiểu khóa. Yêu cầu thứ hai dựa vào yêu cầu cần một tấn công vét cạn dựa trên bản rõ cho trước. Nghĩa là, nếu ta giả sử kẻ địch không biết  $K$  nhưng có thể tiếp cận được hàm MAC và có thể chỉ định các bản tin cho bộ tạo MAC, sau đó kẻ địch có thể thử nhiều bản tin khác nhau có tới khi tìm ra bản tin khớp với nhãn cho trước.

Yêu cầu cuối cùng đòi hỏi thuật toán nhận thực không nên yếu hơn ở một số phần hoặc bít nhất định so với các phần khác hoặc bit khác. Ngược lại, khi đó một kẻ tấn công có  $M$  và  $\text{MAC}(K, M)$  có thể thực hiện thay đổi đối với  $M$  tại điểm yếu và nhiều khả năng thành công sóm trong việc tạo ra bản tin khớp với nhãn cũ.

#### 4.2.4 Tính an toàn của MAC

Nếu chỉ dựa vào các thuật toán mã hóa và các hàm băm, ta có thể phân nhóm các tấn công vào MAC thành hai nhóm: Các tấn công vét cạn và phân tích mã hóa.

##### *Các tấn công vét cạn*

Một tấn công vét cạn đối với MAC khó thực hiện hơn so với tấn công vét cạn vào hàm băm bởi vì nó yêu cầu biết trước các cặp bản tin-nhãn. Để tấn công một mã băm, ta có thể làm theo cách sau: Cho trước một bản tin cố định  $x$  với mã băm  $n$  bit  $h=H(x)$ , một phương pháp vét cạn tìm kiếm xung đột là chọn một chuỗi bit ngẫu nhiên  $y$  và kiểm tra đẳng thức  $H(y)=H(x)$ . Kẻ tấn công có thể lặp lại phép thử này ngoại tuyến. Việc một tấn công ngoại tuyến có thể được sử dụng đối với thuật toán MAC hay không phụ thuộc vào kích thước tương đối của khóa và nhãn.

Để tiến hành, ta cần đưa ra đặc tính an ninh mong muốn của một thuật toán MAC được phát biểu như sau:

**Chống lại tính toán:** Cho trước một hoặc nhiều cặp bản rõ MAC,  $[x_i, \text{MAC}(K, x_i)]$ , việc tính bát kỳ cặp bản rõ MAC,  $[x, \text{MAC}(K, x)]$  đều là không khả dụng với bát kỳ đầu vào mới  $x \neq x_i$  nào.

Nói cách khác, kẻ tấn công muốn khám phá mã MAC có hiệu lực đối với bản tin  $x$  cho trước. Có hai loại tấn công có thể xảy ra: tấn công không gian khóa và tấn công giá trị MAC. Chúng ta lần lượt xem xét các kiểu tấn công này.

Nếu một kẻ tấn công có thể xác định khóa MAC, khi đó hắn có thể tạo một giá trị MAC hợp lệ đối với bất kỳ đầu vào  $x$  nào. Giả sử kích thước khóa bit và kẻ tấn công biết trước một cặp bản rõ-nhãm. Khi đó kẻ tấn công có thể tính toán nhãm  $n$  bit đối với bản rõ đã biết với tất cả các khóa có thể. Ít nhất một khóa đảm bảo tạo ra nhãm đúng, khóa đúng ban đầu được sử dụng để tạo ta cặp bản rõ-nhãm đã biết. Giai đoạn tấn công này sử dụng mức cố gắng tương đương với  $2^k$  (có nghĩa là, một thao tác thực hiện với mỗi một trong  $2^k$  giá trị khóa). Tuy nhiên, như đã được miêu tả trước đó, do MAC là ánh xạ nhiều-tới-một, có thể có các khóa khác tạo ra được đúng đầu ra. Do vậy, nếu có nhiều hơn một khóa được tìm thấy có thể tạo ra giá trị đúng, các cặp bản rõ-nhãm thêm vào cần phải được kiểm tra. Có thể thấy rằng mức độ nỗ lực là vào khoảng  $2^k$  lần thử.

Một kẻ tấn công cũng có thể tập trung vào nhãm mà không cần tạo lại khóa. Ở đây, mục tiêu là tạo ra nhãm hợp lệ đối với bản tin cho trước hoặc tìm ta một bản tin khớp với nhãm cho trước. Trong mỗi trường hợp, mức độ cố gắng là có thể so sánh được đối với trường hợp tấn công đặc tính một chiều hoặc chống chịu xung đột kém của mã băm, hoặc  $2^n$ . Trong trường hợp của MAC, tấn công không thể tiến hành trong điều kiện ngoại tuyến mà không có thêm đầu vào; kẻ tấn công sẽ cần các cặp bản rõ-nhãm được chọn hoặc thông tin về khóa.

Tổng kết, mức độ nỗ lực đối với tấn công vét cạn vào thuật toán MAC có thể được tính bằng  $\min(2^k, 2^n)$ . Độ mạnh được đánh giá là tương đương đối với các thuật toán mã hóa đối xứng nên lựa chọn hợp lý là chiều dài khóa và nhãm thỏa mãn mối quan hệ  $\min(k, n) \geq N$  với  $N$  nằm trong khoảng 128 bit.

### *Phân tích mã*

Như đối với các thuật toán mã hóa và các hàm băm, các tấn công phân tích mã đối với MAC tìm kiếm để khai thác đặc tính nào đó của thuật toán để thực hiện một tấn công nào đó được cho là phổ biến hơn là tìm kiếm vét cạn. Phương thức để đo lường khả năng chống chịu của một thuật toán MAC đối với tấn công phân tích mã là so sánh với nỗ lực yêu cầu đối với một tấn công vét cạn. Nghĩa là, một thuật toán MAC lý tưởng sẽ yêu cầu một nỗ lực phân tích mã lớn hơn hoặc bằng nỗ lực tấn công vét cạn.

Có nhiều biến thể trong cấu trúc của MAC hơn so với các hàm băm, do vậy việc tổng quát hóa các tấn công phân tích mã đối với MAC sẽ là khó khăn hơn. Hơn nữa, mới chỉ có ít nghiên cứu đã được thực hiện để phát triển các tấn công như vậy.

#### 4.2.5 MAC dựa trên hàm băm HMAC

Phần này chúng ta sẽ xem xét các ví dụ của MAC dựa trên việc sử dụng mã khối đối xứng. Theo truyền thống, đây là tiếp cận chung nhất để cấu trúc nên một MAC. Trong những năm gần đây, sự quan tâm đã tăng lên đối với việc phát triển một MAC nhận được từ hàm băm mã hóa. Động lực của mối quan tâm này là:

- + Các hàm băm mã hóa như là MD5 hay SHA nói chung thực hiện nhanh hơn trong phần mềm so với mã khối đối xứng như DES.
- + Thư viện mã cho các hàm băm mã hóa được phổ biến rộng rãi.

Với sự phát triển của AES và sự phổ biến của mã chương trình cho các thuật toán mã hóa, những vấn đề này trở nên ít quan trọng hơn, nhưng MAC dựa trên hàm băm tiếp tục được sử dụng rộng rãi.

Một hàm băm như SHA không được thiết kế cho việc sử dụng như là một MAC và không thể được sử dụng trực tiếp cho mục đích đó, bởi vì nó không dựa trên một khóa bí mật. Đã có một số đề xuất đối với việc liên kết một khóa bí mật với một thuật toán băm có sẵn. Tiếp cận đã nhận được nhiều sự ủng hộ nhất là HMAC. HMAC đã được đề xuất trong RFC 2104, đã được chọn như là MAC buộc phải thực hiện cho an ninh IP, và được sử dụng trong các giao thức Internet khác như là SSL. HMAC cũng được đề xuất là một chuẩn NIST.

##### *Các mục tiêu thiết kế HMAC*

RFC 2104 liệt kê các mục tiêu thiết kế sau đây của HMAC

- + Để sử dụng mà không cần sửa đổi các hàm băm có sẵn. Đặc biệt, để sử dụng các hàm băm thực hiện tốt trong phần mềm và mã chương trình là miễn phí và phổ biến.
- + Để cho phép khả năng thay thế các hàm băm được nhúng nếu các hàm băm khác nhanh hơn hoặc bảo mật hơn được tìm ra hoặc yêu cầu.
- + Để kế thừa hiệu năng ban đầu của hàm băm mà không gây ra sự suy giảm đáng kể.
- + Để sử dụng và xử lý khóa theo một cách đơn giản.
- + Để có được các phân tích mã hóa dễ hiểu về độ mạnh của cơ chế nhận thực dựa trên các giả thiết hợp lý về hàm băm được nhúng.

Hai mục tiêu đầu tiên là quan trọng đối với khả năng chấp nhận HMAC. HMAC coi hàm băm như là một “hộp đen” và đem lại hai lợi ích. Đầu tiên, một sự thực hiện có sẵn của một hàm băm có thể được sử dụng như là một mô-đun trong thực thi HMAC. Theo cách này, cụm mã HMAC được đóng gói trước và sẵn sàng được sử dụng mà không cần sửa đổi. Thứ hai, khi muốn thay thế một hàm băm cho trước trong sự thực thi HMAC, tất cả những gì được yêu cầu là loại bỏ mô-đun hàm băm có sẵn và đưa mô-đun mới vào. Điều này có thể được thực hiện nếu muốn một hàm băm nhanh hơn. Quan trọng hơn, nếu an ninh của hàm băm được nhúng bị xâm hại, an ninh của HMAC có thể được giữ lại chỉ đơn giản bằng cách thay thế hàm băm được nhúng bằng một hàm băm an ninh toàn hơn (ví dụ thay thế SHA-2 bằng SHA-3).

Mục tiêu thiết kế cuối cùng trong danh sách thực tế là ưu điểm chính của HMAC đối với các cơ chế dựa trên hàm băm. Dưới đây sẽ tóm tắt cấu trúc của HMAC.

### **Thuật toán HMAC**

Hình 4.6 minh họa hoạt động của HMAC. Định nghĩa các thuật ngữ sau:

$H$  = hàm băm được nhúng (ví dụ MD5, SHA-1, RIPEMD-160)

$IV$  = giá trị khởi tạo đầu vào của hàm băm

$M$  = bản tin đầu vào cho HMAC (bao gồm phần đệm xác định trong hàm băm được nhúng)

$Y_i$  = block thứ  $i$  của  $M$ ,  $0 \leq i \leq (L - 1)$

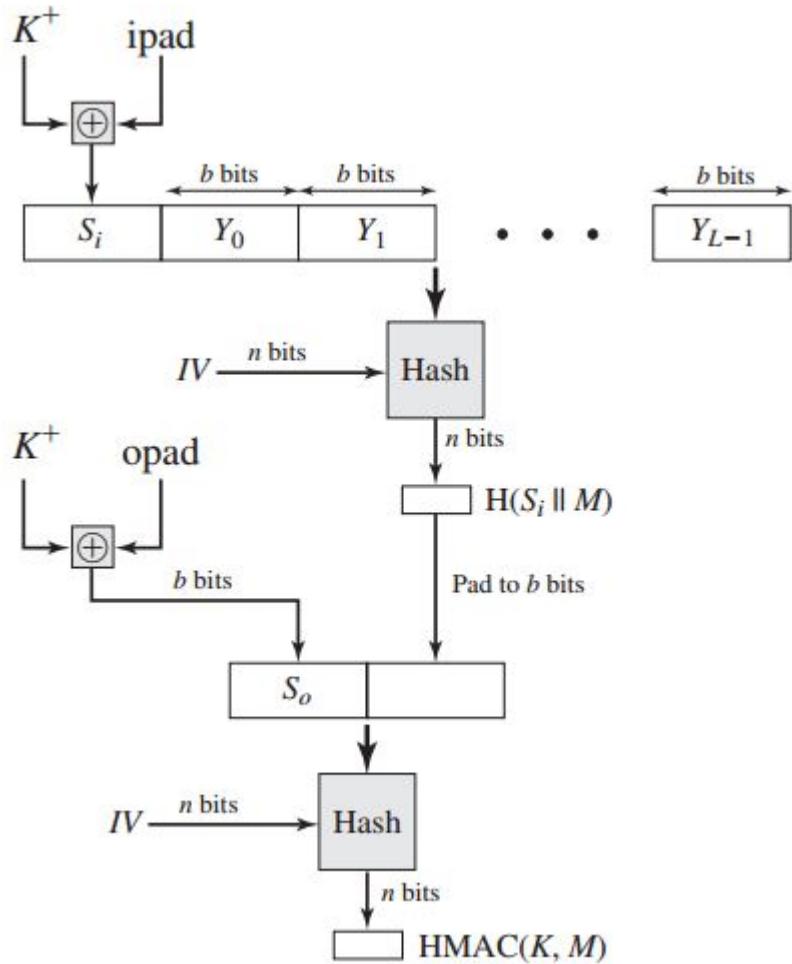
$L$  = số block trong  $M$

$b$  = số bit trong một block

$n$  = chiều dài mã băm được tạo bởi hàm băm được nhúng

$K$  = khóa bí mật; chiều dài được đề xuất  $\geq n$ ; nếu chiều dài khóa lớn hơn  $b$ , khóa là đầu vào của hàm băm để tạo ra một khóa  $n$  bit

$K^+$  =  $K$  bit được đếm vào phía trái để được chiều dài  $b$  bit



Hình 4.6: Cấu trúc HMAC

Ipad = 00110110 (36 trong hệ hexa) được lặp lại b/8 lần

Opad = 01011100 (5C trong hệ hexa) được lặp lại b/8 lần

Khi đó HMAC có thể được biểu diễn như sau:

$$\text{HMAC}(K, M) = \text{H}[(K^+ \oplus \text{opad}) \parallel \text{H}[(K^+ \oplus \text{ipad}) \parallel M]]$$

Ta có thể miêu tả thuật toán như sau:

- (1) Thêm các số 0 vào phía cuối bên trái của K để tạo một chuỗi b bit  $K^+$  (ví dụ nếu K có chiều dài 160 bit và  $b = 512$  khi đó cần thêm 44 số 0 vào K).
- (2) XOR  $K^+$  với ipad để tạo khối b bit  $S_i$ .
- (3) Thêm M vào  $S_i$ .
- (4) Sử dụng hàm H cho luồng được tạo ở bước 3.

(5) XOR  $K^+$  với opad để tạo khối b bit  $S_o$ .

(6) Thêm giá trị băm nhận được ở bước 4 vào  $S_o$ .

(7) Sử dụng hàm H với luồng được tạo trong bước 6 và đưa ra kết quả.

Lưu ý rằng kết quả của phép XOR với ipad làm đảo một nửa số bit của  $K$ . Tương tự, phép XOR với opad làm đảo một nửa số bit của  $K$ , với một tập bit khác. Trong thực tế, bằng cách đưa  $S_i$  và  $S_o$  qua hàm nén của thuật toán băm, ta đã tạo hai khóa từ  $K$  theo phương thức ngẫu nhiên giả tạp âm.

HMAC nên thực hiện trong thời gian xấp xỉ thời gian hàm băm nhúng đối với các bản tin dài. HMAC thêm ba lệnh của hàm nén băm (với  $S_i$  và  $S_o$  và khối được tạo ra từ giá trị băm bên trong).

Có thể thực hiện hiệu quả hơn, như trình bày trong hình 4.7. Hai đại lượng được tính trước:

$$\begin{aligned} f(IV, (K^+ \oplus \text{ipad})) \\ f(IV, (K^+ \oplus \text{ipad})) \end{aligned}$$

Trong đó  $f(cv, block)$  là hàm nén đối với hàm băm được đóng vai trò như các đối số một biến chuỗi  $n$  bit và một khối  $b$  bit và tạo ra một biến chuỗi  $n$  bit. Các đại lượng này chỉ cần được tính lúc ban đầu và mỗi khi khóa thay đổi. Trên thực tế, các đại lượng được tính trước thay thế cho giá trị khởi tạo (IV) trong hàm băm. Với sự thực hiện này, chỉ một trường hợp bổ sung của hàm nén được thêm vào quá trình thông thường được sinh ra bởi hàm băm. Sự thực hiện hiệu quả hơn này đặc biệt có giá trị nếu hầu hết các bản tin cần tính MAC đều là ngắn.

### *An ninh của HMAC*

An ninh của bất kỳ hàm MAC nào dựa trên hàm băm nhúng đều phụ thuộc vào độ mạnh mã hóa của hàm băm bên dưới theo một cách nào đó. Sự ủng hộ HMAC là do người thiết kế ra nó có khả năng thể chứng minh mối quan hệ chính xác giữa độ mạnh của hàm băm được nhúng và độ mạnh của HMAC.

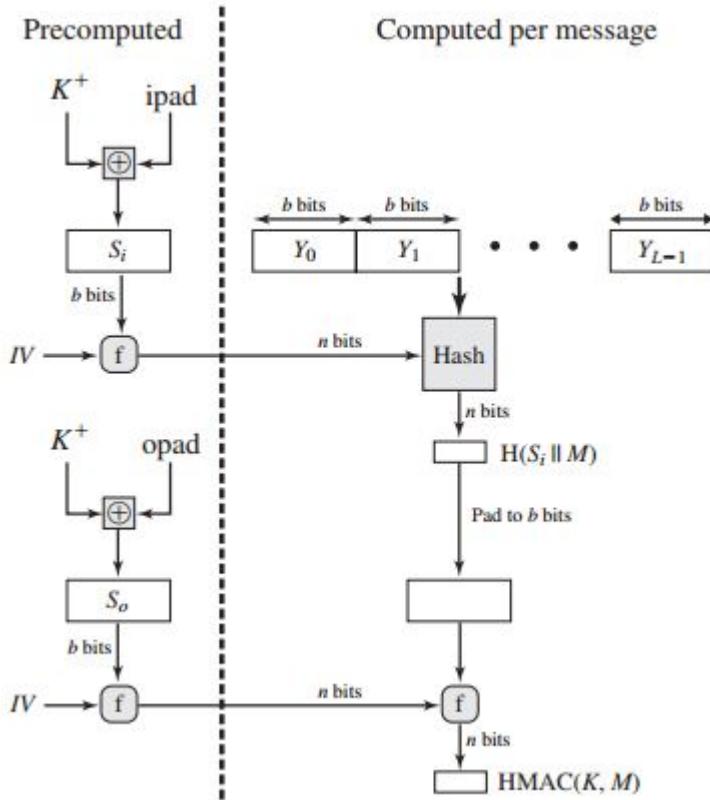
An ninh của một hàm HMAC được biểu diễn tổng quát theo xác suất giả mạo thành công với cùng khoảng thời gian cho trước và số cặp bản tin-mã cho trước được tạo với cùng một khóa. Về cơ bản, đã chứng minh được rằng với một mức độ nỗ lực cho trước (thời gian, các cặp bản tin-nhận) đối với các bản tin được tạo bởi một người dùng hợp

pháp và được theo dõi bởi một kẻ tấn công, xác suất tấn công vào HMAC thành công tương đương với một trong các tấn công sau đây vào hàm băm được nhúng:

- + Kẻ tấn công có khả năng tính toán một đầu ra của hàm néng ngay cả với một IV ngẫu nhiên, bí mật và kẻ tấn công không biết trước.
- + Kẻ tấn công tìm các xung đột trong hàm băm ngay cả khi IV là ngẫu nhiên và bí mật.

Trong tấn công đầu, ta có thể xem hàm néng tương đương với hàm băm áp dụng đối với một bản tin bao gồm một khối  $b$  bit đơn. Với tấn công này, IV của hàm băm được thay thế bởi một giá trị bí mật, ngẫu nhiên  $n$  bit. Một tấn công vào hàm băm này cần có tấn công vét cạn vào khóa – có mức độ nỗ lực  $2^n$ , hoặc tấn công ngày sinh – là trường hợp đặc biệt của tấn công thứ hai, sẽ được thảo luận sau đây.

Trong tấn công thứ hai, kẻ tấn công tìm kiếm hai bản tin  $M$  và  $M'$  tạo ra cùng giá trị băm  $H(M) = H(M')$  còn gọi là tấn công ngày sinh. Tấn công này yêu cầu nỗ lực  $2^{n/2}$  đối với giá trị băm có chiều dài  $n$ . Trên cơ sở đó, an ninh của MD5 được xem xét lại, vì một mức độ nỗ lực  $2^{64}$  có vẻ khả thi với công nghệ hiện nay. Liệu điều này có nghĩa là một hàm băm 128 bit như MD5 là không phù hợp với HMAC? Câu trả lời là không, vì lý do sau đây. Để tấn công MD5, kẻ tấn công có thể chọn bất một tập bản tin nào và làm việc với chúng ngoại tuyến trên phương tiện tính toán chuyên dụng để tìm ra một xung đột. Do kẻ tấn công biết thuật toán băm và giá trị IV mặc định, kẻ tấn công có thể tạo mã băm với mỗi bản tin mà kẻ tấn công tạo ra. Tuy nhiên, khi tấn công HMAC, kẻ tấn công không thể tạo các cặp bản tin/mã ngoại tuyến do kẻ tấn công không biết  $K$ . Do đó, kẻ tấn công phải quan sát một chuỗi các bản tin được tạo bởi HMAC dưới cùng một khóa và thực hiện tấn công vào các bản tin đã biết này. Với chiều dài mã băm 128 bit, nó yêu cầu  $2^{64}$  khối được quan sát ( $2^{72}$  bit) được tạo sử dụng cùng một khóa. Trên một đường truyền tốc độ 1 Gb/giây, một người sẽ cần quan sát một luồng các bản tin liên tục không có sự thay đổi về khóa trong khoảng 150 nghìn năm để có thể thành công. Do vậy, nếu tốc độ được xét tới, có thể hoàn toàn chấp nhận được việc sử dụng MD5 thay vì SHA-1 làm hàm băm nhúng cho HMAC.



Hình 4.7: Sự thực hiện HMAC hiệu quả

Đối với phương pháp mã khóa đối xứng, các thành viên chia sẻ cùng một khóa và khóa được bảo vệ bởi các thành phần ngoài. Hơn nữa, khóa được thay đổi thường xuyên nhằm tránh các tấn công. Vì vậy, sức mạnh của bất kỳ một hệ thống mã hóa nào cũng đều liên quan tới kỹ thuật phân phối khóa, một thuật ngữ sử dụng để tạo ra khóa giữa hai thành viên đảm bảo tính bí mật với các bên khác.

### MAC dựa trên mật mã khóa

Phần này xem xét hai cơ chế hoạt động của MAC trên cơ sở sử dụng mã khóa. Đầu tiên, chúng ta đi vào thuật toán xác thực dữ liệu DAA (Data Authentication Algorithm) là một thuật toán cũ và sau đó tập trung vào thuật toán CMAC, được thiết kế với mục đích khắc phục những nhược điểm của DAA.

#### Thuật toán nhận thực dữ liệu DAA

Thuật toán xác thực dữ liệu DAA dựa trên DES là một trong những thuật toán phổ biến dựa trên FIPS (FIPS PUB 113) và chuẩn ANSI (X9.17). Tuy nhiên, mật bảo mật của thuật toán này đã bị khai thác cho nên chúng dần được thay thế bằng các thuật toán mới, mạnh mẽ hơn.

Thuật toán DAA được tạo ra bằng cách sử dụng chế độ chuỗi mã khối CBC trong hoạt động của DES với vector khởi tạo zero. Dữ liệu (ví dụ như tin nhắn, tệp, chương trình) cần nhận thực được nhóm vào các khối 64 bit liên tiếp nhau:  $D_1, D_2, \dots, D_N$ . Nếu cần, khối cuối bao gồm 64 bit zero được chèn vào bên phải chuỗi. Sử dụng thuật toán mã hóa tiêu chuẩn E và khóa bí mật  $K$ , mã nhận thực dữ liệu DAC (Data Authentication Code) được tính toán như sau (hình 4.8)

$$O_1 = E(K, D_1)$$

$$O_2 = E(K, [D_2 \oplus O_1])$$

$$O_3 = E(K, [D_3 \oplus O_2])$$

.

.

$$O_N = E(K, [D_N \oplus O_{N-1}])$$

DAC hoặc có thể là toàn bộ khối  $O_N$  hoặc là  $M$  bit bên trái của khối  $O_N$ , với  $16 \leq M \leq 64$ .

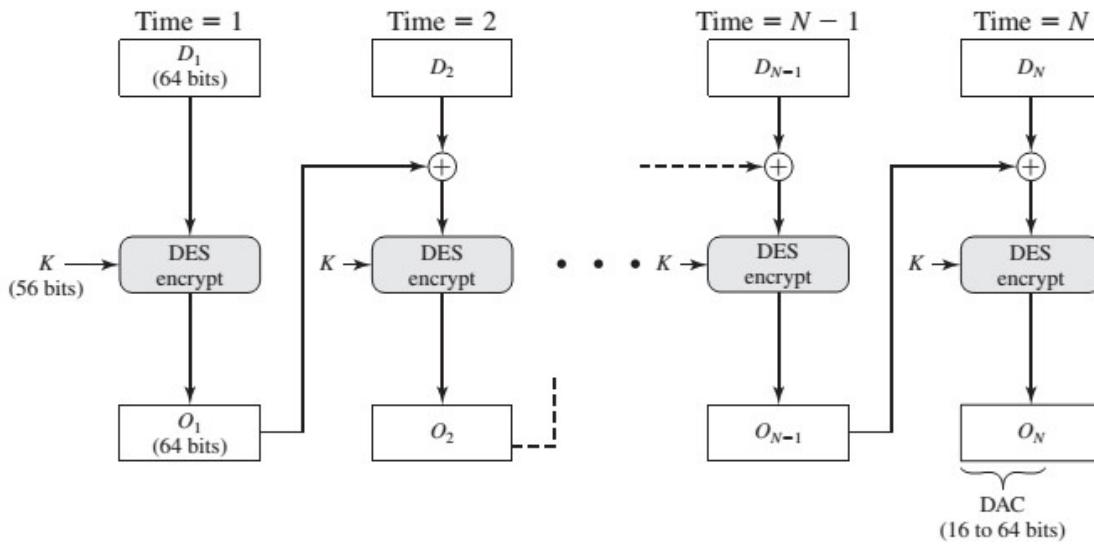
#### CMAC

Như phần trên đã trình bày, DAA được sử dụng rộng rãi trong khi MAC được chứng minh rằng có thể đáp ứng các tiêu chí an ninh với một số hạn chế nhất định. Chỉ các bản tin có chiều dài cố định  $m \times n$  bit được xử lý, với  $n$  là kích thước khối mã và  $m$  là số nguyên dương cố định. Một ví dụ đơn giản, thực hiện CBC MAC cho một khối bản tin X bằng phép  $T = MAC(K, X)$ , kẻ tấn công ngay lập tức biết được CBC MAC cho hai khối bản tin  $X \parallel (X \oplus T)$  khi thực hiện phép biến đổi  $T$  một lần nữa.

Tuy nhiên, hạn chế này có thể được khắc phục bằng cách sử dụng ba khóa: Một khóa  $K$  chiều dài  $k$  được dùng tại mỗi bước của chuỗi mã khối và hai khóa còn lại có chiều dài  $b$  bằng với chiều dài của khối mã. Cơ chế này được đề xuất cụ thể với hai khóa  $n$ -bit dùng để tạo một mã khóa duy nhất thay vì sử dụng đơn lẻ.

Đầu tiên, chúng ta xem xét hoạt động của CMAC khi bản tin là tập hợp  $n$  khối mã, mỗi khối có chiều dài  $b$ . Với AES,  $b = 128$  và với triple DES,  $b = 64$ . Bản tin được chia thành  $n$  khối ( $M_1, M_2, \dots, M_n$ ). Thuật toán sử dụng  $k$ -bit của mã khóa  $K$  và  $b$ -bit cố định

$K_1$ . Với AES, kích thước khóa  $k$  thường là 128, 192 hoặc 256 bit; với triple DES, kích thước khóa hoặc 128 hoặc 168 bit. CMAC được tính như sau (hình 4.8)



Hình 4.8: Thuật toán nhận thực dữ liệu

$$C_1 = E(K, M_1)$$

$$C_2 = E(K, [M_2 \oplus C_1])$$

$$C_3 = E(K, [M_3 \oplus C_2])$$

.

.

$$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])$$

$$T = \text{MSB}_{Tlen}(C_n)$$

Với

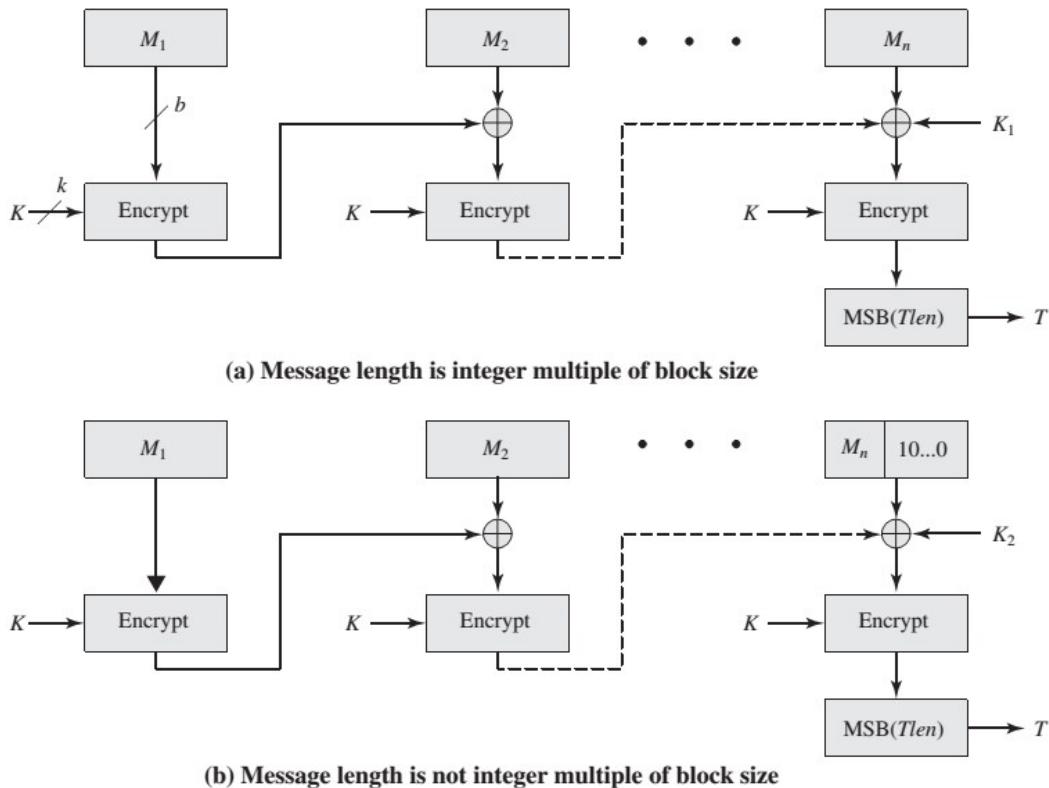
$$T = \text{Mã nhận thực bản tin}$$

$$Tlen = \text{Độ dài của } T \text{ (bit)}$$

$$\text{MSB}_s(X) = s \text{ bit bên trái của chuỗi } X$$

Nếu bản tin không chia được thành một số nguyên chiềudài khốimã, khi đó khốicuối cùng sẽđược chènvào phái bên phải (least significant bits)với môt bít 1 và  $(b - 1)$

bit còn lại bằng 0. Hoạt động sau đó của CMAC được xử lý như trước, thay sử dụng  $b$ -bit  $K_2$  khác nhau bằng  $K_1$ .



Hình 4.9: Mã hóa nhận thực bản tin dựa trên mật mã

Hai khóa  $b$ -bit được đưa ra từ  $k$ -bit của khóa mã như sau

$$L = E(K, 0^b)$$

$$K_1 = L \cdot x$$

$$K_2 = L \cdot x^2 = (L \cdot x) \cdot x$$

ở đây nhân tử  $(.)$  được thực hiện trong trường hữu hạn  $GF(2^b)$ ;  $x$  và  $x^2$  là các đa thức bậc nhất và bậc hai trong các thành phần của  $GF(2^b)$ . Do đó, bit nhị phân biểu diễn cho  $x$  bao gồm  $(b - 2)$  số zero; bit nhị phân biểu diễn cho  $x^2$  bao gồm  $(b - 3)$  số zero. Với hai kích thước được chấp nhận như trên, các đa thức bao gồm  $x^{64} + x^4 + x^3 + x + 1$  và  $x^{128} + x^7 + x^2 + x + 1$ .

Để tạo khóa  $K_1$  và  $K_2$ , mã khôi được sử dụng bao gồm toàn bộ bit 0. Khóa con đầu tiên có nguồn gốc từ kết quả bản mã bằng cách dịch trái một bit và thực hiện phép XOR

phụ thuộc vào kích thước khôi mã. Khóa con thứ hai cũng được đưa ra như cách xây dựng khóa con đầu tiên.

#### 4.2.7 Mật mã được xác thực

Mã hóa xác thực AE (Authenticated Encryption) là một thuật ngữ dùng để mô tả các hệ thống mã hóa với mục đích bảo vệ đồng thời cả tính bảo mật và xác thực (toàn vẹn) cho quá trình truyền thông. Nhiều ứng dụng và giao thức yêu cầu cả hai hình thức an ninh này, nhưng cho đến gần đây hai dịch vụ này đã được thiết kế riêng biệt. Có bốn phương pháp chung để cung cấp cả mật và mã hóa cho một bản tin M.

- **Băm sau khi mã hóa ( $H \rightarrow E$ ):** Đầu tiên tính toán mã hàm băm trên bản tin M bằng  $h = H(M)$ . Sau đó thực hiện mã hóa bản tin đã được thêm hàm băm:  $E(K, (M \parallel h))$ .
- **Xác thực sau khi mã hóa ( $A \rightarrow E$ ):** Sử dụng hai khóa. Đầu tiên nhận thực bản rõ bằng tính toán giá trị MAC bởi  $T = MAC(K_1, M)$ . Sau đó mã hóa bản tin được thêm mã xác thực bản tin:  $E(K_2, [M \parallel T])$ . Tiếp cận này được thực hiện bởi các giao thức SSL/TLS.
- **Mã hóa sau khi nhận thực ( $E \rightarrow A$ ):** Sử dụng hai khóa. Đầu tiên mã hóa bản tin thành bản mã  $C = E(K_2, M)$ . Sau đó xác thực bản mã với  $T = MAC(K_1, M)$  để tạo thành cặp  $(C, T)$ . Tiếp cận này được sử dụng trong giao thức IPSec.
- **Độc lập mã hóa và nhận thực ( $E + A$ ):** Sử dụng hai khóa. Mã hóa bản tin thành bản mã  $C = E(K_2, M)$ . Xác thực bản rõ với  $T = MAC(K_1, M)$  để tạo thành cặp  $(C, T)$ . Hoạt động này có thể được thực hiện độc lập và được ứng dụng trong giao thức SSH.

Giải mã và xác thực hoàn toàn được minh bạch trong mỗi phương pháp. Với  $H \rightarrow E$ ,  $A \rightarrow E$  và  $E + A$ , giải mã trước mới xác nhận. Với  $E \rightarrow A$ , tiến hành xác nhận trước mới giải mã. Cả bốn phương pháp này đều tồn tại lỗ hổng an ninh. Tiếp cận theo  $H \rightarrow E$  được sử dụng trong giao thức WEP nhằm bảo vệ mạng WiFi. Phương pháp này có những nhược điểm cơ bản dẫn đến sự thay thế của giao thức WEP bằng WPA. Tuy vậy, bất kỳ phương pháp nào cũng có thể cung cấp một mức độ an ninh cao nếu có thiết kế phù hợp. Đây là mục tiêu của hai tiếp cận được thảo luận trong phần này, cả hai đã được tiêu chuẩn hóa bởi NIST.

Bộ đếm với CBC-MAC

Cơ chế bộ đếm với bản tin mã hóa theo chuỗi khồi CCM (Counter with Cipher Block Chaining - Message) được chuẩn hóa bởi NIST để hỗ trợ các yêu cầu an ninh cho mạng cục bộ IEEE 802.11 WiFi, nhưng chúng có thể được sử dụng trong bất kỳ ứng dụng mạng nào yêu cầu mã hóa và nhận thực. CCM là một biến thể của phương pháp mã hóa MAC nhằm thực hiện mã hóa nhận thực, được định nghĩa trong NIST SP 800-38C.

Các thành phần then chốt của thuật toán CCM là thuật toán mã hóa AES, chế độ hoạt động CTR và thuật toán nhận thực CMAC. Khóa riêng  $K$  được dùng cả mã hóa và thuật toán MAC. Đầu vào của quá trình mã hóa CCM bao gồm ba phần tử:

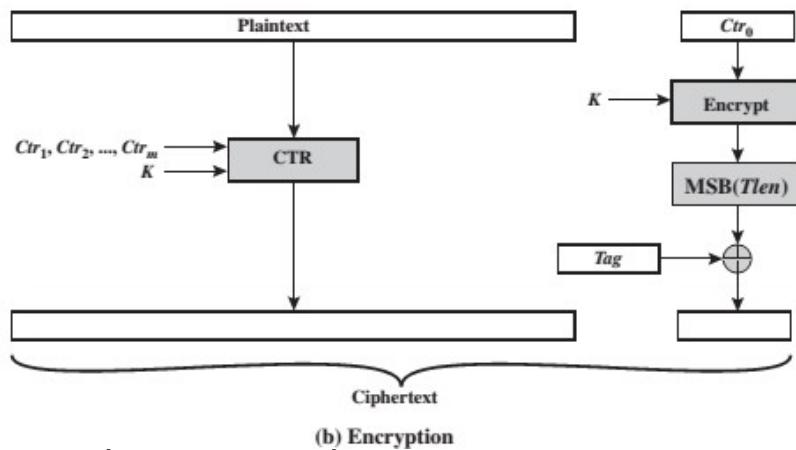
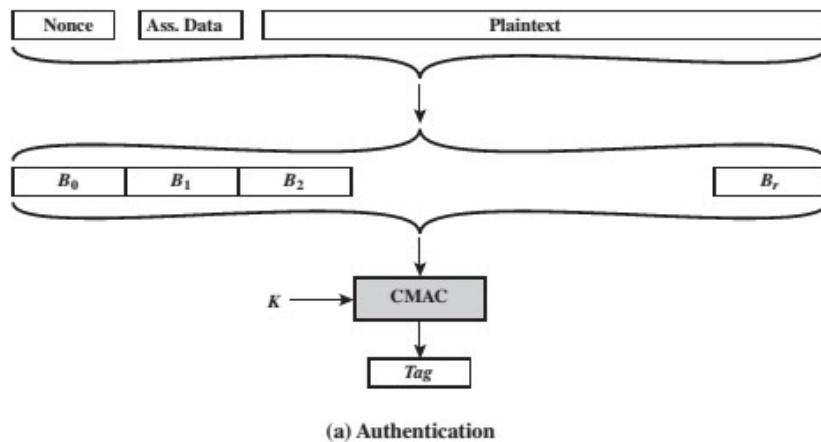
1. Dữ liệu sẽ được nhận thực và mã hóa. Đây là bản rõ P của khồi dữ liệu.
2. Dữ liệu liên quan A sẽ được nhận thực nhưng không được mã hóa. Ví dụ như tiêu đề của giao thức được truyền chính xác cho đúng loại giao thức sẽ cần phải nhận thực.
3. Một nonce  $N$  được ánh định cho tải và dữ liệu liên quan. Đây là một chuỗi duy nhất và khác nhau trong mỗi trường hợp trong toàn bộ xuyên suốt quá trình liên kết và nhằm ngăn chặn các cuộc tấn công lặp và một số loại tấn công khác.

Hình 4.10 minh họa hoạt động của CCM. Đối với nhận thực, đầu vào bao gồm nonce, dữ liệu liên quan và bản rõ. Đầu vào này được định dạng như một chuỗi các khồi từ  $B_0$  đến  $B_r$ . Khối đầu tiên bao gồm nonce được thêm một số bit định dạng biểu thị độ dài của các phần tử N, A và P. Tiếp theo sau là một lượng không âm số khồi chứa A và một lượng không âm số khồi chứa P. Chuỗi kết quả của khồi phục vụ như một đầu vào của thuật toán CMAC, tạo ra giá trị MAC có chiều dài Tlen bé hơn hoặc bằng chiều dài khồi (Hình 4.10a).

Với mã hóa, một chuỗi của bộ đếm được tạo gần như độc lập với nonce. Mã nhận thực được mã hóa trong chế độ CTR sử dụng bộ đếm đơn Ctr0. Các bit quan trọng nhất MSB (Most Significant bits) ở đầu ra được XOR với mã nhận thực để tạo thành một thẻ mã hóa. Phần còn lại của bộ đếm được sử dụng cho chế độ mã hóa CTR của bản rõ. Bản rõ đã mã hóa được nối với thẻ mã hóa để tạo thành các đầu ra cho bản mã (Hình 4.10b). SP 800-38C định nghĩa quá trình nhận thực/mã hóa như sau

1. Áp dụng hàm định dạng gồm  $(N, A, P)$  để tạo các khồi  $B_0, B_1, \dots, B_r$ .
2. Thiết lập  $Y_0 = E(K, B_0)$ .
3. Với  $i = 1$  tới  $r$ , thực hiện  $Y_i = E(K, (B_i \oplus Y_{i-1}))$ .
4. Thiết lập  $T = \text{MSB}_{Tlen}(Y_r)$ .

5. Áp dụng hàm khởi tạo bộ đếm để tạo ra các khối đếm  $Ctr_0, Ctr_1, \dots, Ctr_m$  với  $m = \lceil Plen / 128 \rceil$ .
6. Với  $j = 0$  tới  $m$ , thực hiện  $S_j = E(K, Ctr_j)$ .
7. Thiết lập  $S = S_1 \parallel S_1 \parallel \dots \parallel S_m$ .
8.  $C = (P \oplus \text{MSB}_{Plen}(S)) \parallel (T \oplus \text{MSB}_{Tlen}(S_0))$ .



Hình 4.10: Bộ đếm với chuỗi khối mã hóa - mã nhận thực bản tin

Đối với giải mã và xác thực, phía nhận yêu cầu các đầu vào sau: Bản mã C, nonce N, dữ liệu liên quan A, khóa K, bộ đếm khởi tạo Ctr0. Các bước thực hiện như sau:

1. Nếu  $Clen \leq Tlen$ , trả về INVALID.
2. Sử dụng hàm khởi tạo bộ đếm để tạo các khối đếm  $Ctr_0, Ctr_1, \dots, Ctr_m$  với  $m = \lceil Clen / 128 \rceil$ .
3. Với  $j = 0$  tới  $m$ , thực hiện  $S_j = E(K, Ctr_j)$ .
4. Thiết lập  $S = S_1 \parallel S_1 \parallel \dots \parallel S_m$ .

5. Thiết lập  $P = \text{MSB}_{Clen-Tlen}(C) \oplus \text{MSB}_{Clen-Tlen}(S)$ .
6. Thiết lập  $T = \text{LSB}_{Tlen}(C) \oplus \text{MSB}_{Tlen}(S_0)$ .
7. Áp dụng hàm định dạng  $(N, A, P)$  để tạo các khối  $B_0, B_1, \dots, B_r$ .
8. Thiết lập  $Y_0 = E(K, B_0)$ .
9. Với  $i = 1$  tới  $r$ , thực hiện  $Y_i = E(K, (B_i \oplus Y_{i-1}))$ .
10. Nếu  $T \neq \text{MSB}_{Tlen}(Y_r)$  thì trả kết quả INVALID, nếu không trả kết quả  $P$ .

CCM là một thuật toán tương đối phức tạp. Lưu ý rằng nó đòi hỏi hai công việc trên các bản rõ, một lần để tạo ra giá trị MAC, và một lần để mã hóa. Hơn nữa, các chi tiết của các đặc điểm kỹ thuật yêu cầu một sự cân bằng giữa độ dài của nonce và độ dài của thẻ mã dù đây là một hạn chế không cần thiết. Cũng lưu ý rằng khóa mã được sử dụng hai lần trong chế độ mã hóa CTR: một lần để tạo ra thẻ mã và một lần để mã hóa bản rõ cộng với thẻ mã. Cho dù những sự phức tạp thêm vào sự an toàn của thuật toán là không rõ ràng thì trong mọi trường hợp, các phân tích cũng kết luận rằng CCM cung cấp một mức độ bảo mật cao.

#### *Chế độ Galois/bộ đếm*

Chế độ GCM (Galois/Counter Mode) được chuẩn hóa bởi NIST trong NIST SP 800-38D được thiết kế song song để cung cấp thông lượng cao với chi phí và trễ thấp. Về bản chất, bản tin được mã hóa trong biến thể của chế độ CTR. Kết quả của bản mã được nhân với thông tin về khóa và chiều dài bản tin thông qua  $\text{GF}(2^{128})$  để tạo thẻ nhận thực. Các tiêu chuẩn cũng quy định một chế độ hoạt động chỉ cung cấp cho MAC, được gọi là GMAC.

Chế độ GMAC tận dụng hai hàm: GHASH là một khóa hàm băm, và GCTR là chế độ thiết yếu với bộ đếm được xác định bởi một số gia đơn giản trong một thao tác.

GHASH(X) có các đầu vào là khóa băm H và một chuỗi bit X có  $\text{len}(X) = 128m$  bit ( $m$  nguyên dương) để tạo ra một giá trị MAC 128 bit. Hàm có thể được xác định như sau (Hình 4.11a)

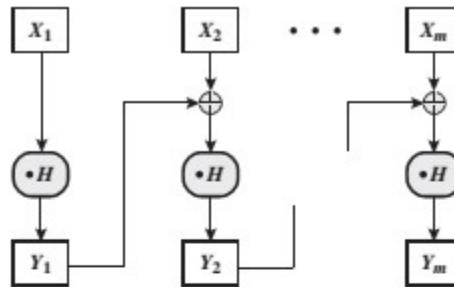
1. Gọi  $X_1, X_2, \dots, X_{m-1}, X_m$  là một chuỗi duy nhất gồm các khối và  $X = X_1 \parallel X_2 \parallel \dots \parallel X_{m-1} \parallel X_m$ .
2. Gọi  $Y_0$  là một khối gồm 128 bit 0, biểu diễn bằng  $0^{128}$ .
3. Với  $i = 1, \dots, m$ , đặt  $Y_i = (Y_{i-1} \oplus X_i) \square H$ , với  $\square$  là phép nhân trong  $\text{GF}(2^{128})$ .
4. Trả về  $Y_m$ .

Hàm GHASH<sub>H</sub>(X) có thể xây dựng bằng

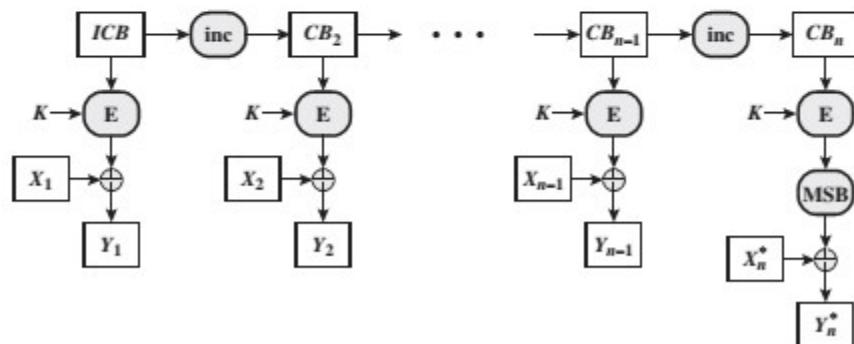
$$(X_1 \square H^m) \oplus (X_2 \square H^{m-1}) \oplus \dots \oplus (X_{m-1} \square H^2) \oplus (X_m \square H).$$

Nếu các khóa băm giống nhau được sử dụng để nhận thực nhiều bản tin, khi đó giá trị  $H^2, H^3, \dots$  có thể tính toán trước khi sử dụng cho mỗi bản tin đã được nhận thực ngay trước đó. Sau đó các khối dữ liệu đã được nhận thực ( $X_1, X_2, \dots, X_m$ ) có thể được xử lý song song bởi các quá trình tính toán cho mỗi khối là độc lập.

GCTR<sub>K</sub>(ICB, X) có đầu vào là một khóa bí mật K và một chuỗi bit X có độ dài tùy ý, kết quả trả ra là một bản mã Y có độ dài bằng len(X). Hàm có thể được xác định như sau (Hình 4.11b).



(a)  $\text{GHASH}_H(X_1 \parallel X_2 \parallel \dots \parallel X_m) = Y_m$



(b)  $\text{GCTR}_K(ICB, X_1 \parallel X_2 \parallel \dots \parallel X_n^*) = Y_1 \parallel Y_2 \parallel \dots \parallel Y_n^*$

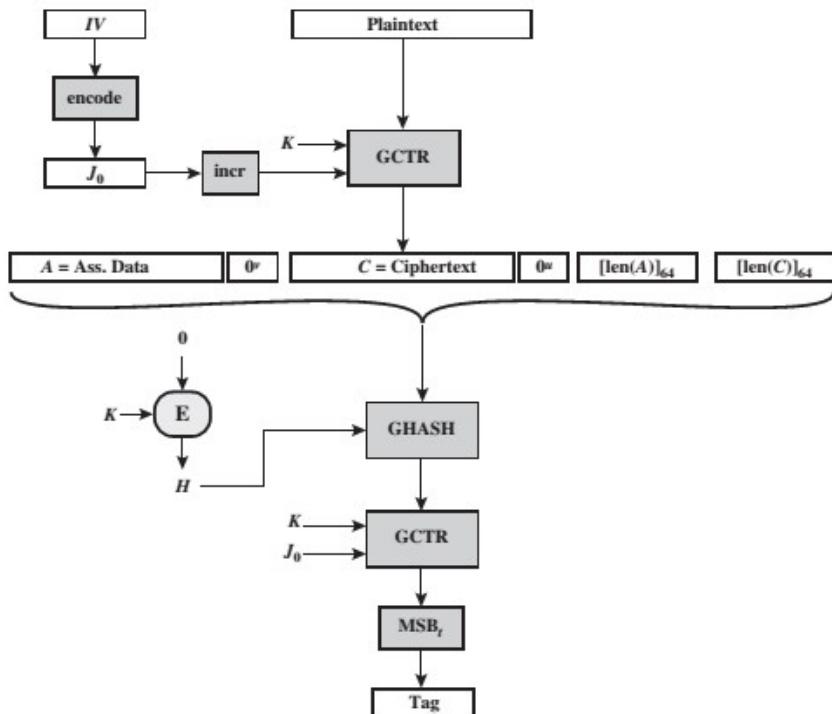
Hình 4.11: Chức năng mã hóa và nhận thực GCM

1. Nếu X là chuỗi rỗng, trả về một chuỗi rỗng Y.
2. Đặt  $n = \lceil (\text{len}(X) / 128) \rceil$ , ở đây n là số nguyên nhỏ nhất lớn hơn hoặc bằng  $\text{len}(X)/128$ .
3. Gọi  $X_1, X_2, \dots, X_{n-1}, X_n^*$  là các chuỗi 128 bit duy nhất và  $X = X_1 \parallel X_2 \parallel \dots \parallel X_{n-1} \parallel X_n^*$ .
4. Đặt  $CB_1 = ICB$ .

5. Với  $i = 2$  đến  $n$ , đặt  $CB_i = \text{inc}_{32}(CB_{i-1})$ , trong đó hàm  $\text{inc}_{32}(S)$  là phép tăng 32 bit bên phải của S lên 1 đơn vị, và các bit còn lại không đổi.
6. Với  $i = 1$  đến  $n - 1$ , thực hiện  $Y_i = X_i \oplus E(K, CB_i)$ .
7. Đặt  $Y_n^* = X_n^* \oplus \text{MSB}_{\text{len}(X_n^*)}(\text{E}(K, CB_n))$ .
8. Đặt  $Y = Y_1 \parallel Y_2 \parallel \dots \parallel Y_{n-1} \parallel Y_n^*$ .
9. Trả kết quả  $Y$ .

Chú ý rằng giá trị của bộ đếm có thể tạo nhanh và quá trình mã hóa này có thể thực hiện song song.

Có thể định nghĩa toàn bộ quá trình mã hóa nhận thực như trong hình 4.12. Đầu vào bao gồm một khóa bí mật K, vector khởi tạo IV, bản rõ P và dữ liệu xác thực bổ sung A. Ký hiệu  $[x]_s$  có nghĩa là s-bit nhị phân biểu diễn cho số nguyên không âm x.



Hình 4.12: Bộ đếm galois- Mã nhận thực bản tin

Các bước được thực hiện như sau

1. Đặt  $H = E(K, 0^{128})$ .
2. Định nghĩa khối  $J_0$  sao cho  
Nếu  $\text{len}(\text{IV}) = 96$ , thì  $J_0 = IV \parallel 0^{31} \parallel 1$ .

Nếu  $\text{len}(\text{IV}) \neq 96$ , thì  $J_0 = \text{GHASH}_H(\text{IV} \parallel 0^{s+64} \parallel [\text{len}(\text{IV})]_{64})$  với  
 $s = 128\lceil \text{len}(\text{IV}) / 128 \rceil - \text{len}(\text{IV})$ .

3. Đặt  $C = \text{GCTR}_K(\text{inc}_{32}(J_0), P)$ .
4. Đặt  $u = 128\lceil \text{len}(C) / 128 \rceil - \text{len}(C)$  và  $v = 128\lceil \text{len}(A) / 128 \rceil - \text{len}(A)$ .
5. Định nghĩa khối  $S$  với  

$$S = \text{GHASH}_H(A \parallel 0^v \parallel C \parallel 0^u \parallel [\text{len}(A)]_{64} \parallel \text{len}(C)_{64}).$$
6. Đặt  $T = \text{MSB}_t(\text{GCTR}_K(J_0, S))$  với  $t$  là độ dài thẻ hỗ trợ.
7. Trả về  $(C, T)$

Trong bước 1, khóa băm được tạo bằng cách mã hóa khối toàn bit 0 với khóa bí mật K. Ở bước 2, bộ đếm trước khối ( $J_0$ ) được tạo từ vector khối tạo IV. Đặc biệt, với độ dài IV bằng 96 bit, khi đó chuỗi đếm  $0^{31} \parallel 1$  được thêm vào IV để tạo thành khối đếm trước. Nếu không, vector IV được đếm vào một số lượng tối thiểu bit 0 (có thể không cần) do đó độ dài của chuỗi thu được là bội số của 128 bit (kích thước khối); chuỗi này lần lượt được nối thêm 64 bit 0, theo sau 64 bit biểu diễn cho độ dài vector IV, và hàm GHASH được áp dụng cho chuỗi thu được để hình thành khối đếm trước.

Vì vậy, GCM dựa trên chế độ hoạt động CTR và MAC thêm vào để nhận thực cả bản tin và dữ liệu bổ sung mà chỉ yêu cầu nhận thực. Hàm tính toán băm này chỉ sử dụng phép nhân trong trường Galois. Lựa chọn này được đề xuất bởi nó dễ dàng thực hiện phép nhân trong trường Galois và dễ hoạt động trong các phần cứng. Đã có nghiên cứu thực hiện các phương thức mã hóa khối có sẵn và cho thấy cách tiếp cận mã hóa nhận thực dựa trên CTR là phương thức hoạt động hiệu quả nhất cho các mạng gói tốc độ cao. Ngoài ra cũng chứng tỏ rằng GCM đáp ứng một mức độ cao về yêu cầu an ninh.

#### *Tạo số giả ngẫu nhiên sử dụng hàm băm và MAC*

Các yếu tố cần thiết với bất kỳ bộ tạo số giả ngẫu nhiên PRNG (Pseudorandom Number Generator) nào đều là các giá trị hạt giống (seed) và một thuật toán xác định để tạo ra mỗi chuỗi bit giả ngẫu nhiên. Nếu thuật toán được dùng như một hàm giả ngẫu nhiên PRF (Pseudorandom Function) để tạo một giá trị yêu cầu, ví dụ như khóa phiên, khi đó hạt giống chỉ được biết bởi người sử dụng trong PRF. Nếu thuật toán được dùng để tạo ra một chuỗi hàm mã hóa, thì hạt giống đóng vai trò như một khóa bí mật chỉ được biết bởi người gửi và người nhận.

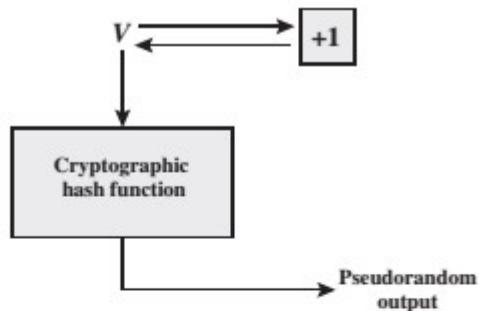
Do thuật toán mã hóa cho đầu ra gần như ngẫu nhiên, nó có thể hoạt động như cơ chế cơ bản của PRNG. Tương tự, một hàm băm hoặc MAC cũng cho đầu ra gần ngẫu

nhiên và có thể được sử dụng để xây dựng PRNG. Chuẩn ISO 18031 (Tạo bit ngẫu nhiên) và NIST SP 800-90 (Đề xuất cho số giả ngẫu nhiên sử dụng bộ tạo bit ngẫu nhiên xác định) đã định nghĩa một phương pháp nhằm tạo số ngẫu nhiên bằng hàm băm mật mã hóa. SP 800-90 cũng định nghĩa bộ tạo số ngẫu nhiên dựa trên HMAC. Hai phương pháp được trình bày dưới đây.

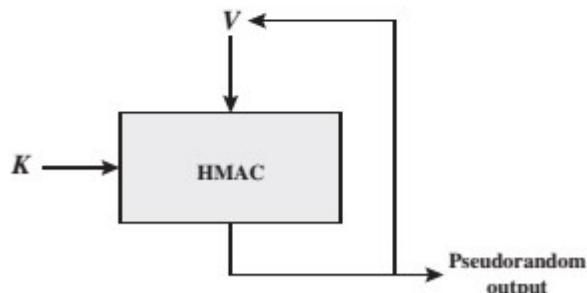
### **PRNG dựa trên hàm băm**

Hình 4.13a biểu diễn chiến lược cơ bản cho PRNG dựa trên hàm băm được xác định trong SP 800-90 và ISO 18031. Thuật toán có các đầu vào như sau

- $V$  = hạt giống
- $Seedlen$  = độ dài bit của  $V$ ,  $V \geq k + 64$  với  $k$  là mức bảo mật mong muốn thể hiện trong các bit
- $n$  = số mong muốn của bit đầu ra



(a) PRNG using cryptographic hash function



(b) PRNG using HMAC

Hình 4.13: Kiến trúc cơ sở của hàm băm dựa trên PRNG

Thuật toán sử dụng hàm băm mật mã hóa  $H$  với giá trị băm đầu ra outlen bit. Hoạt động cơ bản của thuật toán bao gồm

$$m = \lceil n / \text{outlen} \rceil$$

$$\text{data} = v$$

$$W = \text{chuỗi null}$$

```

For  $i = 1$  to  $m$ 
 $W_i = H(data)$ 
 $W = W \parallel w_i$ 
 $data = (data + 1) \bmod 2^{seedlen}$ 
Trả về  $n$  bit bên trái của  $W$ 

```

Do đó, luồng bit giả ngẫu nhiên thu được  $w_1 \parallel w_2 \dots \parallel w_m$  với khối cuối cùng có thể loại bỏ nếu cần thiết.

### **PRNG dựa trên hàm MAC**

Mặc dù với thuật toán băm mật mã hóa mạnh như SHA-2, có thể đạt được độ tin cậy cao hơn bằng cách sử dụng MAC. Bên cạnh đó, HMAC được sử dụng để xây dựng PRNG dựa trên MAC do HMAC được sử dụng rộng rãi như một hàm MAC tiêu chuẩn hóa và có thể áp dụng trong nhiều giao thức và ứng dụng. Như trong SP 800-90, những bất lợi của phương pháp này so với phương pháp dựa trên hàm băm là thời gian thực hiện tăng lên gấp đôi, bởi vì HMAC liên quan đến hai quá trình băm cho mỗi khối đầu ra. Ưu điểm của phương pháp HMAC là chúng cung cấp mức độ tin cậy cao hơn đối với an ninh so với tiếp cận theo hàm băm đơn thuần.

Đối với phương pháp dựa trên MAC, cần hai đầu vào gồm: một khóa  $K$  và một hạt giống  $V$ . Sự kết hợp của  $K$  và  $V$  tạo thành một hạt giống chung cho PRNG. Hình 4.13b cho thấy cấu trúc cơ bản của cơ chế PRNG. Lưu ý rằng các khóa còn lại tương tự nhau cho mỗi khối đầu ra, và các dữ liệu đầu vào cho mỗi khối chính là thẻ đầu ra của khối trước. Các đặc điểm kỹ thuật SP 800-90 cũng cập nhật định kỳ các tham số  $K$  và  $V$  để tăng cường an ninh.

Trong thực tế, với chuẩn bảo mật mạng LAN không dây IEEE 802.11i, dữ liệu đầu vào gồm hạt giống kết hợp với một bộ đếm. Bộ đếm tăng với mỗi khối đầu ra  $w_i$ . Phương pháp này có thể cung cấp mức độ bảo mật nâng cao so với tiếp cận trong SP 800-90. Trong SP 800-90, dữ liệu đầu vào cho khối đầu ra  $w_i$  chính là đầu ra của quá trình ( $w_{i-1}$ ) trước của HMAC. Vì vậy, một kẻ tấn công có thể quan sát đầu ra giả ngẫu nhiên và biết cả đầu vào lần đầu ra của HMAC. Mặc dù vậy, với giả định rằng HMAC an toàn, hiểu biết về đầu vào và đầu ra không đủ để tái tạo lại khóa  $K$  và do đó kẻ tấn công không đủ khả năng để đoán trước bit giả ngẫu nhiên trong tương lai. Vì vậy, các phương pháp đưa ra bởi giao thức băm mật lớp truyền tải nói chung và giao thức bảo mật lớp truyền tải không dây nói riêng liên quan đến việc thực hiện hai lần HMAC cho mỗi khối đầu ra  $w_i$ .

### **Câu hỏi ôn tập chương 4**

- 
1. Ứng dụng của hàm băm và các yếu tố ảnh hưởng tới bảo mật của hàm băm
  2. Các đặc trưng và nguyên tắc hoạt động của mã xác thực bản tin
  3. Độ an toàn của mã xác thực bản tin MAC
  4. Đặc trưng cơ bản của mã xác thực bản tin dựa trên hàm băm HMAC
  5. Các phương pháp mã hóa xác thực

## Chương 5: Xác thực

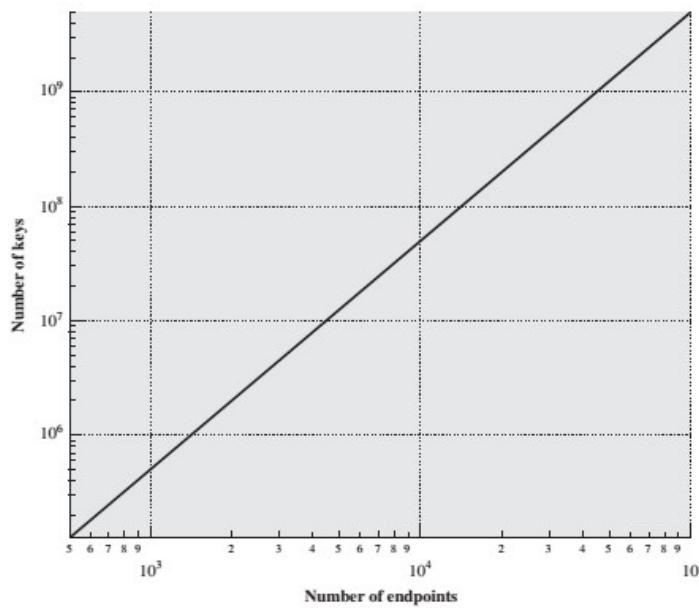
### 5.1 Quản lý và phân phối khóa

#### 5.1.1 Phân phối khóa đối xứng sử dụng mật mã hóa đối xứng

Đối với phương pháp mã khóa đối xứng, các thành viên chia sẻ cùng một khóa và khóa được bảo vệ bởi các thành phần ngoài. Hơn nữa, khóa được thay đổi thường xuyên nhằm tránh các tấn công. Vì vậy, sức mạnh của bất kỳ một hệ thống mã hóa nào cũng đều liên quan tới kỹ thuật phân phối khóa, một thuật ngữ sử dụng để tạo ra khóa giữa hai thành viên đảm bảo tính bí mật với các bên khác.

Với hai bên A và B, sự phân phối khóa có thể thực hiện theo một số cách sau:

1. A lựa chọn một khóa và chuyển phát tới B.
2. Một thành viên thứ 3 lựa chọn khóa và chuyển phát tới A và B.
3. Nếu A và B cùng sử dụng một khóa trước đó, một thành viên có thể chuyển một khóa mới dựa trên việc mã hóa khóa cũ.
4. Nếu A và B có một kết nối mã hóa với một thành viên C, C có thể chuyển phát khóa cho cả A và B.



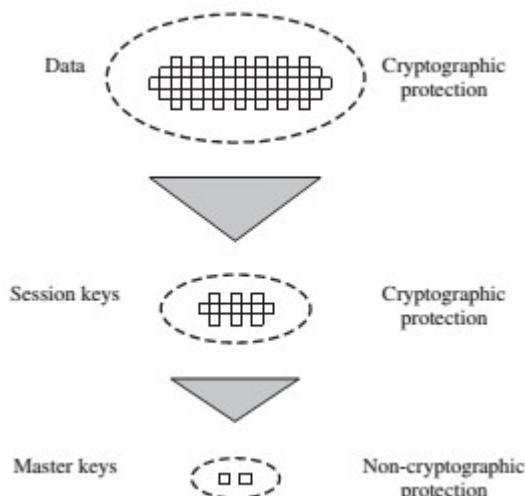
Hình 5.1: Số lượng các khóa yêu cầu cho các kết nối ngẫu nhiên giữa các điểm cuối

Tùy chọn 1 và 2 yêu cầu chuyển phát nhân công. Với các liên kết bảo mật, đây là yêu cầu hợp lý do mỗi thiết bị mã hóa liên kết dữ liệu sẽ trao đổi số liệu trên một phần khác của liên kết. Tuy nhiên, với các mã hóa từ đầu cuối tới đầu cuối qua mạng, chuyển

phát nhân công tạo ra sự phức tạp. Trong một hệ thống phân tán, bất kỳ một đầu cuối nào đều có thể cần đặt chỗ trước sự trao đổi với các đầu cuối khác. Vì vậy cần một số lượng khóa lớn cung cấp động. Vấn đề này đặc biệt khó trong các hệ thống mạng phân tán diện rộng. Độ phức tạp phụ thuộc vào số lượng các cặp kết nối, nếu tại mức IP có  $N$  đầu cuối thì số lượng khóa yêu cầu là  $N(N-1)/2$ , nếu tại mức ứng dụng thì có số lượng gấp nhiều lần host. Hình 5.1 chỉ ra độ phức tạp của nhiệm vụ phân phối khóa. Một mạng có 100 nodes cần  $\frac{1}{2}$  triệu khóa, 10.000 ứng dụng cần tới hơn 50 triệu khóa cho mã hóa lớp ứng dụng.

Tùy chọn 3 là có thể sử dụng cho mã hóa liên kết hoặc mã hóa từ đầu cuối tới đầu cuối, nhưng nếu một kẻ tấn công thành công trong việc tiếp cận với một khóa thì tất cả các khóa tiếp theo sẽ bị lộ. Hơn nữa, sự phân bổ khởi tạo lên đến hàng triệu khóa vẫn phải được thực hiện.

Đối với mã hóa từ đầu cuối tới đầu cuối, một số biến thể dựa trên tùy chọn 4 được sử dụng rộng rãi. Trong lược đồ này, một trung tâm phân phối khóa chịu trách nhiệm phân phối khóa cho các cặp người dùng (máy chủ, các quy trình, các ứng dụng) khi cần thiết. Mỗi người dùng phải chia sẻ một khóa duy nhất với các trung tâm phân phối khóa.



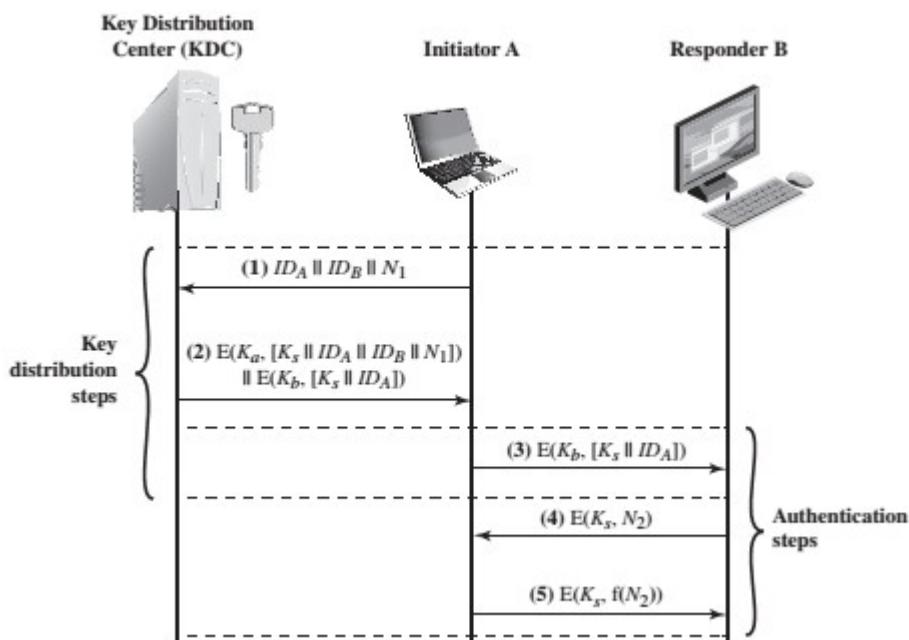
Hình 5.2: Mô hình phân cấp khóa

Ở mức thấp nhất, hai mức khóa được sử dụng như trên hình 5.2. Truyền thông giữa các đầu cuối được mã hóa bằng một khóa tạm thời, thường được gọi là một khóa phiên. Thông thường, một khóa phiên được sử dụng trong khoảng thời gian của một kết nối logic tồn tại, ví dụ như một kết nối Frame Relay hoặc kết nối truyền tải và sau đó bị loại bỏ. Mỗi khóa phiên thu được từ các trung tâm phân phối khóa trên cùng một hạ tầng

truyền thông của mạng. Các khóa phiên được truyền đi trong dạng mã hóa, sử dụng một khóa chủ được chia sẻ bởi các trung tâm phân phối khóa và một hệ thống đầu cuối hoặc người sử dụng. Đối với mỗi hệ thống đầu cuối hoặc người sử dụng, có một khóa chủ duy nhất mà nó chia sẻ với trung tâm phân phối khóa. Số lượng khóa chủ chỉ là  $N$  và có thể phân phối không cần mã hóa.

### Kịch bản phân phối khóa

Việc phân phối khóa có thể được triển khai theo một số kịch bản khác nhau. Một kịch bản điển hình được minh họa trên hình 5.3, với giả thiết mỗi người dùng chia sẻ một khóa chủ duy nhất với trung tâm phân phối khóa KDC (Key Distribution Center).



Hình 5.3: Kịch bản phân phối khóa

Giả định rằng người dùng A muốn thiết lập một kết nối logic với B và yêu cầu khóa phiên một lần để bảo vệ các dữ liệu được truyền qua kết nối. A có một khóa chủ,  $K_a$ , chỉ riêng A và KDC; tương tự, B chia sẻ khóa chủ  $K_b$  với KDC. Các bước được thực hiện như sau:

1. A phát hành yêu cầu đến KDC cho một khóa phiên để bảo vệ một kết nối logic tới B. Bản tin bao gồm nhận dạng của A và B và một định danh duy nhất cho phiên này,  $N_1$ .  $N_1$  được gọi là nonce. Nonce có thể là một tem thời gian, một số đếm, hoặc một số ngẫu nhiên; yêu cầu tối thiểu là phải khác nhau với mỗi yêu cầu. Ngoài ra, để ngăn chặn tấn công giả trang, Nounce cần phải khó đoán nên thường sử dụng số ngẫu nhiên.

2. KDC phản hồi với một bản tin được mã hóa bằng cách sử dụng khóa  $Ka$ . Như vậy, A là người duy nhất đọc tin nhắn thành công, và A biết rằng bản tin xuất phát từ KDC. Bản tin gồm hai nội dung dành cho A:

- + Khóa phiên dùng một lần  $Ks$ , sử dụng cho phiên làm việc
- + Bản tin yêu cầu nguồn gốc bao gồm cả nonce, để cho phép A ghép phản hồi này với yêu cầu thích hợp.

Như vậy, A có thể xác minh rằng yêu cầu ban đầu của nó không bị thay đổi trước khi tiếp nhận bởi KDC, từ nonce cho thấy đây không phải là của bản tin phát lại trước đó.

Ngoài ra, bản tin bao gồm hai nội dung dành cho B:

- + Các khóa phiên một lần  $Ks$ , sử dụng cho phiên làm việc
- + Một nhận dạng của A (ví dụ, địa chỉ mạng của nó), IDA

Hai nội dung cuối được mã hóa với  $Kb$ , chúng được gửi đến B để thiết lập kết nối và chứng minh nhận dạng của A.

3. A lưu trữ khóa phiên cho phiên sắp tới và chuyển tiếp đến B thông tin có nguồn gốc tại KDC cho B, cụ thể là, E( $Kb, [Ks\} IDA$ ). Do thông tin này được mã hóa với  $Kb$ , nên được bảo vệ khỏi tấn công nghe trộm. B khi đó sẽ biết khóa phiên ( $Ks$ ), biết bên kia là A (IDA), và biết rằng các thông tin có nguồn gốc tại KDC (vì nó được mã hóa bằng  $Kb$ ).

Tại thời điểm này, một khóa phiên được chuyển phát an toàn cho A và B cùng với hai bước phụ:

4. Sử dụng khóa phiên mới được tạo ra để mã hóa, B sẽ gửi một nonce  $N_2$  tới A.
5. Ngoài ra, sử dụng khóa phiên  $Ks$ , A trả lời B với  $f(N_2)$  với  $f$  là một hàm thực hiện một số biến đổi trên  $N_2$  (ví dụ: cộng một).

Những bước này đảm bảo với B rằng bản tin nguyên gốc mà nó nhận được (bước 3) không phải là một bản tin phát lại.

Lưu ý rằng việc phân phối khóa thực tế chỉ liên quan đến các bước 1 đến 3, nhưng bước 4 và 5, cũng như bước 3 đều thực hiện chức năng xác thực.

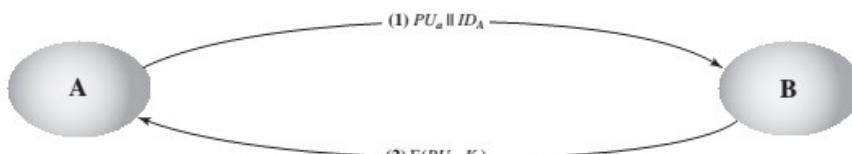
### 5.1.2 Phân phối khóa đối xứng bằng mật mã hóa bất đối xứng

Một trong những ứng dụng quan trọng nhất của một hệ thống mật mã hóa công khai là để mã hóa các khóa bí mật. Các nguyên tắc chung được trình bày dưới đây.

### Phân phối khóa bí mật cơ bản

Một lược đồ đơn giản được đưa ra bởi Merkle như minh họa trong Hình 5.4. Nếu A muốn truyền thông với B, các thủ tục sau đây được sử dụng:

1. Tạo ra một cặp khóa công khai / riêng  $\{PU_a, PR_a\}$  và phát đi một bản tin tới B gồm PUa và định danh của A, IDA.
2. B tạo ra một khóa bí mật  $K_s$  và truyền nó cho A, được mã hóa với khóa công khai của A.
3. A tính D ( $PR_a, E(PU_a, K_s)$ ) để khôi phục lại các khóa bí mật. Bởi vì chỉ có A có thể giải mã bản tin, chỉ có A và B sẽ biết nhận dạng của  $K_s$ .
4. A loại bỏ  $PU_a$  và  $PR_a$  và B loại bỏ  $PU_a$ .



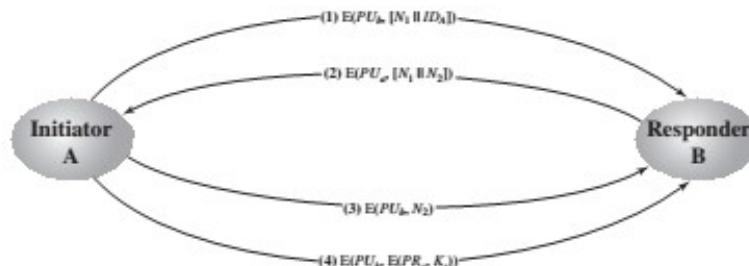
Hình 5.4: thủ tục phân phối khóa bí mật đơn giản

### Phân phối khóa bí mật với nhận thực và bảo mật

Hình 5.5 chỉ ra một phương pháp phân phối khóa bí mật cung cấp nhận thực và bảo mật. Giả định rằng A và B đã trao đổi khóa công khai theo một lược đồ nào đó, các bước sau sẽ xảy ra.

A sử dụng khóa công khai của B để mã hóa một bản tin đến B có chứa một định danh của A ( $ID_A$ ) và một nonce ( $N_1$ ), được sử dụng để xác định giao dịch duy nhất này.

2. B gửi một bản tin đến A được mã hóa với  $PU_a$  và chứa của một nonce ( $N_1$ ) cũng như một nonce mới được tạo ra bởi B ( $N_2$ ). Bởi vì chỉ có B có thể giải mã thông báo (1), sự hiện diện của  $N_1$  trong bản tin (2) đảm bảo A rằng trả lời là từ B.



Hình 5.5: thủ tục phân phối khóa bí mật cung cấp bảo mật và nhận thực

3. A trả lại  $N_2$ , mã hóa bằng khóa công khai của B, để đảm bảo B biết đáp ứng đó là từ A.

4. A chọn một khóa bí mật  $K_s$  và gửi  $M = E(PU_b, E(PR_a, K_s))$  đến B. Bản tin mã hóa này với khóa công khai của B đảm bảo rằng chỉ có B có thể đọc nó; mã hóa với khóa riêng của A đảm bảo rằng chỉ có A có thể gửi bản tin đó.

5. B tính  $D(PU_a, D(PR_b, M))$  để khôi phục lại các khóa bí mật.

Kết quả là lược đồ này đảm bảo cả tính bảo mật và xác thực trong việc trao đổi khóa bí mật.

### 5.1.3 Phân phối khóa công khai

Một số giải pháp kỹ thuật đã được đề xuất cho việc phân phối khóa công khai có thể nhóm lại thành các loại chung sau:

- + Thông báo công khai
- + Thư mục khóa công khai
- + Trung tâm thẩm quyền khóa công khai
- + Chứng thư khóa công khai

#### *Thông báo công khai*

Theo loại này, điểm của mã hóa khóa công khai là nơi khóa công khai được đưa ra. Nếu thuật toán khóa công khai được chấp thuận rộng rãi như RSA thì bất kỳ một thành viên tham gia có thể gửi khóa công khai của mình cho bất kỳ thành viên tham gia khác (hình 5.6).



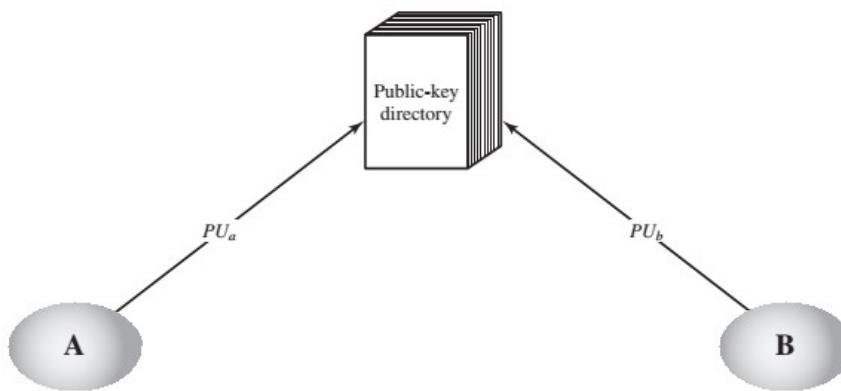
Hình 5.6: Phân phối khóa tự do

Mặc dù phương pháp này là thuận tiện nhưng có một nhược điểm lớn là ai cũng có thể giả mạo thông báo công khai như vậy. Một số người dùng có thể giả vờ là người sử

dung A và gửi một khóa công khai cho thành viên khác. Khi người dùng A phát hiện ra sự giả mạo và cảnh báo người tham gia khác thì kẻ giả mạo đã có thể có thể đọc tất cả các bản tin mã hóa dành cho A và có thể sử dụng các khóa giả mạo để xác thực.

#### *Thư mục khóa công khai*

Một mức độ cao hơn về an ninh có thể đạt được bằng cách duy trì một thư mục động của khóa công khai. Bảo dưỡng và phân phối thư mục khóa công khai được chịu trách nhiệm của một số thực thể hoặc tổ chức đáng tin cậy (Hình 5.7).



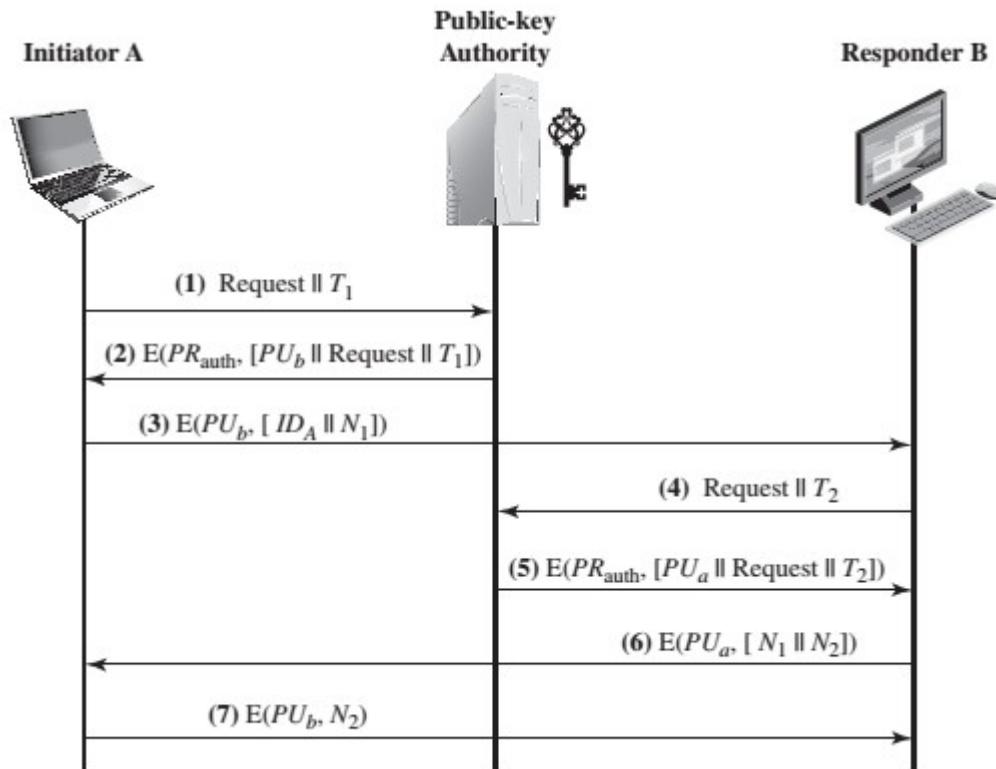
Hình 5.7: Phân phối khóa qua thư mục khóa công khai

Phương pháp này gồm các yếu tố sau:

1. Người có thẩm quyền duy trì một danh mục với mỗi khoản mục {tên, khóa công khai} cho từng thành viên tham gia.
2. Mỗi người tham gia đăng ký một khóa công khai với bên thẩm quyền quản lý danh mục. Việc đăng ký thư mục qua các hình thức truyền thông được chứng thực an toàn.
3. Một người tham gia có thể thay thế khóa bất cứ lúc nào.
4. Những người tham gia cũng có thể truy cập vào trực tiếp vào thư mục thông qua kênh được nhận thực.

#### *Trung tâm thẩm quyền khóa công khai*

Đây là một giải pháp đảm bảo tính an ninh cao hơn giải pháp cung cấp từ thư mục khóa công khai. Một hệ thống điển hình được thể hiện trên hình 5.8.



Hình 5.8: Kịch bản phân phối khóa công khai

Giải pháp này thực hiện trên một số bước sau:

1. A gửi một bản tin chứa tem thời gian tới trung tâm thẩm quyền khóa công khai có chứa một yêu cầu khóa công khai của B.
2. Trung tâm ủy quyền trả lời một bản tin được mã hóa bằng khóa riêng của chính trung tâm,  $PR_{auth}$ . Như vậy, A có thể giải mã các bản tin bằng khóa công khai của trung tâm thẩm quyền.
3. A lưu trữ công khai của B và sử dụng nó để mã hóa một bản tin đến B chứa định danh của A ( $ID_a$ ) và một nonce ( $N_1$ ) để xác định giao dịch duy nhất này.
4. B thu lấy khóa công khai của A từ trung tâm thẩm quyền theo cách thức tương tự như A lấy khóa công khai của B.
5. Tại thời điểm này, khóa công khai đã được chuyển giao một cách an toàn để A và B có thể bắt đầu trao đổi các thông tin an toàn. Tuy nhiên, có thể bổ sung hai bước sau:
6. B gửi một bản tin đến A được mã hóa với  $PU_a$  và chứa của một nonce ( $N_1$ ) cũng như một nonce mới được tạo ra bởi B ( $N_2$ ). Do chỉ có B có thể có bản tin giải mã (3), sự hiện diện của  $N_1$  trong bản tin tại bước (6) đảm bảo rằng A biết trả lời từ B.

7. A trả lại nounce  $N_2$  được mã hóa bằng khóa công khai của B, để B biết rằng bản tin đó đến từ A.

#### *Chứng thư khóa công khai*

Một cách tiếp cận khác, lần đầu tiên được đề xuất bởi Kohnfelder sử dụng chứng thư cho những người tham gia để trao đổi các khóa. Về bản chất, một giấy chứng thư bao gồm một khóa công khai, một nhận dạng của chủ sở hữu chính và toàn bộ khôi chữ ký của một bên thứ ba đáng tin cậy. Thông thường, các bên thứ ba là một cơ quan cấp chứng thư, chẳng hạn như một cơ quan chính phủ hoặc một tổ chức tài chính, được tin tưởng bởi cộng đồng người dùng. Một người sử dụng có thể gửi khoá công khai tới cơ quan cấp chứng chỉ một cách an toàn và có được một giấy chứng nhận. Người dùng sau đó có thể công bố chứng chỉ. Bất cứ ai cũng cần khóa công khai của người dùng này có thể có được giấy chứng nhận và xác minh rằng nó là hợp lệ bằng chữ ký tin cậy kèm theo. Các thành viên khác có thể xác minh rằng chứng chỉ đã được tạo ra bởi nơi có thẩm quyền.

#### **5.1.4 Chứng thư X.509**

ITU-T đưa ra khuyến nghị X.509 như là một phần của tập tiêu chuẩn X.500 nhằm định nghĩa dịch vụ thư mục. Các thư mục duy trì một cơ sở dữ liệu thông tin về người dùng bao gồm một ánh xạ từ tên người sử dụng, địa chỉ mạng, cũng như các thuộc tính khác và thông tin về người sử dụng. X.509 định nghĩa một khung làm việc cho nhiệm vụ cung cấp dịch vụ chứng thực. Các thư mục có thể phục vụ như là một kho lưu trữ của khóa công khai hoặc chứng chỉ. Ngoài ra, X.509 định nghĩa các giao thức xác thực khác dựa trên việc sử dụng các giấy chứng nhận khóa công khai. Một số phiên bản X.509 được trình bày dưới đây.

##### *X.509 version 1*

Được định nghĩa vào năm 1988, X.509 version 1 giờ đây hầu như không còn được sử dụng.

##### *X.509 version 2*

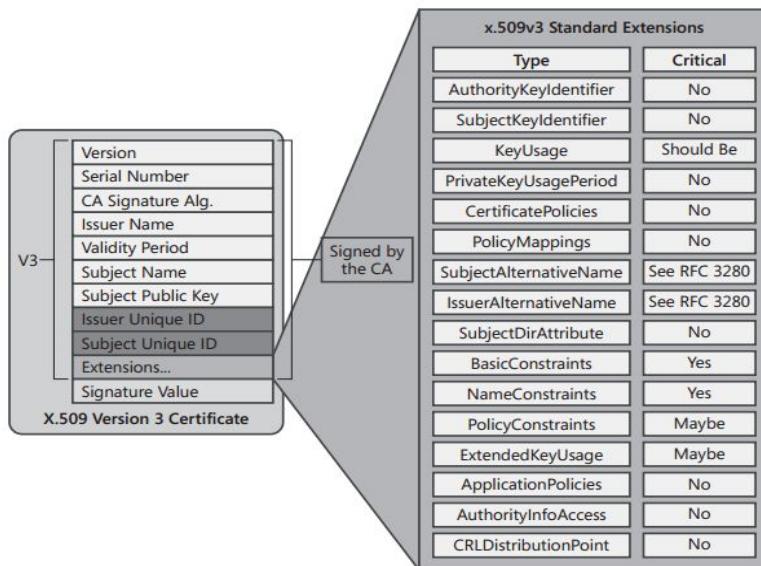
Mặc dù chứng chỉ X.509 version 1 cung cấp khá đầy đủ những thông tin cơ bản về người nắm giữ chứng chỉ nhưng nó lại có ít thông tin về tổ chức cấp phát chứng chỉ khi chỉ bao gồm Issuer Name, CA Signature Algorithm và Signature Value. Điều này không giúp dự phòng trong trường hợp CA được thay mới. Khi chứng chỉ của CA được thay mới, trường Issuer Name trong cả 2 chứng chỉ mới và cũ đều như nhau. Tương tự, có thể có một tổ chức khác muốn tạo một CA có trường Issuer Name trong chứng chỉ giống như

vậy. Giải quyết vấn đề này để có thể sử dụng lại Issuer Name thì chứng chỉ X.509 version 2 đã được giới thiệu vào năm 1993. Trong định dạng mới có thêm 2 trường chức năng:

- + Issuer Unique ID: là một trường không bắt buộc, chứa chuỗi giá trị ở hệ 16, mang tính duy nhất và dành để nhận dạng CA. Khi CA thay mới chứng chỉ của chính nó, một Issuer Unique ID mới được khởi tạo cho chứng chỉ đó.
- + Subject Unique ID: là một trường không bắt buộc, chứa chuỗi giá trị ở hệ 16, mang tính duy nhất và dùng để nhận dạng chủ thẻ của chứng chỉ. Nếu chủ thẻ này cũng chính là CA thì trường này sẽ giống với Issuer Unique ID.

### X.509 version 3

Được ra đời vào năm 1996, định dạng X.509 version 3 được bổ sung thêm các phần mở rộng (extension) để khắc phục các vấn đề liên quan tới việc so khớp Issuer Unique ID và Subject Unique ID cũng như là các vấn đề về xác thực chứng chỉ. Một chứng chỉ X.509 version 3 có thể chứa một hoặc nhiều extension, như được thể hiện trong hình dưới đây:



Hình 5.9: Trường mở rộng của chứng chỉ X.509

Mỗi extension trong chứng chỉ X.509 version 3 gồm 3 phần:

- + Extension Identifier: là một mã nhận dạng đối tượng (Object Identifier – OID) cho biết kiểu định dạng và các định nghĩa của extension.
- + Criticality Flag: là một dấu hiệu cho biết thông tin trong extension có quan trọng (critical) hay không. Nếu một ứng dụng không nhận diện được

trạng thái critical của extension hoặc extension không hề chứa giá trị nào thì chứng chỉ đó không thể được chấp nhận hoặc được sử dụng. Nếu mục criticality flag này không được thiết lập thì một có thể sử dụng chứng chỉ ngay cả khi ứng dụng đó không nhận diện được extension.

- + Extension Value: là giá trị được gán cho extension. Nó phụ thuộc vào từng extension cụ thể.

Trong một chứng chỉ X.509 version 3, các extension sau có thể có là:

- + Authority Key Identifier: extension này có thể chứa một hoặc hai giá trị, chúng có thể là:
- + Subject Name của CA và Serial Number của chứng chỉ của CA mà đã cấp phát chứng chỉ này.
- + Giá trị băm của khóa công khai của chứng chỉ của CA mà đã cấp phát chứng chỉ này.

Subject Key Identifier: extension này chứa giá trị băm của khóa công khai của chứng chỉ.

Key Usage: một CA, người dùng, máy tính, thiết bị mạng hoặc dịch vụ có thể sở hữu nhiều hơn một chứng chỉ. Extension này định nghĩa các dịch vụ bảo mật mà một chứng chỉ có thể cung cấp như:

Digital Signature: khóa công khai có thể được dùng để kiểm tra chữ ký. Khóa này cũng được sử dụng để xác thực máy khách và xác minh nguồn gốc của dữ liệu.

Non-Repudiation: khóa công khai có thể được dùng để xác minh nhận dạng của người ký, ngăn chặn người ký này từ chối rằng họ không hề ký lên bản tin hoặc đối tượng nào đó.

Key Encipherment: khóa công khai có thể được dùng để trao đổi khóa, ví dụ như đối xứng (hoặc khóa phiên). Giá trị này được dùng khi một khóa RSA được dùng cho việc quản lý khóa.

Data Encipherment: khóa công khai có thể được dùng để mã hóa dữ liệu một cách trực tiếp thay vì phải trao đổi một khóa đối xứng (hay khóa phiên) để mã hóa dữ liệu.

Key Agreement: khóa công khai có thể được dùng để trao đổi khóa, ví dụ như khóa đối xứng. Giá trị này được dùng khi một khóa Diffie-Hellman được dùng cho việc quản lý khóa.

Key Cert Sign: khóa công khai có thể được dùng để kiểm tra chữ ký của chứng chỉ số.

CRL Sign: khóa công khai có thể được dùng để kiểm tra chữ ký của CRL (danh sách chứa các chứng chỉ bị thu hồi).

Encipher Only: giá trị này được dùng kết hợp với các extension Key Agreement và Key Usage. Kết quả là khóa đối xứng chỉ có thể được dùng để mã hóa dữ liệu.

Decipher Only: giá trị này được dùng kết hợp với các extension Key Agreement và Key Usage. Kết quả là khóa đối xứng chỉ có thể được dùng để mã hóa dữ liệu.

Private Key Usage Period: extension này cho phép khóa bí mật có khoảng thời gian hiệu lực khác so với khoảng thời gian hiệu lực của chứng chỉ. Giá trị này có thể được đặt ngắn hơn so với khoảng thời gian hiệu lực của chứng chỉ. Điều này giúp khóa bí mật có thể được dùng để ký lên các tài liệu trong một khoảng thời gian ngắn (ví dụ, một năm) trong khi khóa công khai có thể được dùng để xác minh chữ ký trong khoảng thời gian hiệu lực của chứng chỉ là 5 năm.

Certificate Policies: extension này mô tả các chính sách và thủ tục được dùng để xác minh chủ thẻ của chứng chỉ trước khi chứng chỉ được cấp phát. Các chính sách chứng chỉ được đại diện bởi các OID. Ngoài ra, một chính sách chứng chỉ có thể bao gồm một đường dẫn (URL) tới trang web mô tả nội dung của chính sách và thủ tục.

Policy Mappings: extension này cho phép chuyển đổi thông tin về chính sách giữa hai tổ chức. Ví dụ, thử tưởng tượng rằng một tổ chức định nghĩa một chính sách chứng chỉ có tên là Management Signing mà trong đó các chứng chỉ được dùng để ký lên một lượng lớn các đơn đặt hàng. Một tổ chức khác có thể có một chính sách chứng chỉ tên là Large Orders mà cũng được dùng để ký lên một lượng lớn các đơn đặt hàng. Khi đó, Policy Mapping cho phép hai chính sách chứng chỉ này được đánh giá ngang nhau.

Subject Alternative Name: extension này cung cấp một danh sách các tên thay thế cho chủ thẻ của chứng chỉ. Trong khi định dạng cho Subject Name thường tuân theo chuẩn X.500 thì Subject Alternative Name cho phép thể hiện theo các dạng khác như User Principal Name (UPN), địa chỉ email, địa chỉ IP hoặc tên miền (DNS).

Issuer Alternative Name: extension này cung cấp một danh sách các tên thay thế cho CA. Mặc dù thường không được áp dụng nhưng extension này có thể chứa địa chỉ email của CA.

Subject Dir Attribute: extension này có thể bao gồm bất kỳ thuộc tính nào từ danh mục LDAP hoặc X.500 của tổ chức, ví dụ, thuộc tính country. Extension này có thể chứa nhiều thuộc tính và với mỗi thuộc tính phải gồm OID và giá trị tương ứng của nó.

Basic Constraints: extension này cho biết chứng chỉ có phải của CA hay của các chủ thẻ như người dùng, máy tính, thiết bị, dịch vụ. Ngoài ra, extension này còn bao gồm một ràng buộc về độ dài của đường dẫn mà giới hạn số lượng các CA thứ cấp (subordinate CA) có thể tồn tại bên dưới CA mà cấp phát chứng chỉ này.

Name Constraints: extension này cho phép một tổ chức chỉ định không gian tên (namespace) nào được phép hoặc không được phép sử dụng trong chứng chỉ.

Policy Constraints: extension này có thể có trong các chứng chỉ của CA. Nó có thể ngăn cấm Policy Mapping giữa các CA hoặc yêu cầu mỗi chứng chỉ trong chuỗi chứng chỉ phải bao gồm một OID của chính sách chứng chỉ.

Enhanced Key Usage: extension này cho biết khóa công khai của chứng chỉ có thể được sử dụng như thế nào. Những cái này không có trong extension Key Usage. Ví dụ, Client Authentication (có OID là 1.3.6.1.5.5.7.3.2), Server Authentication (có OID là 1.3.6.1.5.5.7.3.1), và Secure E-mail (có OID là 1.3.6.1.5.5.7.3.4). Khi ứng dụng nhận được một chứng chỉ, nó có thể yêu cầu sự có mặt của một OID trong các OID kể trên.

CRL Distribution Points: extension này chứa một hoặc nhiều URL dẫn tới tập tin chứa danh sách các chứng chỉ đã bị thu hồi (CRL) được phát hành bởi CA. Nếu việc kiểm tra trạng thái thu hồi của chứng chỉ được cho phép thì một ứng dụng sẽ sử dụng các URL này để tải về phiên bản cập nhật của CRL. Các URL có thể sử dụng một trong các giao thức như HTTP, LDAP, FTP, File.

Authority Information Access: extension này có thể chứa một hoặc nhiều URL dẫn tới chứng chỉ của CA. Một ứng dụng sử dụng URL này để tải về chứng chỉ của CA khi xây dựng chuỗi chứng chỉ nếu như nó không có sẵn trong bộ nhớ đệm của ứng dụng.

Freshest CRL: extension này chứa một hoặc nhiều URL dẫn tới delta CRL do CA phát hành. Delta CRL chỉ chứa các chứng chỉ bị thu hồi kể từ lần cuối base CRL được phát hành. Nếu việc kiểm tra trạng thái thu hồi của chứng chỉ được cho phép thì một ứng dụng sẽ sử dụng các URL này để tải về phiên bản cập nhật của delta CRL. Các URL có thể sử dụng một trong các giao thức như HTTP, LDAP, FTP, File.

Subject Information Access: extension này chứa thông tin cho biết cách thức để truy cập tới các chi tiết khác về chủ thẻ của chứng chỉ. Nếu đây là chứng chỉ của CA thì

thông tin này có thể bao gồm các chi tiết về các dịch vụ xác minh chứng chỉ hay chính sách của CA. Nếu chứng chỉ được cấp cho người dùng, máy tính, thiết bị mạng, hoặc dịch vụ thì extension này có thể chứa thông tin về các dịch vụ được các chủ thể này cung cấp và cách thức để truy cập tới các dịch vụ đó.

## 5.2 Xác thực người sử dụng

### 5.2.1 Nguyên lí xác thực người sử dụng từ xa

Trong hầu hết các hoàn cảnh an ninh máy tính, nhận thực người dùng là khối xây dựng cơ bản và là hàng phòng thủ cơ sở. Nhận thực người dùng là cơ sở của hầu hết các loại điều khiển truy nhập và cho trách nhiệm người dùng. RFC 4949 (bảng thuật ngữ an ninh internet) định nghĩa nhận thực người dùng như trình bày trong các trang tiếp theo đây.

Ví dụ, người dùng Alice có thể có định danh người dùng ABTOKLAS. Thông tin này cần được lưu trữ trên bất kỳ hệ thống máy chủ hoặc máy tính nào mà Alice muốn dùng và có thể được nhận biết bởi những người quản trị hệ thống và các người dùng khác. Một mục của thông tin nhận thực liên kết với định danh người dùng này là một mật khẩu được giữ bí mật (chỉ được biết bởi Alice và hệ thống). Nếu không ai có thể nhận được hoặc đoán được mật khẩu của Alice, khi đó cắp nhận dạng người dùng-mật khẩu của Alice cho phép người quản trị thiết lập quyền truy nhập của Alice và kiểm tra hoạt động của cô ấy. Do định danh của Alice không bí mật, các người dùng hệ thống có thể gửi cho cô ấy email, nhưng do mật khẩu của cô ấy là bí mật, không ai có thể giả làm Alice.

Về cơ bản, nhận dạng là phương tiện qua đó một người dùng cung cấp một định danh xác nhận tới hệ thống; nhận thực người dùng là phương tiện thiết lập tính hiệu lực của xác nhận. Lưu ý rằng nhận thực người dùng khác với nhận thực bản tin. Như đã định nghĩa, nhận thực bản tin là một thủ tục cho phép các bên truyền thông xác minh nội dung của bản tin không bị thay đổi và nguồn là xác thực.

Có bốn phương tiện tổng quát đối với danh tính người dùng có thể được sử dụng độc lập hay kết hợp.

- + Một vài điều chỉ người dùng biết: Ví dụ như mật khẩu, một số nhận dạng cá nhân (Personal Identification Number - PIN) hoặc các câu trả lời đố với một tập câu hỏi được sắp xếp trước.

- + Một vài điều cá nhân sở hữu: Ví dụ như các khóa mật mã, các thẻ khóa điện, các thẻ thông minh và các thẻ vật lý. Loại phương tiện nhận thực này được xem như một dấu hiệu.
- + Một vài điều thuộc về bản chất người dùng (các tham số sinh trắc học tĩnh): Ví dụ bao gồm sự xác nhận bằng vân tay, võng mạc và khuôn mặt.
- + Một vài điều người dùng làm (các tham số sinh trắc học động): Các ví dụ bao gồm sự xác nhận bằng mẫu giọng nói, các đặc tính chữ viết tay và nhịp gõ.

Tất cả các phương pháp này khi được thực hiện và sử dụng đúng cách có thể cung cấp nhận thực người dùng an ninh. Tuy nhiên, mỗi phương pháp đều có những vấn đề. Một kẻ địch có thể đoán được hoặc đánh cắp mật khẩu. Tương tự, một kẻ địch có thể có khả năng đánh lừa hoặc đánh cắp một dấu hiệu. Một người dùng có thể quên mật khẩu hoặc đánh mất dấu hiệu. Hơn nữa, có một lượng truyền thông quản lý đáng kể cho việc quản lý mật khẩu và thông tin dấu hiệu trên hệ thống và bảo mật những thông tin này trên hệ thống. So với nhận thực bằng sinh trắc, có rất nhiều vấn đề, bao gồm việc đối phó với các lỗi, chấp nhận người dùng, giá thành, và sự tiện dụng. Đối với nhận thực người dùng dựa trên mạng, các phương pháp quan trọng nhất bao gồm các khóa mật mã và một vài điều mà cá nhân người dùng biết, như là một mật khẩu.

#### *Nhận thực lẫn nhau*

Một lĩnh vực ứng dụng quan trọng là các giao thức nhận thực lẫn nhau. Các giao thức này cho phép các bên tham gia truyền thông thuyết phục lẫn nhau về danh tính của mỗi bên và để trao đổi các khóa phiên. Trọng tâm của bài toán trao đổi khóa nhận thực là hai vấn đề: tính bảo mật và vấn đề vô hạn. Để ngăn chặn sự mạo danh và ngăn chặn sự xâm hại khóa phiên, thông tin nhận dạng và khóa phiên cần thiết phải được trao đổi ở dạng mã hóa. Điều này yêu cầu sự có mặt trước đó của các khóa bí mật hoặc công khai mà có thể được sử dụng cho mục đích này. Thứ hai, vấn đề vô hạn là quan trọng vì nguy cơ phát lại bản tin. Sự phát lại như vậy trong trường hợp tồi tệ nhất có thể cho phép một kẻ địch xâm hại một khóa phiên hoặc mạo danh một bên nào đó thành công. Trong trường hợp ít nguy hại nhất, một tấn công phát lại thành công có thể làm phá vỡ sự vận hành bằng cách đưa cho các bên các bản tin mạo danh bản tin thật. Các ví dụ sau đây liệt kê các kiểu tấn công phát lại:

- (1) Tấn công phát lại đơn giản nhất là trường hợp trong đó kẻ địch chỉ đơn giản sao chép một bản tin và phát lại nó sau đó.

- (2) Một kẻ địch có thể phát lại bản tin được gắn nhãn thời gian trong cửa sổ thời gian có hiệu lực. Nếu cả bản gốc và bản phát lại đến trong cùng cửa sổ thời gian, biến cố này có thể được ghi lại (logged).
- (3) Như với ví dụ (2), một kẻ địch có thể phát lại một bản tin được gắn nhãn thời gian trong cửa sổ thời gian có hiệu lực, nhưng thêm vào đó, kẻ tấn công loại bỏ bản tin gốc. Do vậy, sự lặp lại không thể bị phát hiện.
- (4) Một tấn công khác liên quan tới phát lại chiêu nghịch không sửa đổi. Đây là một phát lại tới người gửi. Tấn công này là có thể xảy ra nếu mã hóa đối xứng được sử dụng và người gửi không thể dễ dàng ghi nhận sự khác biệt giữa các bản tin được gửi và các bản tin được nhận trên cơ sở nội dung.

Một phương pháp tiếp cận để đối phó với các tấn công phát lại là thêm vào một số thứ tự đối với mỗi bản tin được sử dụng trong một trao đổi nhận thực. Một bản tin mới được chấp nhận chỉ khi số trình tự của nó là đúng tuần tự. Khó khăn đối với tiếp cận này là nó yêu cầu mỗi bên duy trì số trình tự gần nhất đối với mỗi bên tham gia xác nhận mà nó vừa thỏa thuận. Do đó, các số trình tự thường không được sử dụng cho việc trao đổi nhận thực và khóa. Thay vào đó, một trong các tiếp cận phổ biến sau đây được sử dụng:

- + Các nhãn thời gian: Bên A chấp nhận bản tin là tươi mới chỉ khi bản tin chứa một nhãn thời gian mà với sự đánh giá của A, nó là đủ gần với sự hiểu biết của A về thời gian đúng. Tiếp cận này yêu cầu các đồng hồ giữa các bên tham gia khác nhau được đồng bộ.
- + Thách đố/đáp ứng: Bên A, chấp nhận một bản tin tươi mới từ B, đầu tiên gửi B một “thách đố” và yêu cầu bản tin tiếp theo (phản hồi) nhận được từ B chứa giá trị phản hồi đúng tương ứng với thách đố.

Có thể lập luận rằng tiếp cận nhãn thời gian không nên được sử dụng đối với các ứng dụng hướng kết nối vì các khó khăn có hưu với kỹ thuật này. Đầu tiên, một số loại giao thức cần duy trì đồng bộ giữa các đồng hồ của các bộ vi xử lý khác nhau. Giao thức này cũng phải chịu được lỗi, để đối phó với các lỗi trong mạng và bảo mật, để đối phó với các tấn công thù địch. Thứ hai, cơ hội cho một tấn công thành công sẽ nảy sinh nếu có sự mất đồng bộ tạm thời do một lỗi trong cơ cấu đồng hồ của một trong các bên tham gia. Cuối cùng, do bản tính thay đổi và không dự đoán được của trễ trong mạng, các đồng hồ được phân bố không thể duy trì sự đồng bộ chính xác. Do đó, bất kỳ thủ tục dựa trên

nhǎn thời gian nào phải cho phép một cửa sổ thời gian đủ lớn để chịu được trễ trong mạng và đủ nhỏ để tối thiểu hóa cơ hội tấn công.

Mặt khác, tiếp cận thách đố/phản hồi là không phù hợp với ứng dụng phi kết nối, do nó yêu cầu truyền thông bắt tay trước khi có bất kỳ truyền dẫn kết nối nào, đi ngược lại phần lớn đặc tính chính của truyền thông phi kết nối. Với các ứng dụng như thế, sự tin tưởng đối với một vài máy chủ bảo mật thời gian và một nỗ lực kiên định của mỗi bên để giữ các đồng hồ của mình đồng bộ có lẽ là tiếp cận tốt nhất.

#### *Nhận thực một chiều*

Mã hóa đối với nhận thực một chiều được phát triển phổ biến là thư điện tử. Bản tính rất tự nhiên của thư điện tử, và các lợi ích chính của nó là không cần người gửi và người nhận trực tuyến ở tại cùng một thời điểm. Thay vào đó, bản tin thư điện tử được chuyển tiếp tới hộp thư điện tử của người nhận, nơi nó được lưu đệm cho tới khi người nhận có khả năng đọc được nó.

“Bì thư” hoặc tiêu đề của bản tin email cần phải rõ ràng để bản tin có thể được xử lý bởi giao thức lưu và chuyển tiếp email, như giao thức truyền tải thư đơn giản (Simple Mail Transfer Protocol - SMTP) hoặc X.400. Tuy nhiên, thông thường ta muốn giao thức xử lý không yêu cầu tiếp cận dạng bản rõ của bản tin, bởi vì điều đó yêu cầu việc tin cậy cơ chế xử lý thư. Theo đó, bản tin email cần được mã hóa và hệ thống xử lý mail không sở hữu khóa giải mã. Yêu cầu thứ hai là sự nhận thực. Thông thường, người nhận muốn một vài đảm bảo rằng bản tin đến từ đúng người gửi.

#### **5.2.2 Xác thực người dùng sử dụng mật mã khóa đối xứng**

##### **Xác thực lẫn nhau**

Như đã được thảo luận ở phần trên, hai mức độ của các khóa mã hóa đối xứng có thể được sử dụng để cung cấp bảo mật cho truyền thông trong môi trường phân phối. Nhìn chung, chiến lược này liên quan đến việc sử dụng trung tâm phân phối khóa đáng tin cậy (KDC). Mỗi sự kiện trong mạng chia sẻ một khóa bí mật, được biết đến như một khóa chính, với KDC. KDC có trách nhiệm tạo các khóa được sử dụng với thời gian ngắn qua kết nối giữa hai bên, được biết như các khóa phiên, và phân phối những khóa sử dụng các khóa chính để bảo vệ sự phân tán.

Needham và Schroeder đưa ra một giao thức phân phối khóa bí mật sử dụng KDC gồm các đặc điểm xác thực. Giao thức có thể được tóm tắt như sau.

- 
1. A -> KDC:  $ID_A \parallel ID_B \parallel N_1$
  2. KDC -> A:  $E(K_a, [K_s \parallel ID_B \parallel N_1 \parallel E(K_b, [K_s \parallel ID_A])])$
  3. A -> B:  $E(K_b, [K_s \parallel ID_A])$
  4. B -> A:  $E(K_s, N_2)$
  5. A -> B:  $E(K_s, f(N_2))$

Các khóa bí mật  $K_a$  và  $K_b$  được chia sẻ tương ứng giữa A với KDC và B với KDC. Mục đích của giao thức là để phân phối một cách an toàn khóa phiên  $K_s$  tới A và B. Để an toàn có được một khóa phiên mới trong bước 2. Bản tin trong bước 3 có thể được giải mã, và chỉ B mới có thể hiểu được. Bước 4 B gửi lại  $K_s$  cho A, và bước 5 A đảm bảo khóa  $K_s$  cho B, và đảm bảo với B rằng đây là bản tin mới vì sử dụng  $N_2$ . Mục đích của bước 4 và 5 là để ngăn chặn một số loại tấn công phát lại. Cụ thể, nếu một kẻ tấn công có thể nắm bắt được bản tin ở bước 3 và phát lại nó, điều này sẽ làm gián đoạn hoạt động tại B.

Mặc dù việc bắt tay ở bước 4 và bước 5, giao thức vẫn có thể bị tấn công bởi tấn công phát lại. Giả sử một kẻ tấn công X đã có thể thỏa hiệp một khóa phiên cũ. Thì nhận rằng, đây là một sự cố khó khăn hơn khi kẻ tấn công đơn giản quan sát và ghi lại bước 3. Tuy nhiên, đó là một nguy cơ bảo mật tiềm tàng. X có thể mạo danh A và lừa B sử dụng khóa cũ bằng cách phát lại bước 3. Nếu B không nhớ rằng tất cả khóa phiên vô hạn trước đó được sử dụng với A, B sẽ không thể xác định rằng đây là phát lại. Nếu X có thể đánh chặn tại bản tin bắt tay ở bước 4, sau đó nó có thể mạo danh để gửi phản hồi cho B tại bước 5. Từ thời điểm này, X có thể gửi tin nhắn giả đến B để B đưa tới A một khóa phiên được xác thực.

Denning đề xuất khắc phục điểm yếu này bằng cách sửa đổi các giao thức Needham/Schroeder trong đó bao gồm việc bổ sung các tem thời gian cho các bước 2 và 3. Với giả thiết rằng khóa chính,  $K_a$  và  $K_b$ , được an toàn, và nó bao gồm các bước sau đây:

1. A -> KDC:  $ID_A \parallel ID_B$
2. KDC -> A:  $E(K_a, [K_s \parallel ID_B \parallel T \parallel E(K_b, [K_s \parallel ID_A \parallel T])])$
3. A -> B:  $E(K_b, [K_s \parallel ID_A \parallel T])$
4. B -> A:  $E(K_s, N_1)$
5. A -> B:  $E(K_s, f(N_1))$

T là tem thời gian đảm bảo rằng A và B sử dụng khóa phiên mới được tạo ra. Vì thế, cả A và B biết rằng phân phối khóa là một trao đổi mới. A và B có thể xác nhận kịp thời bằng cách kiểm tra như sau:

$$|Clock - T| < \Delta t_1 + \Delta t_2$$

Trong đó  $\Delta t_1$  là được ước tính là thời gian chêch lệch giữa đồng hồ của KDC và đồng hồ cục bộ (A hoặc B) và  $\Delta t_2$  là thời gian trễ dự kiến của mạng. Mỗi nút có thể thiết lập đồng hồ của mình dựa vào một số nguồn tham khảo tiêu chuẩn. Bởi vì tem thời gian T được mã hóa bằng cách sử dụng các khóa chính an toàn, ngay cả khi kẻ tấn công biết được khóa phiên cũ, vẫn không thể thành công bởi vì phát lại ở bước 3 sẽ được phát hiện bởi B. Các bước 4 và 5 để xác nhận nhận được khóa phiên tại B.

Giao thức Denning tăng mức độ bảo mật cho giao thức Needham/Schroeder nhưng một mối quan tâm mới được đặt ra: cụ thể là chương trình mới này đòi hỏi phải thuộc vào đồng hồ được đồng bộ trên toàn mạng và tạo ra nguy cơ khác. Nguy cơ này được dựa trên sự thật rằng các đồng hồ được phân phối có thể không được đồng bộ với việc đồng hồ bị phá hoại hoặc lỗi trong các cơ chế đồng bộ. Vấn đề xảy ra khi đồng hồ của người gửi nhanh hơn đồng hồ của người nhận. Trong trường hợp này, có cơ hội có thể bị đánh chặn một bản tin từ phía người gửi và phát lại nó sau khi tem thời gian trong bản tin trở thành hiện tại tại vị trí của người nhận.

Một cách để đối phó với các cuộc tấn công phát lại-bỏ gói tin là thực thi những yêu cầu thường xuyên kiểm tra đồng hồ của họ dựa vào đồng hồ của KDC. Ngoài ra, việc cần đồng bộ hóa đồng hồ có thể dựa trên giao thức bắt tay sử dụng nounce. Các giao thức Needham/Schroeder dựa trên nounce vẫn có một vài lỗ hổng và được cải thiện như sau:

1. A  $\rightarrow$  B:  $ID_A \parallel N_a$
2. B  $\rightarrow$  KDC:  $ID_B \parallel N_b \parallel E(K_b, [ID_A \parallel N_a \parallel T])$
3. KDC  $\rightarrow$  A:  $E(K_a, [ID_B \parallel N_a \parallel K_s \parallel T_b]) \parallel E(K_b, [ID_A \parallel K_s \parallel T_b]) \parallel N_b$
4. A  $\rightarrow$  B:  $E(K_b, [ID_A \parallel K_s \parallel T_b]) \parallel E(K_s, N_b)$

Với từng bước trao đổi như sau:

1. Khởi đầu bằng việc trao đổi xác thực bằng cách tạo ra một nounce  $N_a$  và gửi cộng thêm nhận dạng của A đến B trong bản rõ. Nonce này sẽ được trả lại cho A trong một tin nhắn mã hóa bao gồm các khóa phiên, đảm bảo tính kịp thời của nó.
2. B cảnh báo KDC rằng cần một khóa phiên. Và gửi bản tin của nó đến KDC bao gồm nhận dạng của B và một nonce  $N_b$ . Nonce này sẽ được trả lại cho B trong một

bản tin được mã hóa bao gồm khóa phiên, đảm bảo tính kịp thời của B. Bản tin của B đến KDC cũng bao gồm một khối được mã hóa và khóa bí mật được chia sẻ bởi B tới KDC. Khối này được sử dụng để hướng dẫn các KDC phát hành chứng chỉ tới A; khối xác định người nhận các thông tin, đưa ra một khoảng thời gian cho các thông tin và các nonce nhận được từ A.

3. KDC đưa cho A nonce của B và một khối đã mã hóa với khóa bí mật của B đã chia sẻ với KDC. Khối phục vụ như một “tấm thẻ” có thể được sử dụng bởi A để xác thực bước tiếp theo. KDC cũng gửi đến A một khối mã hóa với khóa bí mật được chia sẻ bởi A và KDC. Khối này xác nhận rằng B đã nhận được bản tin ban đầu của A ( $ID_B$ ) và điều này là một bản tin kịp thời và không phải là phát lại ( $N_a$ ) và nó cung cấp A với một khóa phiên ( $K_s$ ) và giới hạn thời gian sử dụng của nó ( $T_b$ ).
4. Truyền tải thẻ đến B, cùng với nonce của B, sau này được mã hóa với khóa phiên. Các thẻ cung cấp cho B với một khóa bí mật được sử dụng để giải mã  $E(K_s, N_b)$  để phục hồi các nonce. Thực tế là nonce B được mã hóa với khóa phiên xác nhận rằng bản tin đến từ A và không phải là bản tin phát lại.

Giao thức này cung cấp một cách hiệu quả, phương tiện an toàn cho A và B để thiết lập một phiên làm việc với một khóa phiên an toàn. Hơn nữa, các giao thức đưa cho A sở hữu một khóa có thể được sử dụng để xác thực sau đó tới B, tránh sự liên lạc với máy chủ xác thực nhiều lần. Giả sử rằng A và B thiết lập một phiên bằng cách sử dụng giao thức nói trên và sau đó kết thúc phiên làm việc đó. Sau đó, nhưng trong giới hạn thời gian được thiết lập bởi giao thức, A mong muốn có thêm một phiên làm việc mới với B. Giao thức được mô tả như sau:

1. A -> B:  $E(K_b, [ID_A \parallel K_s \parallel T_b]) \parallel N'_a$
2. B -> A:  $N'_b \parallel E(K_s, N'_a)$
3. A -> B:  $E(K_s, N'_b)$

Khi B nhận được bản tin ở bước 1, nó xác minh rằng thẻ chưa hết hạn. Các nonce mới được tạo ra và  $N'_a$  và  $N'_b$  đảm bảo mỗi bên không có tấn công phát lại.

Trong tất cả những điều trên, thời gian quy định tại  $T_b$  là thời gian tương đối so với đồng hồ B. Như vậy, tem thời gian này không yêu cầu đồng hồ đồng bộ, vì tem thời gian B kiểm tra do chính nó tạo ra.

### Xác thực một chiêu

Trong kịch bản mã hóa đối xứng thì kịch bản phân bổ khóa phân tán là không thực tế. Cơ chế này yêu cầu phía người gửi đặt ra một yêu cầu đến người nhận đã xác định từ trước, đợi một đáp ứng chứa một khóa phiên, và sau đó chỉ gửi một bản tin.

Với một vài phương pháp cải tiến, sử dụng KDC là một ứng cử viên cho mã hóa thư điện tử. Bởi vì chúng ta mong muốn tránh được yêu cầu người nhận (B) phải trực tuyến trên mạng tại cùng thời điểm với người gửi (A), các bước 4 và 5 phải được loại bỏ. Đối với một bản tin có nội dung là M, thứ tự thực hiện như sau:

1.  $A \rightarrow KDC : ID_A \square ID_B \square N_1$
2.  $KDC \rightarrow A : E(K_a, [K_s \square ID_B \square N_1 \square E(K_b, [K_s \square ID_A])])$
3.  $A \rightarrow B : E(K_b, [K_s \square ID_A]) \square E(K_s, M)$

Hướng tiếp cận này đảm bảo rằng chỉ người nhận được xác định trước của bản tin mới có thể đọc nó. Nó cũng cung cấp một mức độ xác thực mà người gửi là A. Như đã chỉ rõ ở trên giao thức này không chống lại được kiểu tấn công phát lại. Một vài giới hạn phòng thủ có thể được cung cấp bằng cách gán thêm một tem thời gian (timestamp) với bản tin. Tuy nhiên, vì các trễ tiềm ẩn trong tiến trình gửi thư điện tử, nên các tem thời gian bị giới hạn về tính hữu dụng.

### Câu hỏi ôn tập chương 5

1. Tóm lược đặc điểm cơ bản của các mô hình phân phối khóa
2. Các đặc trưng cơ bản của chứng chỉ số X509
3. Nguyên tắc xác thực người dùng từ xa
4. Đặc trưng xác thực người dùng sử dụng mật mã khóa đối xứng

## TÀI LIỆU THAM KHẢO

[1] William Stallings, “Cryptography and Network Security Principles and Practice”, Sixth Edition, Pearson, 2014.