

# MỤC LỤC

<b>CHƯƠNG 1. GIỚI THIỆU VỀ AN TOÀN VÀ BẢO MẬT THÔNG TIN.....</b>	<b>8</b>
1.1 Giới thiệu.....	8
1.2 Bảo vệ thông tin trong quá trình truyền thông tin trên mạng.....	8
1.2.1 Các loại hình tấn công .....	8
1.2.2 Yêu cầu của một hệ truyền thông tin an toàn và bảo mật.....	10
1.2.3 Vai trò của mật mã trong việc bảo mật thông tin trên mạng .....	11
1.2.4 Các giao thức (protocol) thực hiện bảo mật.....	11
1.3 Bảo vệ hệ thống khỏi sự xâm nhập phá hoại từ bên ngoài.....	11
1.4 Câu hỏi ôn tập .....	13
<b>CHƯƠNG 2. MÃ HÓA ĐỐI XỨNG CĂN BẢN .....</b>	<b>14</b>
2.1 Mã hóa Ceasar.....	14
2.2 Mô hình mã hóa đối xứng (Symmetric Ciphers) .....	15
2.3 Mã hóa thay thế đơn bảng (Monoalphabetic Substitution Cipher) .....	17
2.4 Mã hóa thay thế đa ký tự.....	19
2.4.1 Mã Playfair .....	19
2.4.2 Mã Hill.....	20
2.5 Mã hóa thay thế đa bảng (Polyalphabetic Substitution Cipher).....	21
2.6 One-Time Pad .....	23
2.7 Mã hoán vị (Permutation Cipher) .....	24
2.8 Tổng kết .....	25
2.9 Câu hỏi ôn tập .....	27
2.10 Bài Tập.....	27
2.11 Bài Tập Thực Hành.....	28
<b>CHƯƠNG 3. MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI.....</b>	<b>30</b>
3.1 Mã dòng (Stream Cipher).....	31
3.1.1 A5/1 .....	32
3.1.2 RC4.....	34
3.2 Mã khối (Block Cipher) .....	37
3.2.1 Mã khối an toàn lý tưởng.....	37
3.2.2 Mạng SPN.....	38
3.2.3 Mô hình mã Feistel .....	38
3.3 Mã TinyDES .....	40
3.3.1 Các vòng của TinyDES.....	40

3.3.2	Thuật toán sinh khóa con của TinyDES.....	42
3.3.3	Ví dụ về TinyDES.....	42
3.3.4	Khả năng chống phá mã known-plaintext của TinyDES.....	42
3.4	Mã DES (Data Encryption Standard).....	43
3.4.1	Hoán vị khởi tạo và hoán vị kết thúc: .....	44
3.4.2	Các vòng của DES .....	45
3.4.3	Thuật toán sinh khóa con của DES .....	46
3.4.4	Hiệu ứng lan truyền (Avalanche Effect) .....	47
3.4.5	Độ an toàn của DES .....	48
3.5	Một số phương pháp mã khối khác .....	49
3.5.1	Triple DES .....	49
3.5.2	Advanced Encryption Standard (AES) .....	49
3.6	Các mô hình ứng dụng mã khối .....	50
3.6.1	Electronic Codebook – ECB .....	50
3.6.2	Cipher Block Chaining – CBC.....	51
3.6.3	Counter – CTR .....	53
3.6.4	Output Feedback – OFB .....	53
3.6.5	Cipher Feedback – CFB .....	54
3.7	Tính chứng thực (authentication) của mã hóa đối xứng.....	55
3.8	Tính không thoái thác (non-repudiation) của mã hóa đối xứng .....	56
3.9	Trao đổi khóa bí mật bằng trung tâm phân phối khóa.....	56
3.10	Câu hỏi ôn tập.....	58
3.11	Bài tập.....	58
3.12	Bài tập thực hành.....	59
<b>CHƯƠNG 4.</b>	<b>MÃ HÓA KHÓA CÔNG KHAI .....</b>	<b>61</b>
4.1	Lý thuyết số .....	63
4.1.1	Một số khái niệm.....	63
4.1.2	Định lý Fermat .....	64
4.1.3	Phép logarit rời rạc.....	64
4.2	RSA .....	66
4.2.1	Nguyên tắc thực hiện của RSA .....	66
4.2.2	Ví dụ RSA.....	67
4.3	Độ phức tạp tính toán trong RSA .....	68
4.3.1	Phép tính mã hóa/giải mã.....	68
4.3.2	Phép tính sinh khóa .....	70
4.4	Độ an toàn của RSA .....	70

4.5	Bảo mật, chứng thực và không thoái thác với mã hóa khóa công khai.....	71
4.6	Trao đổi khóa.....	72
4.6.1	Trao đổi khóa công khai .....	73
4.6.2	Dùng mã hóa khóa công khai để trao đổi khóa bí mật .....	74
4.7	Phương pháp trao đổi khóa Diffie – Hellman .....	75
4.8	Câu hỏi ôn tập .....	76
4.9	Bài tập .....	77
4.10	Bài tập thực hành .....	77
<b>CHƯƠNG 5. MÃ CHỨNG THỰC THÔNG điệp, HÀM BẮM .....</b>		<b>79</b>
5.1	Mã chứng thực thông điệp .....	80
5.2	Hàm băm – Hash function.....	82
5.2.1	Bài toán ngày sinh nhật.....	82
5.2.2	Hàm băm MD5 và SHA-1 .....	84
5.2.3	HMAC .....	92
5.3	Hàm băm và chữ ký điện tử .....	95
5.4	Một số ứng dụng khác của hàm băm .....	92
5.4.1	Lưu trữ mật khẩu.....	92
5.4.2	Đấu giá trực tuyến.....	93
5.4.3	Download file .....	94
5.5	Câu hỏi ôn tập .....	96
5.6	Bài tập .....	97
5.7	Bài tập thực hành .....	97
<b>CHƯƠNG 6. GIAO THỨC .....</b>		<b>100</b>
6.1	Phát lại thông điệp (Replay Attack).....	100
6.2	Giao thức bảo mật .....	101
6.2.1	Định danh và trao đổi khóa phiên dùng mã hóa đối xứng với KDC .....	101
6.2.2	Định danh và trao đổi khóa phiên dùng mã hóa khóa công khai.....	102
6.3	Câu hỏi ôn tập .....	103
6.4	Bài tập .....	103
<b>CHƯƠNG 7. MỘT SỐ ỨNG DỤNG THỰC TIỄN .....</b>		<b>105</b>
7.1	Giới thiệu.....	105
7.2	Chứng thực X.509.....	105
7.2.1	Cấu trúc chứng thực.....	105
7.2.2	Phân cấp chứng thực.....	108
7.2.3	Các định dạng file của chứng chỉ X.509.....	109

7.3	Giao thức bảo mật web Secure Socket Layer version 3 - SSLv3.....	110
7.3.1	Giao thức bắt tay - SSL Handshaking Protocol .....	113
7.3.2	Giao thức truyền số liệu - SSL Record Protocol.....	116
7.3.3	SSL Session và SSL Connection .....	117
7.4	Giao thức bảo mật mạng cục bộ Keberos.....	117
7.4.1	Keberos version 4.....	117
7.5	Câu hỏi ôn tập.....	119
7.6	Bài tập thực hành.....	120
<b>CHƯƠNG 8. PHÁ MÃ VI SAI VÀ PHÁ MÃ TUYẾN TÍNH.....</b>		<b>121</b>
8.1	Phá mã vi sai (Differential Cryptanalysis) .....	121
8.2	Phá mã tuyến tính (Linear Cryptanalysis).....	126
8.3	Kết luận về nguyên tắc thiết kế mã khối. ....	128
<b>CHƯƠNG 9. ADVANCED ENCRYPTION STANDARD – AES .....</b>		<b>129</b>
9.1	Nhóm, vành, trường .....	129
9.1.1	Nhóm (Group).....	129
9.1.2	Vành (Ring).....	130
9.1.3	Trường (Field).....	130
9.2	Số học modulo và trường hữu hạn GF(p).....	131
9.3	Số học đa thức và trường hữu hạn GF(2 <sup>n</sup> ).....	132
9.3.1	Phép toán đa thức thông thường .....	132
9.3.2	Đa thức định nghĩa trên tập Z <sub>p</sub> .....	133
9.3.3	Phép modulo đa thức.....	134
9.3.4	Trường hữu hạn GF(2 <sup>n</sup> ).....	134
9.3.5	Ứng dụng GF(2 <sup>n</sup> ) trong mã hóa .....	136
9.3.6	Tính toán trong GF(2 <sup>n</sup> ).....	137
9.3.7	Tính toán trong GF(2 <sup>n</sup> ) với phần tử sinh.....	138
9.4	Mã hóa AES .....	139
9.4.1	Substitute bytes .....	141
9.4.2	Shift rows .....	145
9.4.3	Mix columns .....	145
9.4.4	Add row key.....	147
9.4.5	Expand key.....	147
9.4.6	Kết luận.....	148
<b>CHƯƠNG 10. MÃ HÓA ĐƯỜNG CONG ELLIPTIC .....</b>		<b>149</b>
10.1	Đường cong Elliptic trên số thực .....	149
10.2	Đường cong Elliptic trên trường Z <sub>p</sub> . ....	152

10.3 Đường cong Elliptic trên trường $GF(2^m)$ .	155
10.4 Đường cong Elliptic trong mã hóa - ECC	156
10.4.1 Trao đổi khóa EC Diffie-Hellman	156
10.4.2 Mã hóa và giải mã EC	157
10.4.3 Độ an toàn của ECC so với RSA	158
10.5 Chuẩn chữ ký điện tử (Digital Signature Standard – DSS).	158
<b>CHƯƠNG 11. MỘT SỐ VẤN ĐỀ AN TOÀN BẢO MẬT</b>	<b>161</b>
11.1 Giấu tin trong ảnh số	161
11.2 Lỗi phần mềm	162
11.2.1 Tràn bộ đệm (Buffer Overflow)	162
11.2.2 Chèn câu lệnh SQL (SQL Injection)	166
11.2.3 Chèn câu lệnh script (Cross-site Scripting XSS)	168
11.3 Bài tập thực hành	170
<b>PHỤ LỤC 1</b>	<b>172</b>
Chi Tiết các S-box của mã hóa DES	172
<b>PHỤ LỤC 2</b>	<b>174</b>
Thuật toán Euclid	174
Phương pháp kiểm tra số nguyên tố lớn Miller-Rabin	176
Định lý số dư Trung Hoa	179
Cài đặt giao thức SSL cho Web server IIS	181
<b>TÀI LIỆU THAM KHẢO</b>	<b>182</b>

# CHƯƠNG 1. GIỚI THIỆU VỀ AN TOÀN VÀ BẢO MẬT THÔNG TIN

## 1.1 Giới thiệu

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 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 (xem chương 2).
- 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 bảo mật 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)

Phần tiếp theo sau sẽ lần lượt trình bày các đặc điểm chính của hai mô hình trên.

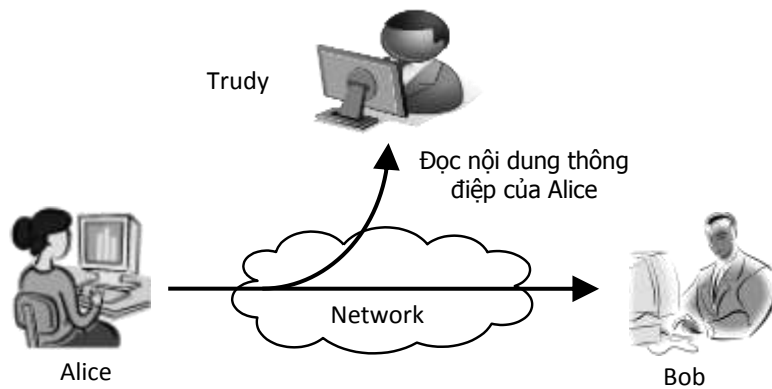
## 1.2 Bảo vệ thông tin trong quá trình truyền thông tin trên mạng

### 1.2.1 Các loại hình tấn công

Để xem xét những vấn đề bảo mật liên quan đến truyền thông trên mạng, chúng ta hãy lấy một bối cảnh sau: có ba nhân vật tên là Alice, Bob và Trudy, trong đó Alice và Bob thực hiện trao đổi thông tin với nhau, còn Trudy là kẻ xấu, đặt thiết bị can thiệp vào kênh truyền tin giữa Alice và Bob. Sau đây là các loại hành động tấn công của Trudy mà ảnh hưởng đến quá trình truyền tin giữa Alice và Bob:

- 1) Xem trộm thông tin (Release of Message Content)

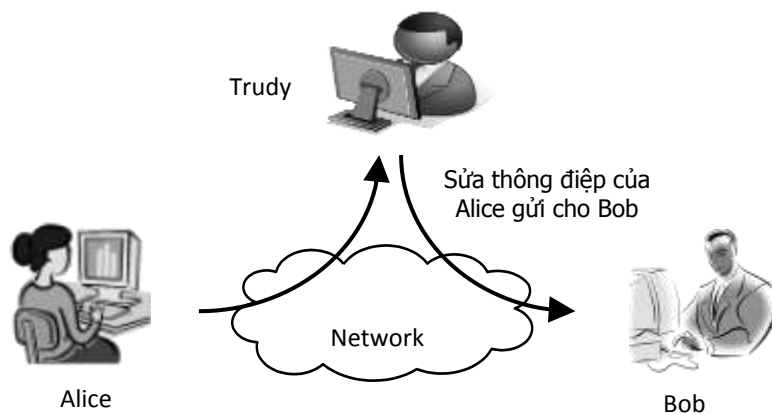
Trong trường hợp này Trudy chặn các thông điệp Alice gửi cho Bob, và xem được nội dung của thông điệp.



**Hình 1-1. Xem trộm thông điệp**

## 2) Thay đổi thông điệp (Modification of Message)

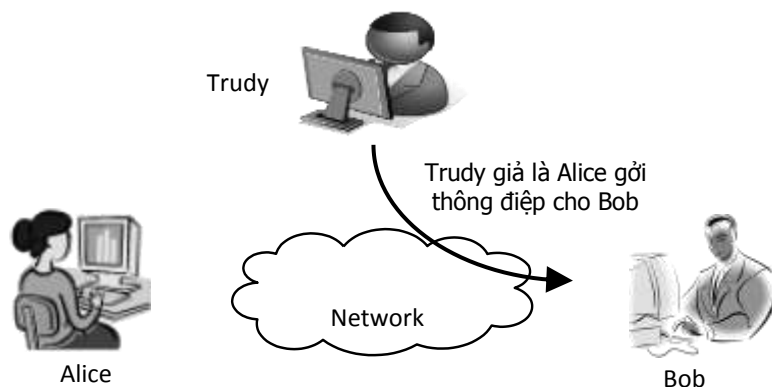
Trudy chặn các thông điệp Alice gửi cho Bob và ngăn không cho các thông điệp này đến đích. Sau đó Trudy thay đổi nội dung của thông điệp và gửi tiếp cho Bob. Bob nghĩ rằng nhận được thông điệp nguyên bản ban đầu của Alice mà không biết rằng chúng đã bị sửa đổi.



**Hình 1-2. Sửa thông điệp**

## 3) Mạo danh (Masquerade)

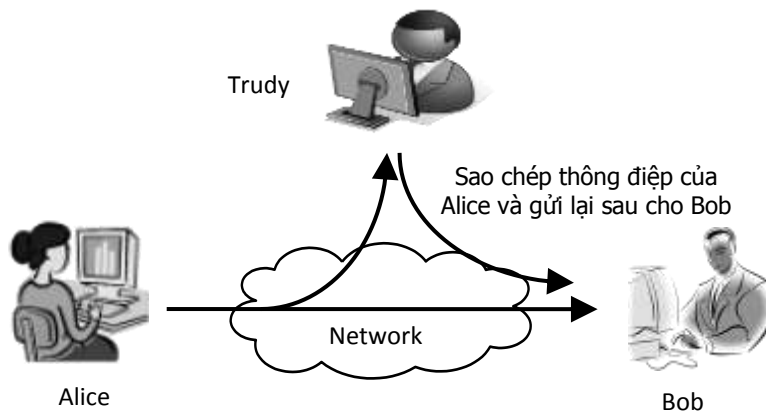
Trong trường hợp này Trudy giả là Alice gửi thông điệp cho Bob. Bob không biết điều này và nghĩ rằng thông điệp là của Alice.



**Hình 1-3. Mạo danh**

#### 4) Phát lại thông điệp (Replay)

Trudy sao chép lại thông điệp Alice gửi cho Bob. Sau đó một thời gian Trudy gửi bản sao chép này cho Bob. Bob tin rằng thông điệp thứ hai vẫn là từ Alice, nội dung hai thông điệp là giống nhau. 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 việc giả mạo thông điệp. Xét tình huống sau: giả sử Bob là ngân hàng còn Alice là một khách hàng. Alice gửi thông điệp đề nghị Bob chuyển cho Trudy 1000\$. Alice có áp dụng các biện pháp như chữ ký điện tử với mục đích không cho Trudy mạo danh cũng như sửa thông điệp. Tuy nhiên nếu Trudy sao chép và phát lại thông điệp thì các biện pháp bảo vệ này không có ý nghĩa. Bob tin rằng Alice gửi tiếp một thông điệp mới để chuyển thêm cho Trudy 1000\$ nữa.



Hình 1-4. Phát lại thông điệp

#### 1.2.2 Yêu cầu của một hệ truyền thông tin an toàn và bảo mật

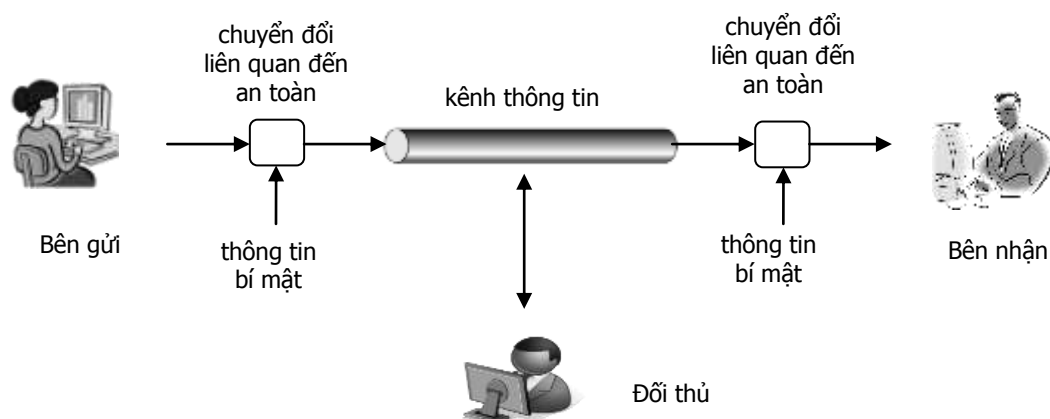
Phần trên đã trình bày các hình thức tấn công, một hệ truyền tin được gọi là an toàn và bảo mật thì phải có khả năng chống lại được các hình thức tấn công trên. Như vậy hệ truyền tin phải có các đặc tính sau:

- 1) Tính bảo mật (**Confidentiality**): Ngăn chặn được vấn đề xem trộm thông điệp.
- 2) Tính chứng thực (**Authentication**): Nhằm đảm bảo cho Bob rằng thông điệp mà Bob nhận được thực sự được gửi đi từ Alice, và không bị thay đổi trong quá trình truyền tin. Như vậy tính chứng thực ngăn chặn các hình thức tấn công sửa thông điệp, mạo danh, và phát lại thông điệp.
- 3) Tính không từ chối (**Nonrepudiation**): xét tình huống sau:

Giả sử Bob là nhân viên môi giới chứng khoán của Alice. Alice gửi thông điệp yêu cầu Bob mua cổ phiếu của công ty Z. Ngày hôm sau, giá cổ phiếu công ty này giảm hơn 50%. Thấy bị thiệt hại, Alice nói rằng Alice không gửi thông điệp nào cả và quy trách nhiệm cho Bob. Bob phải có cơ chế để xác định rằng chính Alice là người gửi mà Alice không thể từ chối trách nhiệm được.

Khái niệm chữ ký trên giấy mà con người đang sử dụng ngày nay là một cơ chế để bảo đảm tính chứng thực và tính không từ chối. Và trong lĩnh vực máy tính, người ta cũng thiết lập một cơ chế như vậy, cơ chế này được gọi là chữ ký điện tử.





**Hình 1-5. Mô hình bảo mật truyền thông tin trên mạng**

### 1.2.3 Vai trò của mật mã trong việc bảo mật thông tin trên mạng

Mật mã hay mã hóa dữ liệu (cryptography), là một công cụ cơ bản thiết yếu của bảo mật thông tin. Mật mã đáp ứng được các nhu cầu về tính bảo mật (confidentiality), tính chứng thực (authentication) và tính không từ chối (non-repudiation) của một hệ truyền tin.

Tài liệu này trước tiên trình bày về *mật mã cổ điển*. Những hệ mật mã cổ điển này tuy ngày nay tuy ít được sử dụng, nhưng chúng thể hiện những nguyên lý cơ bản được ứng dụng trong mật mã hiện đại. Dựa trên nền tảng đó, chúng ta sẽ tìm hiểu về *mã hóa đối xứng* và *mã hóa bất đối xứng*, chúng đóng vai trò quan trọng trong mật mã hiện đại. Bên cạnh đó chúng ta cũng sẽ tìm hiểu về *hàm Hash*, cũng là một công cụ bảo mật quan trọng mà có nhiều ứng dụng lý thú, trong đó có chữ ký điện tử.

Các chương 2, 3, 4, 5 sẽ lần lượt trình bày những nội dung liên quan đến mật mã.

### 1.2.4 Các giao thức (protocol) thực hiện bảo mật.

Sau khi tìm hiểu về mật mã, chúng ta sẽ tìm hiểu về cách ứng dụng chúng vào thực tế thông qua một số giao thức bảo mật phổ biến hiện nay là:

- Kerberos: là giao thức dùng để chứng thực dựa trên mã hóa đối xứng.
- Chuẩn chứng thực X509: dùng trong mã hóa khóa công khai.
- Secure Socket Layer (SSL): là giao thức bảo mật Web, được sử dụng phổ biến trong Web và thương mại điện tử.
- PGP và S/MIME: bảo mật thư điện tử email.

Mô hình lý thuyết và nội dung các giao thức trên được trình bày trong chương 6 và chương 7.

## 1.3 Bảo vệ hệ thống khỏi sự xâm nhập phá hoại từ bên ngoài

Ngày nay, khi mạng Internet đã kết nối các máy tính ở khắp nơi trên thế giới lại với nhau, thì vấn đề bảo vệ máy tính khỏi sự thâm nhập phá hoại từ bên ngoài là một điều cần thiết. Thông qua mạng Internet, các hacker có thể truy cập vào các máy tính trong một tổ chức (dùng telnet chẳng hạn), lấy trộm các dữ liệu quan trọng như mật khẩu, thẻ tín dụng, tài liệu... Hoặc đơn giản chỉ là phá hoại, gây trục trặc hệ thống mà tổ chức đó phải tốn nhiều chi phí để khôi phục lại tình trạng hoạt động bình thường.

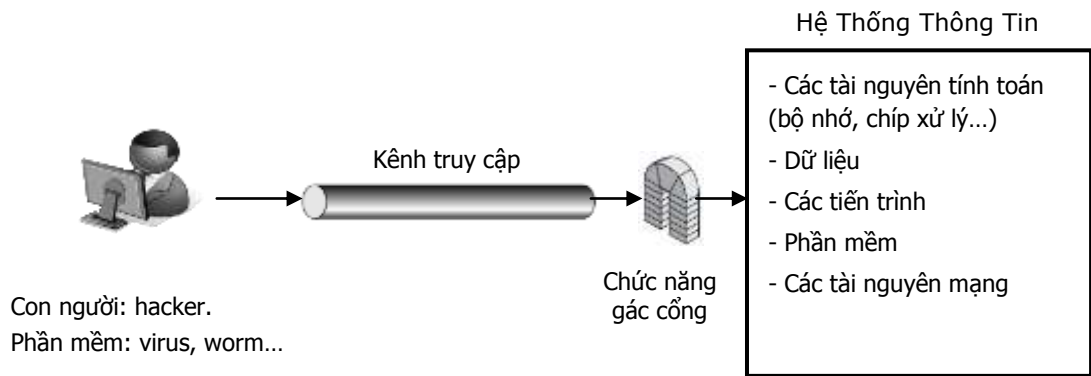
Để thực hiện việc bảo vệ này, người ta dùng khái niệm “kiểm soát truy cập” (Access Control). Khái niệm kiểm soát truy cập này có hai yếu tố sau:

- Chứng thực truy cập (Authentication): xác nhận rằng đối tượng (con người hay chương trình máy tính) được cấp phép truy cập vào hệ thống. Ví dụ: để sử dụng máy tính thì trước tiên đối tượng phải logon vào máy tính bằng username và password. Ngoài ra, còn có các phương pháp chứng thực khác như sinh trắc học (dấu vân tay, móng mắt...) hay dùng thẻ (thẻ ATM...).
- Phân quyền (Authorization): các hành động được phép thực hiện sau khi đã truy cập vào hệ thống. Ví dụ: bạn được cấp username và password để logon vào hệ điều hành, tuy nhiên bạn chỉ được cấp quyền để đọc một file nào đó. Hoặc bạn chỉ có quyền đọc file mà không có quyền xóa file.

Với nguyên tắc như vậy thì một máy tính hoặc một mạng máy tính được bảo vệ khỏi sự thâm nhập của các đối tượng không được phép. Tuy nhiên thực tế chúng ta vẫn nghe nói đến các vụ tấn công phá hoại. Để thực hiện điều đó, kẻ phá hoại tìm cách phá bỏ cơ chế Authentication và Authorization bằng các cách thức sau:

- Dùng các đoạn mã phá hoại (Malware): như virus, worm, trojan, backdoor... những đoạn mã độc này phát tán lan truyền từ máy tính này qua máy tính khác dựa trên sự bất cẩn của người sử dụng, hay dựa trên các lỗi của phần mềm. Lợi dụng các quyền được cấp cho người sử dụng (chẳng hạn rất nhiều người login vào máy tính với quyền administrator), các đoạn mã này thực hiện các lệnh phá hoại hoặc dò tìm password của quản trị hệ thống để gửi cho hacker, cài đặt các cổng hậu để hacker bên ngoài xâm nhập.
- Thực hiện các hành vi xâm phạm (Intrusion): việc thiết kế các phần mềm có nhiều lỗ hổng, dẫn đến các hacker lợi dụng để thực hiện những lệnh phá hoại. Những lệnh này thường là không được phép đối với người bên ngoài, nhưng lỗ hổng của phần mềm dẫn đến được phép. Trong những trường hợp đặc biệt, lỗ hổng phần mềm cho phép thực hiện những lệnh phá hoại mà ngay cả người thiết kế chương trình không ngờ tới. Hoặc hacker có thể sử dụng các cổng hậu do các backdoor tạo ra để xâm nhập.

Để khắc phục các hành động phá hoại này, người ta dùng các chương trình có chức năng gác cổng, phòng chống. Những chương trình này dò tìm virus hoặc dò tìm các hành vi xâm phạm để ngăn chặn chúng, không cho chúng thực hiện hoặc xâm nhập. Đó là các chương trình chống virus, chương trình firewall... Ngoài ra các nhà phát triển phần mềm cần có quy trình xây dựng và kiểm lỗi phần mềm nhằm hạn chế tối đa những lỗ hổng bảo mật có thể có.



**Hình 1-6. Mô hình phòng chống xâm nhập và phá hoại hệ thống**

Trong khuôn khổ của tài liệu này chỉ đề cập các nội dung về an toàn và bảo mật truyền tin trên mạng. Các bạn có thể tìm hiểu cụ thể hơn các nội dung liên quan đến bảo vệ chống xâm nhập trong [3].

#### **1.4 Câu hỏi ôn tập**

- 1) Nêu các hình thức tấn công trong quá trình truyền tin trên mạng.
- 2) Bảo vệ thông tin trong quá trình truyền đi trên mạng là gì?
- 3) Bảo vệ hệ thống khỏi sự tấn công bên ngoài là gì?

## CHƯƠNG 2. MÃ HÓA ĐỐI XỨNG CĂN BẢN

Trong chương này chúng ta sẽ tìm hiểu một số khái niệm cơ bản về phương pháp mã hóa đối xứng. Đây là phương pháp chủ yếu trong việc bảo đảm tính bảo mật (confidentiality) của một hệ truyền tin. Trước tiên, chúng ta sẽ tìm hiểu phương pháp mã hóa Caesar và sau đó là mô hình tổng quát của phương pháp mã hóa đối xứng cùng một số tính chất liên quan. Phần còn lại của chương trình bày một số phương pháp mã hóa cổ điển phổ biến khác.

### 2.1 Mã hóa Caesar

Thế kỷ thứ 3 trước công nguyên, nhà quân sự người La Mã Julius Caesar đã nghĩ ra phương pháp mã hóa một bản tin như sau: thay thế mỗi chữ trong bản tin bằng chữ đứng sau nó  $k$  vị trí trong bảng chữ cái. Giả sử chọn  $k = 3$ , ta có bảng chuyển đổi như sau:

Chữ ban đầu: a b c d e f g h i j k l m n o p q r s t u v w x y z

Chữ thay thế: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

(sau Z sẽ vòng lại là A, do đó  $x \rightarrow A$ ,  $y \rightarrow B$  và  $z \rightarrow C$ )

Giả sử có bản tin gốc (bản rõ): meet me after the toga party

Như vậy bản tin mã hóa (bản mã) sẽ là: PHHW PH DIWHU WKH WRJD SDUWB

Thay vì gửi trực tiếp bản rõ cho các cấp dưới, Caesar gửi bản mã. Khi cấp dưới nhận được bản mã, tiến hành giải mã theo quy trình ngược lại để có được bản rõ. Như vậy nếu đối thủ của Caesar có lấy được bản mã, thì cũng *không hiểu được ý nghĩa của bản mã*.

Chúng ta hãy gán cho mỗi chữ cái một con số nguyên từ 0 đến 25:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Phương pháp Caesar được biểu diễn như sau: với mỗi chữ cái  $p$  thay bằng chữ mã hóa  $C$ , trong đó:

$$C = (p + k) \bmod 26 \quad (\text{trong đó } \bmod \text{ là phép chia lấy số dư})$$

Và quá trình giải mã đơn giản là:

$$p = (C - k) \bmod 26$$

$k$  được gọi là khóa. Dĩ nhiên là Caesar và cấp dưới phải cùng dùng chung một giá trị khóa  $k$ , nếu không bản tin giải mã sẽ không giống bản rõ ban đầu.

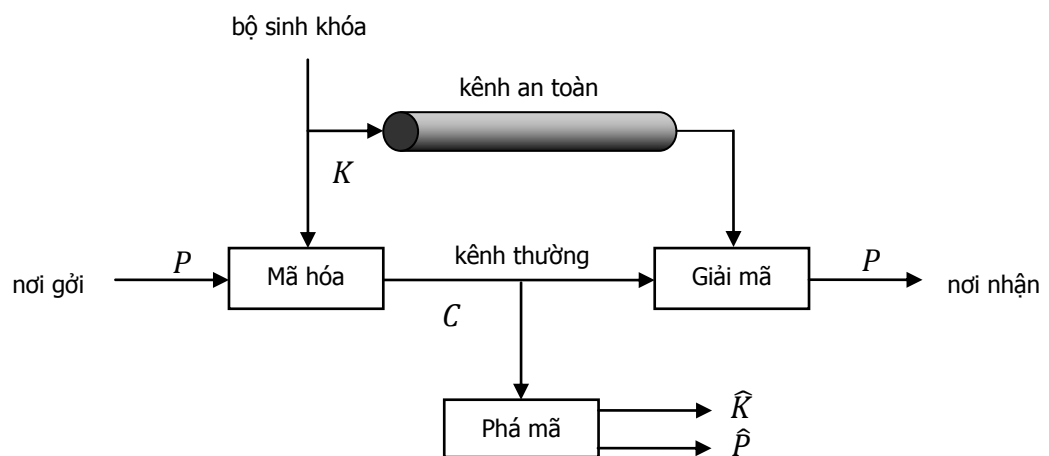
Ngày nay phương pháp mã hóa của Caesar không được xem là an toàn. Giả sử đối thủ của Caesar có được bản mã PHHW PH DIWHU WKH WRJD SDUWB và biết được phương pháp mã hóa và giải mã là phép cộng trừ modulo 26. Đối thủ có thể thử tất cả 25 trường hợp của  $k$  như sau:

KEY	PHHW	PH	DIWHU	WKH	WRJD	SDUWB
1	oggv	og	chvgt	vjg	vqic	rctva
2	nffu	nf	bgufs	uif	uphb	qbsuz
3	meet	me	after	the	toga	party
4	ldds	ld	zesdq	sgd	snfz	ozqsx
5	kccr	kc	ydrpc	rfc	rmey	nyprw
6	jbbq	jb	xcqbo	qeb	qldx	mxoqv
7	iaap	ia	wbpan	pda	pkcw	lwnpu
8	hzzo	hz	vaozm	ocz	objv	kvmot
9	gyyn	gy	uznyl	nby	niau	julns
10	fxxm	fx	tymxk	max	mhzt	itkmr
11	ewwl	ew	sxlwj	lzw	lgys	hsjlk
12	dvvk	dv	rwkvi	kyv	kfxr	grikp
13	cuuj	cu	qvjuh	jxu	jewq	fqhjo
14	btti	bt	puitg	iwt	idvp	epgin
15	assh	as	othsf	hvs	hcuo	dofhm
16	zrrg	zr	nsgre	gur	gbtn	cnegl
17	yqqf	yq	mrfqd	ftq	fasm	bmdfk
18	xppe	xp	lqepc	esp	ezrl	alcej
19	wood	wo	kpdob	dro	dyqk	zkbdi
20	vnnc	vn	jocna	cqn	cxpj	yjach
21	ummb	um	inbmz	bpm	bwoi	xizbg
22	tlla	tl	hmaly	aol	avnh	whyaf
23	skkz	sk	glzkx	znk	zumg	vgxze
24	rjyy	rj	fkyjw	ymj	ytlf	ufwyd
25	qiix	qi	ejxiv	xli	xske	tevxc

Trong 25 trường hợp trên, chỉ có trường hợp  $k=3$  thì bản giải mã tương ứng là có ý nghĩa. Do đó đối thủ có thể chắc chắn rằng ‘meet me after the toga party’ là bản rõ ban đầu.

## 2.2 Mô hình mã hóa đối xứng (Symmetric Ciphers)

Phương pháp Ceasar là phương pháp mã hóa đơn giản nhất của mã hóa đối xứng. Về mặt khái niệm, phương pháp mã hóa đối xứng tổng quát được biểu diễn bằng mô hình sau:



Hình 2-1. Mô hình mã hóa đối xứng

Mô hình trên gồm 5 yếu tố:

- Bản rõ  $P$  (plaintext)
- Thuật toán mã hóa  $E$  (encrypt algorithm)
- Khóa bí mật  $K$  (secret key)
- Bản mã  $C$  (ciphertext)
- Thuật toán giải mã  $D$  (decrypt algorithm)

Trong đó:

$$C = E(P, K)$$

$$P = D(C, K)$$

Thuật toán mã hóa và giải mã sử dụng chung một khóa, thuật toán giải mã là phép toán ngược của thuật toán mã hóa (trong mã hóa Ceasar,  $E$  là phép cộng còn  $D$  là phép trừ). Vì vậy mô hình trên được gọi là *phương pháp mã hóa đối xứng*.

Bản mã  $C$  được gửi đi trên kênh truyền. Do bản mã  $C$  đã được biến đổi so với bản rõ  $P$ , cho nên những người thứ ba can thiệp vào kênh truyền để lấy được bản mã  $C$ , thì *không hiểu được ý nghĩa của bản mã*. Đây chính là đặc điểm quan trọng của mã hóa, cho phép đảm bảo tính bảo mật (confidentiality) của một hệ truyền tin đã đề cập trong chương 1.

Một đặc tính quan trọng của mã hóa đối xứng là 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. Có thể đặt ra câu hỏi là nếu đã có một kênh an toàn để chuyển khóa như vậy thì tại sao không dùng kênh đó để chuyển bản tin, tại sao cần đến chuyện mã hóa? Câu trả lời là nội dung bản tin thì có thể rất dài, còn khóa thì thường là ngắn. Ngoài ra một khóa còn có thể áp dụng để truyền tin nhiều lần. Do đó nếu chỉ chuyển khóa trên kênh an toàn thì đỡ tốn kém chi phí.

Đặc tính quan trọng thứ hai của một hệ mã hóa đối xứng là tính an toàn của hệ mã. Như đã thấy ở phần mã hóa Ceasar, từ một bản mã có thể dễ dàng suy ra được bản rõ ban đầu mà không cần biết khóa bí mật. Hành động đi tìm bản rõ từ bản mã mà không cần khóa như vậy được gọi là *hành động phá mã* (cryptanalysis). Do đó một hệ mã hóa đối xứng được gọi là an toàn khi và chỉ khi nó không thể bị phá mã (điều kiện lý tưởng) hoặc thời gian phá mã là bất khả thi.

Trong phương pháp Ceasar, lý do mà phương pháp này kém an toàn là ở chỗ khóa  $k$  chỉ có 25 giá trị, do đó kẻ phá mã có thể thử được hết tất cả các trường hợp của khóa rất nhanh chóng. Phương pháp tấn công này được gọi là phương pháp vét cạn khóa (brute-force attack). Chỉ cần nói rộng miền giá trị của khóa thì có thể tăng thời gian phá mã đến một mức độ được coi là bất khả thi. Bảng dưới đây liệt kê một số ví dụ về thời gian phá mã trung bình tương ứng với kích thước của khóa.

Kích thước khóa (bít)	Số lượng khóa	Thời gian thực hiện (tốc độ thử: $10^3$ khóa/giây)	Thời gian thực hiện (tốc độ thử: $10^9$ khóa/giây)
32	$2^{32} \approx 4.3 \times 10^9$	35.8 phút	2.15 mili giây
56	$2^{56} \approx 7.2 \times 10^{16}$	1142 năm	10.01 giờ
128	$2^{128} \approx 3.4 \times 10^{38}$	$5.4 \times 10^{24}$ năm	$5.4 \times 10^{18}$ năm
168	$2^{168} \approx 3.7 \times 10^{50}$	$5.9 \times 10^{36}$ năm	$5.9 \times 10^{30}$ năm
hoán vị 26 ký tự	$26! \approx 4 \times 10^{26}$	$6.4 \times 10^{12}$ năm	$6.4 \times 10^6$ năm

(tốc độ CPU hiện nay khoảng  $3 \times 10^9$  Hz, tuổi vũ trụ vào khoảng  $\approx 10^{10}$  năm)

### Bảng 2-1. Thời gian vét cạn khóa theo kích thước khóa

Phần 2.3 sẽ trình bày phương pháp mã hóa đơn bảng, đây là phương pháp mà miền giá trị của khóa là  $26!$ . Do đó mã hóa đơn bảng an toàn đối với phương pháp tấn công vét cạn trên khóa.

Phần 2.6 trình bày phương pháp mã hóa One-Time Pad, phương pháp này có đặc tính là tồn tại rất nhiều khóa mà mỗi khóa khi đưa vào giải mã đều cho ra bản tin có ý nghĩa (phương pháp Caesar chỉ tồn tại *một* khóa giải mã cho ra bản tin có ý nghĩa). Do đó việc vét cạn khóa không có ý nghĩa đối với mã hóa One-Time Pad. Về mặt lý thuyết, phương pháp này được chứng minh là an toàn tuyệt đối.

Hiện nay, ngoài phương pháp One-Time Pad, người ta chưa tìm ra phương pháp mã hóa đối xứng an toàn tuyệt đối nào khác. Do đó chúng ta chấp nhận rằng một phương pháp mã hóa đối xứng là an toàn nếu phương pháp đó có điều kiện sau:

- Không tồn tại kỹ thuật tấn công tốt hơn phương pháp vét cạn khóa
- Miền giá trị khóa đủ lớn để việc vét cạn khóa là bất khả thi.

## 2.3 Mã hóa thay thế đơn bảng (Monoalphabetic Substitution Cipher)

Xét lại phương pháp Caesar với  $k=3$ :

Chữ ban đầu: a b c d e f g h i j k l m n o p q r s t u v w x y z

Chữ thay thế: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Phương pháp đơn bảng tổng quát hóa phương pháp Caesar bằng cách dùng mã hóa không phải là một dịch chuyển  $k$  vị trí của các chữ cái A, B, C, ... nữa mà là một *hoán vị* của 26 chữ cái này. Lúc này mỗi hoán vị được xem như là một khóa. Giả sử có hoán vị sau:

Chữ ban đầu: a b c d e f g h i j k l m n o p q r s t u v w x y z

Khóa : Z P B Y J R S K F L X Q N W V D H M G U T O I A E C

Như vậy bản rõ meet me after the toga party

được mã hóa thành: NJJU NJ ZRUJM UKJ UVSZ DZMUE

Quá trình giải mã được tiến hành ngược lại để cho ra bản rõ ban đầu.

Việc mã hóa được tiến hành bằng cách thay thế một chữ cái trong bản rõ thành một chữ cái trong bản mã, nên phương pháp này được gọi là phương pháp thay thế. Số lượng hoán vị của 26 chữ cái là  $26!$ , đây cũng chính là số lượng khóa của phương pháp này. Vì  $26!$  là một con số khá lớn nên việc tấn công phá mã vét cạn khóa là bất khả thi (6400 thiên niên kỷ với tốc độ thử khóa là  $10^9$  khóa/giây). Vì vậy mã hóa đơn bảng đã được xem là một phương pháp mã hóa an toàn trong suốt 1000 năm sau công nguyên.

Tuy nhiên vào thế kỷ thứ 9, một nhà hiền triết người Ả Rập tên là Al-Kindi đã phát hiện ra một phương pháp phá mã khả thi khác. Phương pháp phá mã này dựa trên nhận xét sau:

Trong ngôn ngữ tiếng Anh, tần suất sử dụng của các chữ cái không đều nhau, chữ E được sử dụng nhiều nhất, còn các chữ ít được sử dụng thường là Z, Q, J. Tương tự như vậy

đối với cụm 2 chữ cái (digram), cụm chữ TH được sử dụng nhiều nhất. Bảng sau thống kê tần suất sử dụng của các chữ cái, cụm 2 chữ, cụm 3 chữ (trigram) trong tiếng Anh:

Chữ cái (%)		Cụm 2 chữ (%)		Cụm 3 chữ (%)		Từ (%)	
E	13.05	TH	3.16	THE	4.72	THE	6.42
T	9.02	IN	1.54	ING	1.42	OF	4.02
O	8.21	ER	1.33	AND	1.13	AND	3.15
A	7.81	RE	1.30	ION	1.00	TO	2.36
N	7.28	AN	1.08	ENT	0.98	A	2.09
I	6.77	HE	1.08	FOR	0.76	IN	1.77
R	6.64	AR	1.02	TIO	0.75	THAT	1.25
S	6.46	EN	1.02	ERE	0.69	IS	1.03
H	5.85	TI	1.02	HER	0.68	I	0.94
D	4.11	TE	0.98	ATE	0.66	IT	0.93
L	3.60	AT	0.88	VER	0.63	FOR	0.77
C	2.93	ON	0.84	TER	0.62	AS	0.76
F	2.88	HA	0.84	THA	0.62	WITH	0.76
U	2.77	OU	0.72	ATI	0.59	WAS	0.72
M	2.62	IT	0.71	HAT	0.55	HIS	0.71
P	2.15	ES	0.69	ERS	0.54	HE	0.71
Y	1.51	ST	0.68	HIS	0.52	BE	0.63
W	1.49	OR	0.68	RES	0.50	NOT	0.61
G	1.39	NT	0.67	ILL	0.47	BY	0.57
B	1.28	HI	0.66	ARE	0.46	BUT	0.56
V	1.00	EA	0.64	CON	0.45	HAVE	0.55
K	0.42	VE	0.64	NCE	0.45	YOU	0.55
X	0.30	CO	0.59	ALL	0.44	WHICH	0.53
J	0.23	DE	0.55	EVE	0.44	ARE	0.50
Q	0.14	RA	0.55	ITH	0.44	ON	0.47
Z	0.09	RO	0.55	TED	0.44	OR	0.45

**Bảng 2-2. Bảng liệt kê tần suất chữ cái tiếng Anh**

Phương pháp mã hóa đơn bằng ánh xạ một chữ cái trong bản rõ thành một chữ cái khác trong bản mã. Do đó các chữ cái trong bản mã cũng sẽ tuân theo luật phân bố tần suất trên. Nếu chữ E được thay bằng chữ K thì tần suất xuất hiện của chữ K trong bản mã là 13.05%. Đây chính là cơ sở để thực hiện phá mã.

Xét bản mã sau:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ  
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZHUSX  
EPYEPDPDZSZUFPOMBZWPDPDTGUDTMOHMQ

Số lần xuất hiện của các chữ cái là:

A 2	F 3	K 0	P 17	U 9
B 2	G 3	L 0	Q 3	V 5
C 0	H 6	M 7	R 0	W 4
D 6	I 1	N 0	S 10	X 5
E 6	J 0	O 9	T 4	Y 2



				Z 13
--	--	--	--	------

Số lần xuất hiện của các digram (xuất hiện từ 2 lần trở lên) là:

DT 2	HZ 2	PE 2	TS 2	XU 2
DZ 2	MO 2	PO 3	UD 2	ZO 2
EP 3	OH 2	PP 2	UZ 3	ZS 2
FP 3	OP 3	SX 3	VU 2	ZU 2
HM 2	PD 3	SZ 2	WS 2	ZW 3

Do đó ta có thể đoán P là mã hóa của e, Z là mã hóa của t. Vì TH có tần suất cao nhất trong các digram nên trong 4 digram ZO, ZS, ZU, ZW có thể đoán ZW là th. Chú ý rằng trong dòng thứ nhất có cụm ZWSZ, nếu giả thiết rằng 4 chữ trên thuộc một từ thì từ đó có dạng th\_t, từ đó có thể kết luận rằng S là mã hóa của a (vì từ THAT có tần suất xuất hiện cao). Như vậy đến bước này, ta đã phá mã được như sau:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ

t a e e te a that e e a a

VUEPHZHMZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX

e t ta t ha e ee a e th t a

EPYEPDPDZSZUFPOMBZWPPDPTGUDTMOHMQ

e e e tat e thee e

Cứ tiếp tục như vậy, dĩ nhiên việc thử không phải lúc nào cũng suôn sẻ, có những lúc phải thử và sai nhiều lần. Cuối cùng ta có được bản giải mã sau khi đã tách từ như sau:

it was disclosed yesterday that several informal but  
direct contacts have been made with political  
representatives of the enemy in moscow

Như vậy việc phá mã dựa trên tần suất chữ cái tốn thời gian ít hơn nhiều so với con số 6400 thiên niên kỷ. Lý do là ứng một chữ cái trong bản gốc thì cũng là một chữ cái trong bản mã nên vẫn bảo toàn quy tắc phân bố tần suất của các chữ cái. Để khắc phục điểm yếu này, có hai phương pháp. Phương pháp thứ nhất là *mã hóa nhiều chữ cái cùng lúc*. Phương pháp thứ hai là làm sao để *một chữ cái trong bản rõ thì có tương ứng nhiều chữ cái khác nhau trong bản mã*. Hai phương án trên sẽ lần lượt được trình bày trong phần tiếp theo.

## 2.4 Mã hóa thay thế đa ký tự

### 2.4.1 Mã Playfair

Mã hóa Playfair xem hai ký tự đứng sát nhau là một đơn vị mã hóa, hai ký tự này được thay thế cùng lúc bằng hai ký tự khác. Playfair dùng một ma trận 5x5 các ký tự như sau:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Trong bảng trên, khóa là từ MONARCHY được điền vào các dòng đầu của bảng, các chữ cái còn lại được điền tiếp theo. Riêng hai chữ I, J được điền vào cùng một ô vì trong tiếng Anh, ít khi nhầm lẫn giữa chữ I và chữ J. Ví dụ, nếu gặp đoạn ký tự CL\_MATE, ta sẽ biết đó là từ CLIMATE chứ không phải là từ CLJMATE.

Trước khi mã hóa, bản rõ được tách ra thành các cặp ký tự. Nếu hai ký tự trong một cặp giống nhau thì sẽ được tách bằng chữ X (trong tiếng Anh ít khi có 2 ký tự X sát nhau). Ví dụ: từ balloon được tách thành ba lx lo on. Việc mã hóa từng cặp được thực hiện theo quy tắc:

- Nếu hai ký tự trong cặp thuộc cùng một hàng, thì được thay bằng hai ký tự tiếp theo trong hàng. Nếu đến cuối hàng thì quay về đầu hàng. Ví dụ cặp ar được mã hóa thành RM.
- Nếu hai ký tự trong cặp thuộc cùng một cột, thì được thay bằng hai ký tự tiếp theo trong cột. Nếu đến cuối cột thì quay về đầu cột. Ví dụ cặp ov được mã hóa thành HO.
- Trong các trường hợp còn lại, hai ký tự được mã hóa sẽ tạo thành đường chéo của một hình chữ nhật và được thay bằng 2 ký tự trên đường chéo kia. Ví dụ: hs trở thành BP (B cùng dòng với H và P cùng dòng với S); ea trở thành IM (hoặc JM)

Như vậy nếu chỉ xét trên 26 chữ cái thì mã khóa Playfair có  $26 \times 26 = 676$  cặp chữ cái, do đó các cặp chữ cái này ít bị chênh lệch về tần suất hơn so với sự chênh lệch tần suất của từng chữ cái. Ngoài ra số lượng các cặp chữ cái nhiều hơn cũng làm cho việc phá mã tần suất khó khăn hơn. Đây chính là lý do mà người ta tin rằng mã hóa Playfair không thể bị phá và được quân đội Anh sử dụng trong chiến tranh thế giới lần thứ nhất.

### 2.4.2 Mã Hill

Trong mã Hill, mỗi chữ cái được gán cho một con số nguyên từ 0 đến 25:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Mã Hill thực hiện mã hóa một lần  $m$  ký tự bản rõ (ký hiệu  $p_1, p_2, \dots, p_m$ ), thay thế thành  $m$  ký tự trong bản mã (ký hiệu  $c_1, c_2, \dots, c_m$ ). Việc thay thế này được thực hiện bằng  $m$  phương trình tuyến tính. Giả sử  $m = 3$ , chúng ta minh họa  $m$  phương trình đó như sau:

$$c_1 = k_{11}p_1 + k_{12}p_2 + k_{13}p_3 \mod 26$$

$$c_2 = k_{21}p_1 + k_{22}p_2 + k_{23}p_3 \mod 26$$

$$c_3 = k_{31}p_1 + k_{32}p_2 + k_{33}p_3 \mod 26$$

Ba phương trình trên có thể biểu diễn thành vector và phép nhân ma trận như sau:

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \mod 26$$

Hay:  $C = KP \mod 26$  với P và C là vector đại diện cho bản rõ và bản mã, còn K là ma trận dùng làm khóa.

Xét ví dụ bản rõ là paymoremoney cùng với khóa K là

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

Ba chữ cái đầu tiên của bản rõ tương ứng với vector (15, 0, 24). Vậy

$$\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix} \mod 26 = \begin{bmatrix} 11 \\ 13 \\ 18 \end{bmatrix} = \text{LNS}$$

Thực hiện tương tự ta có bản mã đầy đủ là LNSHDLEWMTRW

Để giải mã chúng ta cần sử dụng ma trận nghịch đảo của K là  $K^{-1}$ , tức là  $K^{-1}K \mod 26 = I$  là ma trận đơn vị (không phải mọi ma trận K đều tồn tại ma trận nghịch đảo, tuy nhiên nếu tồn tại thì ta có thể tìm được ma trận đơn vị bằng cách tính hạng *det* của ma trận)

Ví dụ ma trận nghịch đảo của ma trận trên là:

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

Vì :

$$\begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} = \begin{bmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{bmatrix} \mod 26 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Khi đó bảng giải mã là:  $K^{-1}C \mod 26 = K^{-1}KP \mod 26 = P$

Có thể thấy mã hóa Hill ẩn giấu các thông tin về tần suất nhiều hơn mã hóa Playfair do có thể mã hóa 3 hoặc nhiều hơn nữa các ký tự cùng lúc.

## 2.5 Mã hóa thay thế đa bảng (Polyalphabetic Substitution Cipher)

Với sự phát hiện ra quy luật phân bố tần suất, các nhà phá mã đang tạm thời chiếm ưu thế trong cuộc chiến mã hóa-phá mã. Cho đến thế kỷ thứ 15, một nhà ngoại giao người Pháp tên là Vigenere đã tìm ra phương án mã hóa thay thế đa bảng. Phương pháp Vigenere dựa trên bảng sau đây:

key	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

**Bảng 2-3. Bảng mã Vigenere**

Dòng thứ  $k$  của bảng là một mã hóa Caesar  $k-1$  vị trí. Ví dụ, dòng thứ 4, ứng với từ khóa D là mã hóa Caesar 3 vị trí. (Trong trường hợp tổng quát, mỗi dòng của bảng Vigenere không phải là một mã hóa Caesar nữa mà là một mã hóa đơn bảng, do đó có tên gọi là mã hóa đa bảng).

Để mã hóa một bản tin thì cần có một khóa có chiều dài bằng chiều dài bản tin. Thường thì khóa là một cụm từ nào đó và được viết lặp lại cho đến khi có chiều dài bằng chiều dài bản tin. Ví dụ với bản tin là ‘*We are discovered, save yourself*’ và khóa là từ *DECEPTIVE*, chúng ta mã hóa như sau:

plaintext:       wearediscoveredsaveyourself  
key:             DECEPTIVEDECEPTIVEDECEPTIVE  
ciphertext:     ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Trong ví dụ trên, ứng với chữ w trong bản rõ là chữ D trong khóa, nên dòng mã hóa thứ 4 ứng với khóa D trong bảng Vigenere được chọn. Do đó chữ w được mã hóa thành chữ Z. Tương tự như vậy cho các chữ còn lại.

Trong ví dụ trên, các chữ e trong bản rõ được mã hóa tương ứng thành I, T, G, T, H, M trong bản mã. Do đó phương pháp phá mã dựa trên thống kê tần suất chữ cái là không

Đến thế kỷ 19, nhà khoa học người Anh Charles Babbage, đã tìm ra cách phá mã Vigenere. Việc phá mã bằng cách thống kê sự lặp lại của các cụm từ để phỏng đoán chiều dài của khóa, trong ví dụ trên cụm từ VTW được lặp lại cách nhau 9 vị trí nên có thể đoán chiều dài của khóa là 9. Và từ đó có thể tách bản mã thành 9 phần, phần thứ nhất gồm các chữ 1, 10, 19, 28, ... phần thứ hai gồm các chữ 2, 11, 20, 29....cho đến phần thứ chín. Mỗi phần coi như được mã hóa bằng phương pháp mã hóa đơn bảng. Từ đó áp dụng phương pháp phá mã dựa trên tần suất chữ cái cho từng phần một. Cuối cùng ráp lại sẽ tìm ra được bản rõ.

Có thể thấy rằng điểm yếu của mã hóa đa bảng là do sự lặp lại các từ trong khóa, ví dụ từ *DECEPTIVE* được lặp đi lặp lại nhiều lần. Điều này làm cho vẫn tồn tại một mối liên quan giữa bản rõ và bản mã, ví dụ cụm từ **red** trong bản rõ được lặp lại thì cụm từ VTW cũng được lặp lại trong bản mã. Người phá mã tận dụng mối liên quan này để thực hiện phá mã. Do đó vấn đề ở đây là làm sao để giữa bản rõ và bản mã thật sự *ngẫu nhiên*, không tồn tại mối quan hệ nào. Để giải quyết vấn đề này, Joseph Mauborgne, giám đốc viện nghiên cứu mật mã của quân đội Mỹ, vào cuối cuộc chiến tranh thế giới lần thứ nhất, đã đề xuất phương án là dùng khóa ngẫu nhiên. Khóa ngẫu nhiên có chiều dài bằng chiều dài của bản rõ, mỗi khóa chỉ sử dụng một lần.

Bản mã C: BLWPOODEMJFBTZNJVJNJQOJORGGU

[illegible]

23

tin có ý nghĩa, do đó sẽ không biết được bản tin nào là bản rõ. Điều này chứng minh phương pháp One-Time Pad là phương pháp mã hóa an toàn tuyệt đối, và được xem là *ly thánh* của khoa mật mã cổ điển.

Một điều cần chú ý là để phương pháp One-Time Pad là an toàn tuyệt đối thì mỗi khóa chỉ được sử dụng một lần. Nếu một khóa được sử dụng nhiều lần thì cũng không khác gì việc lặp lại một từ trong khóa (ví dụ khóa có từ *DECEPTIVE* được lặp lại). Ngoài ra các khóa phải thật sự ngẫu nhiên với nhau. Nếu các điều này bị vi phạm thì sẽ có một mối liên hệ giữa bản rõ và bản mã, mà người phá mã sẽ tận dụng mối quan hệ này.

Tuy nhiên, phương pháp One-Time Pad không có ý nghĩa sử dụng thực tế. Vì chiều dài khóa bằng chiều dài bản tin, mỗi khóa chỉ sử dụng một lần, nên thay vì truyền khóa trên kênh an toàn thì có thể truyền trực tiếp bản rõ mà không cần quan tâm đến vấn đề mã hóa.

Vì vậy sau chiến tranh thế giới thứ nhất, người ta vẫn chưa thể tìm ra loại mật mã nào khác mà không bị phá mã. Mọi cố gắng vẫn là tìm cách thực hiện một mã thay thế đa bằng dùng một khóa dài, ít lặp lại, để hạn chế phá mã. Máy ENIGMA được quân đội Đức sử dụng trong chiến tranh thế giới lần 2 là một máy như vậy. Sử dụng máy ENIGMA, Đức đã chiếm ưu thế trong giai đoạn đầu của cuộc chiến. Tuy nhiên trong giai đoạn sau, các nhà phá mã người Ba Lan và Anh (trong đó có Alan Turing, người phá minh ra máy tính có thể lập trình được) đã tìm ra cách phá mã máy ENIGMA. Việc phá mã thực hiện được dựa vào một số điểm yếu trong khâu phân phối khóa của quân Đức. Điều này đóng vai trò quan trọng vào chiến thắng của quân đồng minh trong cuộc chiến.



**Hình 2-2. Hình minh họa cấu trúc máy ENIGMA, gõ chữ vào bàn phím, bản mã hiện ra ở các bóng đèn bên trên. (nguồn: Wikipedia)**

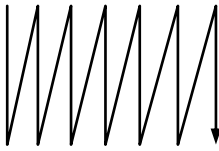
## **2.7 Mã hoán vị (Permutation Cipher)**

Các phương pháp mã hóa đã trình bày cho đến thời điểm này sử dụng phương thức thay một chữ cái trong bản rõ bằng một chữ cái khác trong bản mã (phương pháp thay thế).

Một cách thực hiện khác là xáo trộn thứ tự của các chữ cái trong bản rõ. Do thứ tự của các chữ cái bị mất đi nên người đọc không thể hiểu được ý nghĩa của bản tin dù các chữ đó không thay đổi.

Một cách thực hiện đơn giản là ghi bản rõ theo từng hàng, sau đó kết xuất bản mã dựa trên các cột. Ví dụ bản rõ ‘attackpostponeduntilthisnoon’ được viết lại thành bảng 4 x 7 như sau:

a	t	t	a	c	k	p
o	s	t	p	o	n	e
d	u	n	t	i	l	t
h	i	s	n	o	o	n



khi kết xuất theo từng cột thì có được bản mã: ‘AODHTSUITTNSAPTNCIOIKNLOPETN’

Một cơ chế phức tạp hơn là chúng ta có thể hoán vị các cột trước khi kết xuất bản mã. Ví dụ chọn một khóa là MONARCH, ta có thể hoán vị các cột:

M	O	N	A	R	C	H
a	t	t	a	c	k	p
o	s	t	p	o	n	e
d	u	n	t	i	l	t
h	i	s	n	o	o	n

→

A	C	H	M	N	O	R
a	k	p	a	t	t	c
p	n	e	o	t	s	o
t	l	t	d	n	u	i
n	o	n	h	s	i	o

và có được bản mã: ‘APTKNLOPETNAODHTTNSTSUICOIO’. Việc giải mã được tiến hành theo thứ tự ngược lại.

Để an toàn hơn nữa, có thể áp dụng phương pháp hoán vị 2 lần (double transposition), tức sau khi hoán vị lần 1, ta lại lấy kết quả đó hoán vị thêm một lần nữa:

M	O	N	A	R	C	H
a	p	t	n	k	n	l
o	p	e	t	n	a	o
d	h	t	t	n	s	t
s	u	i	c	o	i	o

→

A	C	H	M	N	O	R
n	n	l	a	t	p	k
t	a	o	o	e	p	n
t	s	t	d	t	h	n
c	i	o	s	i	u	o

Và cuối cùng bản mã là ‘NTTCNASILOTOAODSTETIPPHUKNNO’

Người ta đã đánh giá rằng phá mã phương pháp hoán vị 2 lần không phải là chuyện dễ dàng vì rất khó đoán ra được quy luật hoán vị. Ngoài ra không thể áp dụng được phương pháp phân tích tần suất chữ cái giống như phương pháp thay thế vì tần suất chữ cái của bản rõ và bản mã là giống nhau.

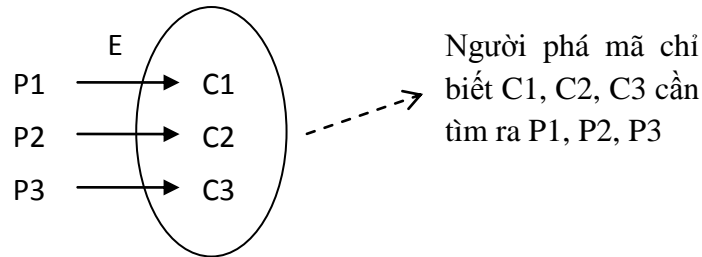
## 2.8 Tổng kết

Các phương pháp mã hóa cổ điển thường dựa trên hai phương thức. Cách thứ nhất là dùng phương thức thay thế một chữ cái trong bản rõ thành một chữ cái khác trong bản mã (substitution). Các mã hóa dùng phương thức này là mã hóa Caesar, mã hóa thay thế đơn bảng, đa bảng, one-time pad. Cách thứ hai là dùng phương thức hoán vị để thay đổi thứ tự

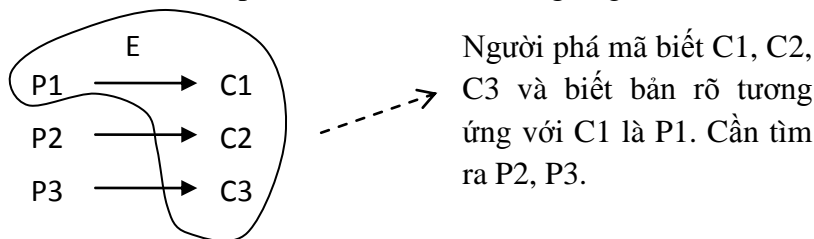
ban đầu của các chữ cái trong bản rõ (permutation). Hai phương thức này cũng đóng vai trò quan trọng trong mã hóa đối xứng hiện đại được trình bày trong chương tiếp theo.

Tron chương này chúng ta đã xem xét một số phương thức phá mã. Mục tiêu của việc phá mã là từ bản mã đi tìm bản rõ, hoặc khóa, hoặc cả hai. Chúng ta giả định rằng người phá mã biết rõ thuật toán mã hóa và giải mã (luật Kerchoff). Việc phá mã sẽ có 3 tình huống sau:

- 1) Chỉ biết bản mã (ciphertext-only): đây là trường hợp gây khó khăn nhất cho người phá mã. Các trường hợp phá mã được trình bày trong chương này thuộc dạng ciphertext only.



- 2) Biết một số cặp bản rõ – bản mã (known-plaintext): trong trường hợp này, người phá mã có được một vài cặp bản rõ và bản mã tương ứng.



Việc biết được một vài cặp bản rõ – bản mã làm cho người phá mã dễ dàng hơn trong việc tìm khóa. Ví dụ, đối với mã hóa Vigenere, nếu người phá mã chỉ cần biết một cặp bản rõ – bản mã thì sẽ dễ dàng suy ra được khóa, từ đó giải các bản mã khác mà cũng được mã hóa bằng khóa này.

Ví dụ: nếu biết bản mã : ZICVTWQNGRZGVTWAVZHCQYGLMGJ có bản rõ tương ứng là wearediscoveredsaveyourself, người phá mã có thể tra ngược bản Vigenere và tìm được khóa DECEPTIVE để giải các bản mã khác.

- 3) Một số cặp bản rõ – bản được lựa chọn (chosen-plaintext): trong trường hợp này, người phá mã có khả năng tự lựa một số bản rõ và quan sát được bản mã tương ứng. Ví dụ khi bạn đi ăn trưa và quên khóa máy, người phá mã có thể dùng chương trình mã hóa của bạn để thực hiện mã hóa một số bản tin chọn trước và có được bản mã tương ứng (dù không biết khóa).

Như vậy đối với trường hợp 2 và 3 thì người phá mã sẽ dễ dàng hơn trong việc phá mã so với trường hợp 1. Điều này đặt ra thách thức cho các nhà nghiên cứu là phải tìm ra các thuật toán mã hóa sao cho không thể bị phá mã không chỉ trong trường hợp 1 mà còn ngay cả trong trường hợp 2 và 3. Đó là một số thuật toán mà chúng ta sẽ tìm hiểu trong chương mã hóa đối xứng hiện đại.



## 2.9 Câu hỏi ôn tập

- 1) Tại sao khi gửi bản mã trên kênh truyền thì không sợ bị lộ thông tin?
- 2) Khóa là gì? Tại sao cần giữ bí mật khóa chỉ có người gửi và người nhận biết?
- 3) Tại sao lại gửi khóa qua kênh an toàn mà không gửi trực tiếp bản rõ trên kênh an toàn?
- 4) Phá mã khác giải mã ở điểm nào?
- 5) Phá mã theo hình thức vét cạn khóa thực hiện như thế nào? Cần làm gì để chống lại hình thức phá mã theo vét cạn khóa?
- 6) Các phương pháp Ceasar, mã hóa đơn bảng, đa bảng, one-time pad dùng nguyên tắc gì để mã hóa?
- 7) Phương pháp hoán vị dùng nguyên tắc gì để mã hóa?
- 8) Tại sao phương pháp mã hóa đơn bảng có thể bị tấn công phá mã dùng thống kê tần suất?
- 9) Hãy cho biết ý nghĩa của mã hóa Vigenere.
- 10) Phân biệt điểm khác nhau giữa ba trường hợp phá mã: ciphertext-only, known-plaintext, chosen-plaintext. Trong hai trường hợp known-plaintext và chosen-plaintext, người phá mã có lợi thế gì so với trường hợp ciphertext-only?

## 2.10 Bài Tập

1. Giải mã bản mã sau, giả sử mã hóa Ceasar được sử dụng để mã hóa với  $k=3$ :

IRXUVFRUHDQGVHYHQBHDUVDJR

2. Nếu một máy tính có thể thử 240 khóa /giây, tính thời gian phá mã bằng phương pháp vét cạn khóa nếu kích thước khóa là 128 bit (đáp án tính theo đơn vị năm).
3. Mã hóa bản rõ sau: 'enemy coming', dùng phương pháp mã hóa thay thế đơn bảng với khóa hoán vị K là: IAUTMOCSNREBDLHVWYFPZJXKGQ
4. Mã hóa từ 'explanation' bằng phương pháp Vigenere, từ khóa là LEG.
5. Mã hóa thông điệp sau bằng phương pháp hoán vị:

we are all together

biết khóa 24153

6. Phá mã bản mã sau, giả sử mã hóa Ceasar được sử dụng:

CSYEVIXIVQMREXIH

7. Phá mã bản mã sau (tiếng Anh), biết phương pháp mã hóa sử dụng là phương pháp thay thế đơn bảng:

GBSXUCGSZQGKGSQPKQKGLSKASPCGBGBKGUKGCEUKUZKGGBSQEICA  
CGKGCEUERWKLKUPKQQGCIICUAEUVSHQKGCEUPCGBCGQOEVSUNSU  
GKUZCGQSNLSHEHIEEDCUOGEPKHZGBSNKUGSUKUASERLSKASCUGB  
SLKACRCACUZSSZEUSBEXHKGSHWKLKUSQSKCHQTXKZHEUQBKZAEN  
NSUASZFENFCUOCUEKBXGBSWKLKUSQSKNFKQQKZEHGEGBSXUCGSZQ  
GKGSQKUZBCQAEIISKOSXSZSICVSHSZGEGBSQSAHSGKHMERQGKGSKR  
EHNKIHS LIMGEKHSASUGKNSHCAKUNSQKOSPBCISGBCQHSLIMQKGG  
SZGBKGCGQSSNSZXQSIISQQGEAEUGCUXSGBSSJCQGCUCOZCLIENKGCA  
USOEGCKGCEUQCGAEUGKCUSZUEGBHSGEHBUCGERPKHEHKHNSZKGGKAD

(Cần viết chương trình hỗ trợ phá mã, xem bài tập thực hành số 3)

8. Tương tự bài tập 7 cho bản mã sau (tiếng Anh):

PBFPVYFBQXZTYFPBFEQJHDXQVAPTPQJKTOYQWIPBVWLXTOXBTFXQWAXBVCXQWAXFQ  
JVWLEQNTQZQGGQLFXQWAKVWLXQWAEBIPBFXFQVXGTJVWLBTPQWAEBFPBFHCVLXBQU  
FEVWLXGDPEQVPQGVPPBFTIXPFHXZHVFAGFOTHFEBQUFTDHBZBQPOTHXTYFTODXQHFT  
DPTOGHFQPBQWAQJJTODXQHFOQPWTBDHHIXQVAPBFZQHCFWPFHPBFIPBQWKFABVYYDZ  
BOTHBPQPQJTTQOTOGHFQAPBFEQJHDXQVAVXEBQPEFZBVFOJIWFFACFCCFHQWUUVWFL  
QHGFVAVFXQHUFHILTTAVWAFFAWTEVOITDHFHFQAITIXPFHXAFQHEFZQWGLVWPTOFFA

9. Xét phương pháp Vigenere. Giả sử biết bản mã 'PVRLHFMJCRNFKKW' có bản rõ tương ứng là 'networksecurity'. Hãy tìm khóa K.

10. Xét bản mã được mã hóa bằng phương pháp One-Time Pad như sau: KITLKE

Nếu bản rõ là 'thrill' thì khóa là gì? Nếu bản rõ là 'tiller' thì khóa là gì?

11. Một trường hợp tổng quát của mã hóa Caesar là mã Affine, trong đó ký tự  $p$  được mã hóa thành ký tự  $C$  theo công thức:

$$C = E(p, [a, b]) = (ap + b) \bmod 26$$

Một yêu cầu của thuật toán mã hóa là tính đơn ánh, tức nếu  $p \neq q$  thì  $E(p) \neq E(q)$ . Mã Affine không phải là đơn ánh với mọi  $a$ . Ví dụ, với  $a=2, b=3$  thì  $E(0) = E(13) = 3$ .

a) Có điều kiện gì đặt ra cho  $b$  hay không? Tại sao?

b) Xác định những giá trị của  $a$  làm cho mã Affine không đơn ánh.

## 2.11 Bài Tập Thực Hành

1. Viết chương trình mã hóa và giải mã một file văn bản ASCII trên máy tính bằng phương pháp mã hóa Caesar.
2. Viết chương trình mã hóa và giải mã một file văn bản ASCII trên máy tính bằng phương pháp mã hóa Playfair.
3. Viết chương trình mã hóa và giải mã một file văn bản ASCII trên máy tính bằng phương pháp mã hóa Vigenere.
4. Viết chương trình hỗ trợ phá mã thay thế đơn bảng (bài tập 7 và 8), chương trình sẽ làm một số thao tác như thống kê tần suất các chữ cái, các digram, thực hiện phép thay thế...



### CHƯƠNG 3. MÃ HÓA ĐỐI XỨNG HIỆN ĐẠI

Đối tượng của các phương pháp mã hóa cổ điển là các bản tin ngôn ngữ, một đơn vị mã hóa là các *chữ cái* để áp dụng phương thức thay thế hay phương thức hoán vị.

Cùng với sự phát triển của máy tính, thông tin ngày một trở nên đa dạng, một bản tin bây giờ không chỉ đơn giản là bản tin gồm các chữ cái, mà có thể gồm cả các thông tin về định dạng văn bản như tài liệu HTML... Ngoài ra bản tin có thể xuất hiện dưới các loại hình khác như hình ảnh, video, âm thanh... Tất cả các bản tin đó đều được biểu diễn trên máy vi tính dưới dạng một dãy các số nhị phân. Trong máy tính các chữ cái được biểu diễn bằng mã ASCII.

Bản tin:                    attack

Mã ASCII:                97 116 116 97 99 107

Biểu diễn nhị phân: 01100001 01110100 01110100 01100001 01100011 01101011

Và cũng tương tự như bản tin ngôn ngữ, trong bản tin nhị phân cũng tồn tại một số đặc tính thống kê nào đó mà người phá mã có thể tận dụng để phá bản mã, dù rằng bản mã bây giờ tồn tại dưới dạng nhị phân. Mã hóa hiện đại quan tâm đến vấn đề *chống phá mã trong các trường hợp biết trước bản rõ (known-plaintext), hay bản rõ được lựa chọn (chosen-plaintext)*.

Để minh họa cách thức thực hiện của mã hóa đối xứng hiện đại, chúng ta sử dụng bản rõ là các chữ cái của một *ngôn ngữ* gồm có 8 chữ cái A, B, C, D, E, F, G, H trong đó mỗi chữ cái được biểu diễn bằng 3 bit.

Chữ cái	Nhị phân
A	000
B	001
C	010
D	011
E	100
F	101
G	110
H	111

Như vậy nếu có bản rõ là 'head' thì biểu diễn nhị phân tương ứng là: 111100000011

Giả sử dùng một khóa K gồm 4 bit 0101 để mã hóa bản rõ trên bằng phép XOR  $\oplus$ :

*bản rõ:*        1111 0000 0011        (head)

*khóa:*        0101 0101 0101

*bản mã:*        1010 0101 0110        (FBCG)

Trong phép mã hóa trên, đơn vị mã hóa không phải là một chữ cái mà là một khối 4 bit. Để giải mã, lấy bản mã XOR một lần nữa với khóa thì có lại bản rõ ban đầu.

Tuy nhiên, mã hóa bằng phép XOR như trên thì khá đơn giản ở hai điểm:

- Khóa được lặp lại, điều này bộc lộ điểm yếu giống như mã hóa Vigenere. Để khắc phục điều này, người ta dùng một bộ sinh số ngẫu nhiên để tạo khóa dài,

giả lập mã hóa One-Time pad. Đây là cơ sở thực hiện của mã dòng (stream cipher).

- Một khối được mã hóa bằng phép XOR với khóa. Điều này không an toàn vì chỉ cần biết *một cặp khối* bản rõ - bản mã (vd: 1111 và 1010), người phá mã dễ dàng tính được khóa. Để khắc phục điều này, người ta tìm ra các phép mã hóa phức tạp hơn phép XOR, và đây là cơ sở ra đời của mã khối (block cipher).

### 3.1 Mã dòng (Stream Cipher)

Mã dòng có các đặc tính sau:

- Kích thước một đơn vị mã hóa: gồm  $k$  bit. Bản rõ được chia thành các đơn vị mã hóa:  $P \rightarrow p_0 p_1 p_2 \dots p_{n-1}$  ( $p_i : k$  bit)
- Một bộ sinh dãy số ngẫu nhiên: dùng một khóa  $K$  ban đầu để sinh ra các số ngẫu nhiên có kích thước bằng kích thước đơn vị mã hóa:  
 $StreamCipher(K) \rightarrow S = s_0 s_1 s_2 \dots s_{n-1}$  ( $s_i : k$  bit)
- Mỗi số ngẫu nhiên được XOR với đơn vị mã hóa của bản rõ để có được bản mã.

$$c_0 = p_0 \oplus s_0, c_1 = p_1 \oplus s_1 \dots ; C = c_0 c_1 c_2 \dots c_{n-1}$$

Quá trình giải mã được thực hiện ngược lại, bản mã  $C$  được XOR với dãy số ngẫu nhiên  $S$  để cho ra lại bản rõ ban đầu:  $p_0 = c_0 \oplus s_0, p_1 = c_1 \oplus s_1 \dots$

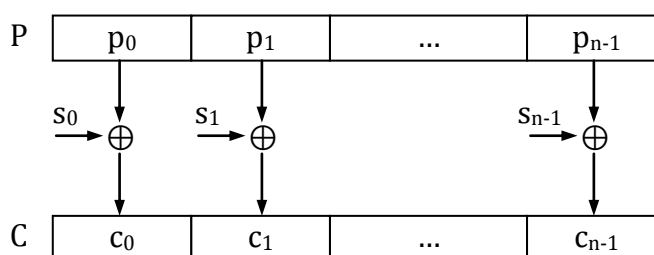
Trong ví dụ trên đơn vị mã hóa có chiều dài  $k = 4$  bit,  $n = 3$ :

$$p_0 = 1111, p_1 = 0000, p_2 = 0011$$

$$s_0 = s_1 = s_2 = K = 0101$$

$$c_0 = 1010, c_1 = 0101, c_2 = 0110$$

Ví dụ này không phải là mã dòng vì  $s_0, s_1, s_2$  lặp lại khóa  $K$ . Về phương diện khóa, ví dụ này giống mã Vigenere hơn. Đối với mã dòng, các số  $s_i$  được sinh ra phải đảm bảo một độ ngẫu nhiên nào đó (chu kỳ tuần hoàn dài):



**Hình 3-1. Mô hình mã dòng**

Như vậy có thể thấy mã hóa dòng tương tự như mã hóa Vigenere và mã hóa One-Time Pad. Điểm quan trọng nhất của các mã dòng là bộ sinh số ngẫu nhiên. Nếu chọn khóa có chiều dài ngắn như mã hóa Vigenere thì không bảo đảm an toàn, còn nếu chọn khóa có chiều dài bằng chiều dài bản tin như One-Time Pad thì lại không thực tế. Bộ sinh số của mã dòng cân bằng giữa hai điểm này, cho phép dùng một khóa ngắn nhưng dãy số sinh 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 hai phương pháp mã hóa dòng tiêu biểu là A5/1 và RC4.

### 3.1.1 A5/1

A5/1 được dùng trong mạng điện thoại GSM, để bảo mật dữ liệu trong quá trình liên lạc giữa máy điện thoại và trạm thu phát sóng vô tuyến. Đơn vị mã hóa của A5/1 là một بیت. Bộ sinh số mỗi lần sẽ sinh ra hoặc bit 0 hoặc bit 1 để sử dụng trong phép XOR. Để đơn giản, trước tiên chúng ta sẽ xem xét một mô hình thu nhỏ của A5/1 gọi là TinyA5/1.

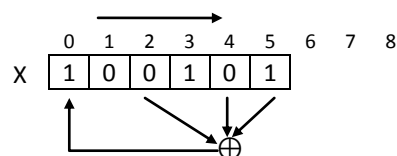
#### 1) TinyA5/1

Cơ chế thực hiện của bộ sinh số TinyA5/1 là như sau:

Bộ sinh số gồm 3 thanh ghi X, Y, Z. Thanh ghi X gồm 6 bit, ký hiệu là  $(x_0, x_1, \dots, x_5)$ . Thanh ghi Y gồm 8 bit  $(y_0, y_1, \dots, y_7)$ . Thanh ghi Z lưu 9 bit  $(z_0, z_1, \dots, z_8)$ . Khóa K ban đầu có chiều dài 23 bit và lần lượt được phân bố vào các thanh ghi:  $K \rightarrow XYZ$ . Các thanh ghi X, Y, Z được biến đổi theo 3 quy tắc:

1) **Quay X** gồm các thao tác:

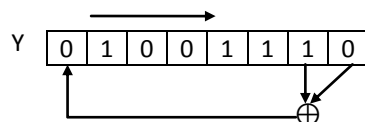
- $t = x_2 \oplus x_4 \oplus x_5$
- $x_j = x_{j-1}$  với  $j = 5, 4, 3, 2, 1$
- $x_0 = t$



Ví dụ: giả sử X là 100101, dẫn đến  $t = 0 \oplus 0 \oplus 1 = 1$ , vậy sau khi quay giá trị của X là 110010.

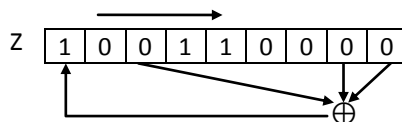
2) **Quay Y**: tương tự như quay X, quay Y là như sau:

- $t = y_6 \oplus y_7$
- $y_j = y_{j-1}$  với  $j = 7, 6, 5, \dots, 1$
- $y_0 = t$



3) **Quay Z**:

- $t = z_2 \oplus z_7 \oplus z_8$
- $z_j = z_{j-1}$  với  $j = 8, 7, 6, \dots, 1$
- $z_0 = t$



Cho ba bit  $x, y, z$ , ta định nghĩa một hàm  $maj(x, y, z)$  là hàm “chiếm đa số”, nghĩa là nếu trong 3 bit  $x, y, z$  có từ hai bit 0 trở lên thì hàm trả về giá trị 0, nếu không hàm trả về giá trị 1.

Tại bước sinh số thứ  $i$ , các phép tính sau được thực hiện:

$$m = maj(x_1, y_3, z_3)$$

If  $x_1 = m$  then thực hiện quay X

If  $y_3 = m$  then thực hiện quay Y

If  $z_3 = m$  then thực hiện quay Z

Và bit được sinh ra là:

$$s_i = x_5 \oplus y_7 \oplus z_8$$

Bit  $s_i$  được XOR với bit thứ  $i$  trong bản rõ để có được bit thứ  $i$  trong bản mã theo quy tắc của mã dòng.

Ví dụ: mã hóa bản rõ P=111 (chữ h) với khóa K là 100101. 01001110.100110000.

Ban đầu giá trị của các thanh ghi X, Y, Z là:

$$X = 1\underline{0}0101$$

$$Y = 010\underline{0}1110$$

$$Z = 100\underline{1}10000$$

Bước 0:  $x_1 = 0, y_3 = 0, z_3 = 1 \rightarrow m = 0 \rightarrow$  quay X, quay Y

$$X = 1\underline{1}0010$$

$$Y = 101\underline{0}0111 \rightarrow s_0 = 0 \oplus 1 \oplus 0 = 1$$

$$Z = 100\underline{1}10000$$

Bước 1:  $x_1 = 1, y_3 = 0, z_3 = 1 \rightarrow m = 1 \rightarrow$  quay X, quay Z

$$X = 1\underline{1}1001$$

$$Y = 101\underline{0}0111 \rightarrow s_1 = 1 \oplus 1 \oplus 0 = 0$$

$$Z = 010\underline{0}11000$$

Bước 2:  $x_1 = 1, y_3 = 0, z_3 = 0 \rightarrow m = 0 \rightarrow$  quay Y, quay Z

$$X = 111001$$

$$Y = 01010011 \rightarrow s_2 = 1 \oplus 1 \oplus 0 = 0$$

$$Z = 001001100$$

Vậy bản mã là  $C = 111 \oplus 100 = 011$  (chữ D)

## 2) A5/1

Về nguyên tắc bộ sinh số A5/1 hoạt động giống như TinyA5/1. Kích thước thanh ghi X, Y, Z lần lượt là 19, 22 và 23 bit. Các bước quay X, Y, Z cụ thể như sau:

### 1) Quay X:

- $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
- $x_j = x_{j-1}$  với  $j = 18, 17, 16, \dots, 1$
- $x_0 = t$

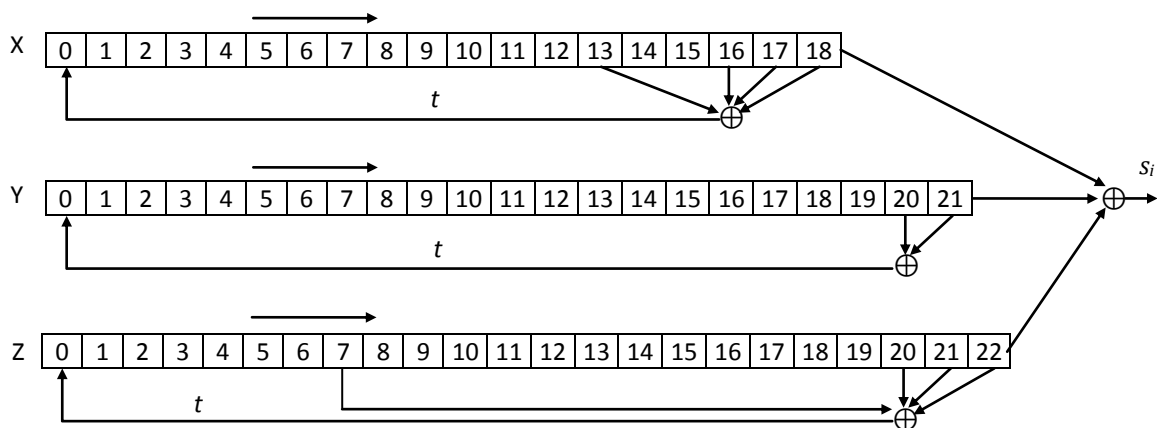
### 2) Quay Y:

- $t = y_{20} \oplus y_{21}$
- $y_j = y_{j-1}$  với  $j = 21, 20, 19, \dots, 1$
- $y_0 = t$

### 3) Quay Z:

- $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
- $z_j = z_{j-1}$  với  $j = 22, 21, 20, \dots, 1$
- $z_0 = t$

Hàm maj được tính trên 3 bit  $x_8, y_{10}, z_{10}$ . Sau khi quay xong bit sinh ra là:  $s_i = x_{18} \oplus y_{21} \oplus z_{22}$ . Toàn bộ quá trình sinh dãy số của A5/1 được minh họa qua hình bên dưới:



**Hình 3-2. Mã dòng A5/1**

Mã hóa A5/1 có thể được thực hiện dễ dàng bằng các thiết bị phần cứng, tốc độ nhanh. Do đó A5/1 đã từng được sử dụng để mã hóa các dữ liệu real-time như các dây bit audio. Ngày nay A5/1 được sử dụng để mã hóa dữ liệu cuộc gọi trong mạng điện thoại GSM.

### 3.1.2 RC4

RC4 được dùng trong giao thức SSL (xem phần 7.3) 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 mã hóa WEP của mạng Wireless LAN. Để đơn giản, chúng ta cũng sẽ xem xét một mô hình thu nhỏ của RC4 gọi là TinyRC4.

#### 1) TinyRC4

Khác với A5/1, đơn vị mã hóa của TinyRC4 là 3 bit. TinyRC4 dùng 2 mảng S và T mỗi mảng gồm 8 số nguyên 3 bit (từ 0 đến 7). Khóa là một dãy gồm N số nguyên 3 bit với N có thể lấy giá trị từ 1 đến 8. Bộ sinh số mỗi lần sinh ra 3 bit để sử dụng trong phép XOR. Quá trình sinh số của TinyRC4 gồm hai giai đoạn:

a) Giai đoạn khởi tạo:

```

/* Khởi tạo dãy số S và T */
for i = 0 to 7 do
    S[i] = i;
    T[i] = K[i mod N];
next i
/* Hoán vị dãy S */
j = 0;
for i = 0 to 7 do
    j = (j + S[i] + T[i]) mod 8;
    Swap(S[i], S[j]);
next i

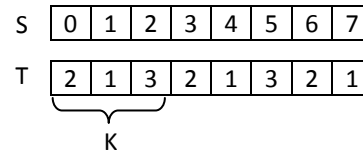
```

Trong giai đoạn này, trước tiên dãy S gồm các số nguyên 3 bit từ 0 đến 7 được sắp thứ tự tăng dần. Sau đó dựa trên các phần tử của khóa K, các phần tử của S được hoán vị lẫn nhau đến một mức độ ngẫu nhiên nào đó.

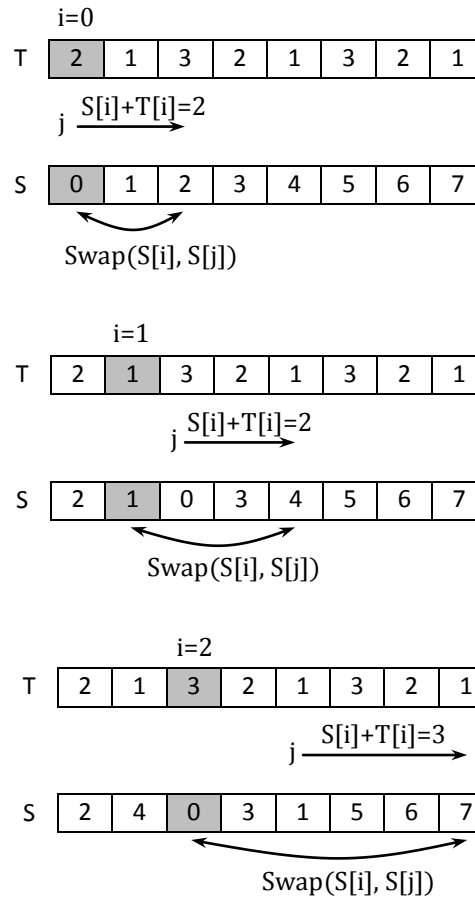
Ví dụ: mã hóa bản rõ P = 001000110 (từ 'bag') với khóa K gồm 3 số 2, 1, 3 (N=3)



- Khởi tạo S và T



- Hoán vị S



Quá trình thực hiện đến khi  $i=7$  và lúc đó dãy S là 6 0 7 1 2 3 5 4

- b) Giai đoạn sinh số:

---

```

i, j = 0;
while (true)
    i = (i + 1) mod 8;
    j = (j + S[i]) mod 8;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 8;
    k = S[t];
end while;

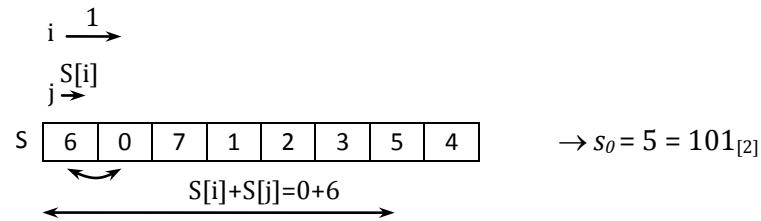
```

---

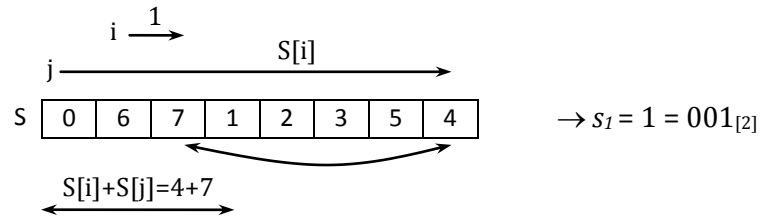
Trong giai đoạn này, các phần tử của S tiếp tục được hoán vị. Tại mỗi bước sinh số, hai phần tử của dãy S được chọn để tính ra số k 3 bit là số được dùng để XOR với đơn vị mã hóa của bản rõ.

Tiếp tục ví dụ trên, quá trình sinh số mã hóa bản rõ ‘bag’ thực hiện như sau:

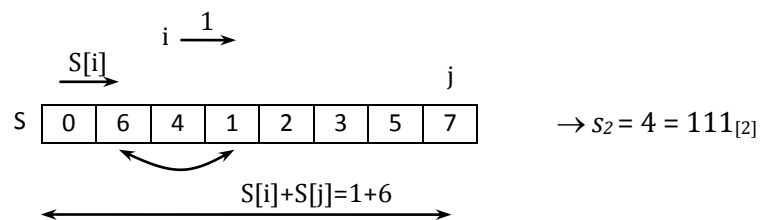
Bước 0:



Bước 1:



Bước 2:



Vậy bản mã là  $C = 001.000.110 \oplus 101.001.111 = 100.001.001$  (từ EBB)

## 2) RC4

Cơ chế hoạt động của RC4 cũng giống như TinyRC4 với các đặc tính sau:

- Đơn vị mã hóa của RC4 là một byte 8 bit.
- Mảng S và T gồm 256 số nguyên 8 bit
- Khóa K là một dãy gồm N số nguyên 8 bit với N có thể lấy giá trị từ 1 đến 256.
- Bộ sinh số mỗi lần sinh ra một byte để sử dụng trong phép XOR.

Hai giai đoạn của RC4 là:

a) Giai đoạn khởi tạo:

```

/* Khoi tao day S va T*/
for i = 0 to 255 do
    S[i] = i;
    T[i] = K[i mod N];
next i
/* Hoan vi day S */
j = 0;
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap(S[i], S[j]);
next i

```

b) Giai đoạn sinh số:

```
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];
end while;
```

Quá trình sinh số của RC4 cũng sinh ra dãy số ngẫu nhiên, khó đoán trước, vì vậy RC4 đạt được mức độ an toàn cao theo tinh thần của mã hóa One-Time Pad. 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 mã khối.

## 3.2 Mã khối (Block Cipher)

### 3.2.1 Mã khối an toàn lý tưởng

Phép toán XOR có một hạn chế là chỉ cần biết *một cặp khối* bản rõ và bản mã, người ta có thể dễ dàng suy ra được khóa và dùng khóa đó để giải các khối bản mã khác (known-plaintext attack). Xét lại ví dụ đầu chương:

*bản rõ:*      1111 0000 0011      (head)

*khóa:*        0101 0101 0101

*bản mã:*      1010 0101 0110      (FBCG)

Nếu biết bản mã  $c_0 = 1010$  có bản rõ tương ứng là  $p_0 = 1111$ , thì có thể dễ dàng suy ra khóa là 0101. Nói một cách tổng quát, nếu giữa bản rõ  $P$  và bản mã  $C$  có mối liên hệ toán học thì việc biết một số cặp bản rõ-bản mã giúp ta có thể tính được khóa  $K$ . (như trong trường hợp mã Hill)

Do đó để chống phá mã trong trường hợp known-plaintext hay choosen-plaintext, chỉ có thể là làm cho  $P$  và  $C$  không có mối liên hệ toán học. Điều này chỉ có thể thực hiện được nếu ta lập một bản tra cứu ngẫu nhiên giữa bản rõ và bản mã. Ví dụ:

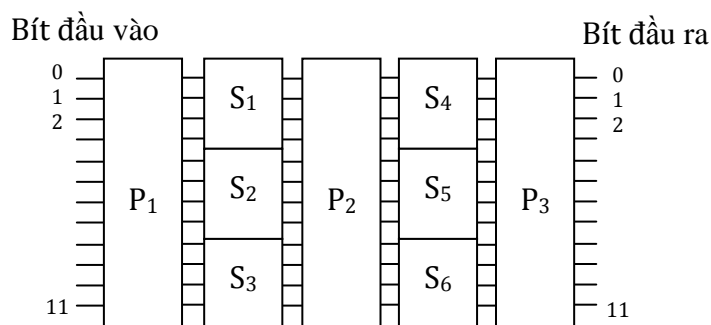
Bản rõ	Bản mã
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Lúc này khóa là toàn bộ bảng trên. Người gửi cũng như người nhận phải biết toàn bộ bảng trên để mã hóa và giải mã. Đối với người phá mã, nếu biết một số cặp bản rõ - bản mã thì cũng chỉ biết được một phần của bảng tra cứu trên. Do đó không suy ra được bản rõ cho các bản mã còn lại. Hay nói cách khác, muốn phá mã thì phải biết được tất cả các cặp bản rõ và bản mã. Nếu chọn kích thước của khối là 64 bit thì số dòng của bảng khóa là  $2^{64}$ , một con số rất lớn (và có khoảng  $2^{64}$ ! bảng khóa như vậy). Lúc này việc nắm tất cả các cặp bản rõ-bản mã của bảng khóa là điều không thể đối với người phá mã. Trường hợp này ta gọi là *mã khối an toàn lý tưởng*.

Tuy nhiên, khi kích thước khối lớn thì số dòng của bảng khóa cũng lớn và gây trở ngại cho việc lưu trữ cũng như trao đổi khóa giữa người gửi và người nhận. Bảng khóa có  $2^{64}$  dòng mỗi dòng 64 bit do đó kích thước khóa sẽ là  $64 \times 2^{64} = 2^{70} \approx 10^{21}$  bit. Do đó mã khối an toàn lý tưởng là không khả thi trong thực tế.

### 3.2.2 Mạng SPN

Trong thực tế, người ta chỉ tìm cách để chỉ cần dùng một khóa có kích thước ngắn để giả lập một bảng tra cứu có độ an toàn xấp xỉ độ an toàn của mã khối lý tưởng. Cách thực hiện là kết hợp hai hay nhiều mã hóa đơn giản lại với nhau để tạo thành một mã hóa tổng (product cipher), trong đó mã hóa tổng an toàn hơn rất nhiều so với các mã hóa thành phần. Các mã hóa đơn giản thường là phép thay thế (substitution, S-box) và hoán vị (Permutation, P-box). Do đó người ta hay gọi mã hóa tổng là Substitution-Permutation Network (mạng SPN). Hình dưới minh họa một mạng SP.



Việc kết hợp các S-box và P-box tạo ra hai tính chất quan trọng của mã hóa là tính khuếch tán (diffusion) và tính gây lẫn (confusion). Hai tính chất này do Claude Shannon giới thiệu vào năm 1946, và là cơ sở của tất cả các mã khối hiện nay.

- Tính khuếch tán: một bit của bản rõ tác động đến tất cả các bit của bản mã, hay nói cách khác, một bit của bản mã chịu tác động của tất cả các bit trong bản rõ. Việc làm như vậy nhằm làm giảm tối đa mối liên quan giữa bản rõ và bản mã, ngăn chặn việc suy ra lại khóa. Tính chất này có được dựa vào sử dụng P-box kết hợp S-box.
- Tính gây lẫn: làm phức tạp hóa mối liên quan giữa bản mã và khóa. Do đó cũng ngăn chặn việc suy ra lại khóa. Tính chất này có được dựa vào sử dụng S-box.

### 3.2.3 Mô hình mã Feistel

Mô hình mã Feistel là một dạng tiếp cận khác so với mạng SP. Mô hình do Horst Feistel đề xuất, cũng là sự kết hợp các phép thay thế và hoán vị. Trong hệ mã Feistel, bản rõ sẽ được biến đổi qua một số vòng để cho ra bản mã cuối cùng:

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

Trong đó bản rõ  $P$  và các bản mã  $C_i$  được chia thành nửa trái và nửa phải:

$$P = (L_0, R_0)$$

$$C_i = (L_i, R_i) \quad i = 1, 2, \dots, n$$

Quy tắc biến đổi các nửa trái phải này qua các vòng được thực hiện như sau:

$$L_i = R_{i-1}$$

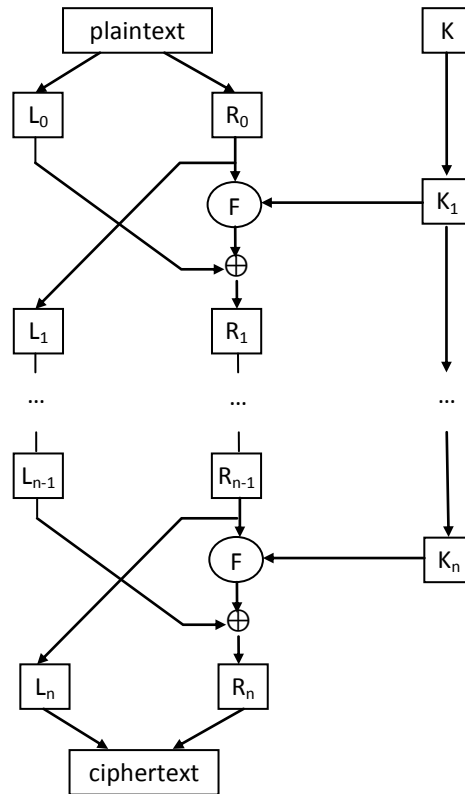
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

$K_i$  là một khóa con cho vòng thứ  $i$ . Khóa con này được sinh ra từ khóa  $K$  ban đầu theo một thuật toán sinh khóa con (key schedule):  $K \rightarrow K_1 \rightarrow K_2 \rightarrow \dots \rightarrow K_n$

$F$  là một hàm mã hóa dùng chung cho tất cả các vòng. Hàm  $F$  đóng vai trò như là phép thay thế còn việc hoán đổi các nửa trái phải có vai trò hoán vị. Bản mã  $C$  được tính từ kết xuất của vòng cuối cùng:

$$C = C_n = (L_n, R_n)$$

Sơ đồ tính toán của hệ mã Feistel được thể hiện trong hình bên dưới:



**Hình 3-3. Mô hình mã khối Feistel**

Để giải mã quá trình được thực hiện qua các vòng theo thứ tự ngược lại:

$$C \rightarrow L_n, R_n$$

$$R_{i-1} = L_i \quad (\text{theo mã hóa } L_i = R_{i-1})$$

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i) \quad (\text{theo mã hóa } R_i = L_{i-1} \oplus F(R_{i-1}, K_i))$$

Và cuối cùng bản rõ là  $P = (L_0, R_0)$ .

Hệ mã Feistel có điểm quan trọng là việc chia các bản mã thành hai nửa trái phải giúp cho hàm  $F$  không cần khả nghịch (không cần có  $F^{-1}$ ). Mã hóa và giải mã đều dùng chiều thuận của hàm  $F$ . Hàm  $F$  và thuật toán sinh khóa con càng phức tạp thì càng khó phá mã.

Ứng với các hàm  $F$  và thuật toán sinh khóa con khác nhau thì ta sẽ có các phương pháp mã hóa khác nhau, phần tiếp theo sẽ trình bày mã hóa DES, là một phương pháp mã hóa dựa trên nguyên tắc của hệ mã Feistel.

### 3.3 Mã TinyDES

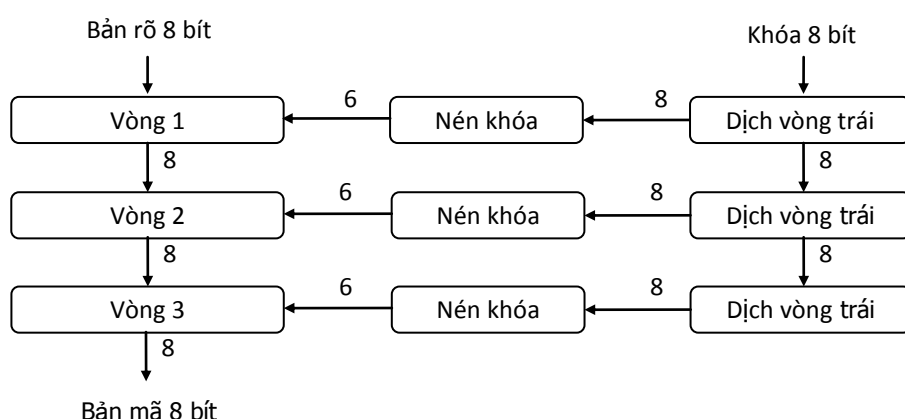
Vào năm 1973, khi lĩnh vực máy tính ngày càng phát triển, nhu cầu ứng dụng bảo mật vào các mục đích dân sự được đặt ra. Lúc này Cục tiêu chuẩn quốc gia Hoa Kỳ kêu gọi các công ty Mỹ thiết lập một chuẩn mã hóa quốc gia. Mã hóa Lucifer của công ty IBM được chọn và sau một vài sửa đổi của cơ quan an ninh Hoa Kỳ, mã hóa Lucifer đã trở thành mã tiêu chuẩn DES (Data Encryption Standard). Qua quá trình sử dụng mã DES đã chứng tỏ độ an toàn cao và được sử dụng rộng rãi.

Tương tự như mã dòng A5/1 và RC4, chúng ta cũng sẽ xem xét một mô hình thu nhỏ của mã DES là TinyDES.

Mã TinyDES có các tính chất sau:

- Là mã thuộc hệ mã Feistel gồm 3 vòng
- Kích thước của khối là 8 bit
- Kích thước khóa là 8 bit
- Mỗi vòng của TinyDES dùng khóa con có kích thước 6 bit được trích ra từ khóa chính.

Hình dưới đây minh họa các vòng của mã TinyDES

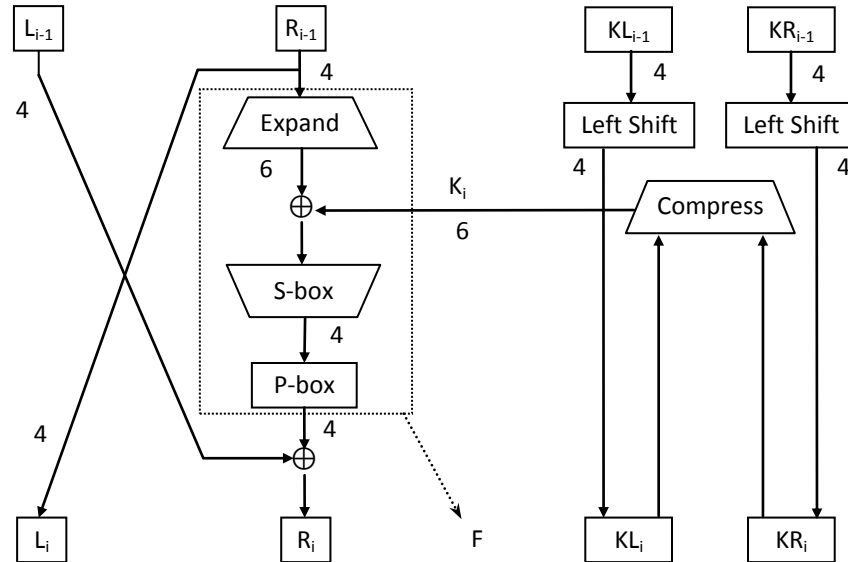


**Hình 3-4. Các vòng Feistel của mã TinyDES**

Sơ đồ mã TinyDES trên gồm hai phần, phần thứ nhất là các vòng Feistel, phần thứ hai là thuật toán sinh khóa con. Chúng ta sẽ lần lượt đi vào chi tiết của từng phần.

#### 3.3.1 Các vòng của TinyDES

Hình sau minh họa một vòng Feistel của TinyDES



**Hình 3-5. Cấu trúc một vòng của mã TinyDES**

Trong TinyDES, hàm F của Feistel là:

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

Trong đó hàm *Expand* vừa mở rộng vừa hoán vị  $R_{i-1}$  từ 4 bit lên 6 bit. Hàm *S-boxes* biến đổi một số 6 bit đầu vào thành một số 4 bit đầu ra. Hàm *P-box* là một hoán vị 4 bit. Mô tả của các hàm trên là như sau:

- *Expand*: gọi 4 bit của  $R_{i-1}$  là  $b_0b_1b_2b_3$ . Hàm *Expand* hoán vị và mở rộng 4 bit thành 6 bit cho ra kết quả:  $b_2b_3b_1b_2b_1b_0$ .  
Ví dụ:  $R_0 = 0110 \Rightarrow Expand(R_0) = 101110$
- *S-box*: Gọi  $b_0b_1b_2b_3b_4b_5$  là 6 bit đầu vào của *S-box*, ứng với mỗi trường hợp của 6 bit đầu vào sẽ có 4 bit đầu ra. Việc tính các bit đầu ra dựa trên bảng sau:

		$b_1b_2b_3b_4$															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
$b_0b_5$	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

Hai bit  $b_0b_1$  xác định thứ tự hàng, bốn bit  $b_1b_2b_3b_4$  xác định thứ tự cột của bảng, Từ đó dựa vào bảng tính được 4 bit đầu ra. Để cho đơn giản, ta có thể viết lại bảng trên dưới dạng số thập lục phân.

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
	1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
	2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
	3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

Ví dụ:  $X = 101010$ . Tra bảng ta có  $S\text{-}box(X) = 0110$ .

- *P-box*: thực hiện hoán vị 4 bit đầu  $b_0b_1b_2b_3$  cho ra kết quả  $b_2b_0b_3b_1$ .

### 3.3.2 Thuật toán sinh khóa con của TinyDES

Khóa  $K$  8 bit ban đầu được chia thành 2 nửa trái phải  $KL_0$  và  $KR_0$ , mỗi nửa có kích thước 4 bit. Tại vòng thứ nhất  $KL_0$  và  $KR_0$  được dịch vòng trái 1 bit để có được  $KL_1$  và  $KR_1$ . Tại vòng thứ hai  $KL_1$  và  $KR_1$  được dịch vòng trái 2 bit để có được  $KL_2$  và  $KR_2$ . Tại vòng tại vòng thứ 3  $KL_2$  và  $KR_2$  được dịch vòng trái 1 bit để có  $KL_3$  và  $KR_3$ .

Cuối cùng khóa  $K_i$  của mỗi vòng được tạo ra bằng cách hoán vị và nén (compress) 8 bit của  $KL_i$  và  $KR_i$  ( $k_0k_1k_2k_3k_4k_5k_6k_7$ ) thành kết quả gồm 6 bit :  $k_5k_1k_3k_2k_7k_0$ .

### 3.3.3 Ví dụ về TinyDES

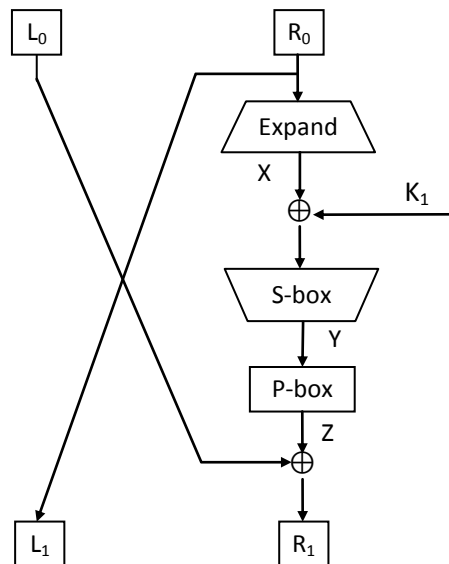
Ví dụ: mã hóa bản rõ  $P = 0101.1100$  (5C) với khóa  $K = 1001.1010$

- $L_0 = 0101, R_0 = 1100, KL_0 = 1001, KR_0 = 1010$
- Vòng 1:
  - $L_1 = R_0 = 1100, Expand(R_0) = 001011$
  - $KL_1 = KL_0 \ll 1 = 0011, KR_1 = KR_0 \ll 1 = 0101$
  - $K_1 = Compress(KL_1KR_1) = 101110$
  - $Expand(R_0) \oplus K_1 = 100101$
  - $S-box(100101) = 1000$
  - $F_1 = P-box(1000) = 0100$
  - $R_1 = L_0 \oplus F_1 = 0001$
- Vòng 2:
  - $L_2 = R_1 = 0001, Expand(R_1) = 010000$
  - $KL_2 = KL_1 \ll 2 = 1100, KR_2 = KR_1 \ll 2 = 0101$
  - $K_2 = Compress(KL_2KR_2) = 110011$
  - $Expand(R_1) \oplus K_2 = 100011$
  - $S-box(100011) = 1100$
  - $F_2 = P-box(1100) = 0101$
  - $R_2 = L_1 \oplus F_2 = 1001$
- Vòng 3:
  - $L_3 = R_2 = 1001, Expand(R_2) = 010001$
  - $KL_3 = KL_2 \ll 1 = 1001, KR_3 = KR_2 \ll 1 = 1010$
  - $K_3 = Compress(KL_3KR_3) = 001001$
  - $Expand(R_2) \oplus K_3 = 011000$
  - $S-box(011000) = 0101$
  - $F_3 = P-box(0101) = 0011$
  - $R_3 = L_2 \oplus F_3 = 0010$
- Kết quả  $C = L_3R_3 = 1001.0010$  (hệ thập lục phân: 92)

### 3.3.4 Khả năng chống phá mã known-plaintext của TinyDES

Xét trường hợp mã TinyDES chỉ có 1 vòng, tức  $P = (L_0, R_0)$  và  $C = (L_1, R_1)$ .





Trong trường hợp này người phá mã biết  $P$  và  $C$ , tuy nhiên không biết  $K$ . Giả sử  $P = 0101.1100$  và  $C = 1100.0001$ . Người phá mã tiến hành tính  $K$  như sau:

- Từ  $R_0$  tính  $X = 001011$ .
- Từ  $L_0$  và  $R_1$  tính  $Z = 0100$ , và từ  $Z$  tính  $Y = 1000$ .
- Tra cứu bảng  $S\text{-box}$  với đầu ra là 1000, ta xác định được các đầu vào  $X \oplus K_1$  có thể xảy ra là:  $\{100101, 100111, 001110, 011111\}$
- Như vậy khóa  $K_1$  là một trong các giá trị  $\{101110, 101100, 000101, 010100\}$
- Thử tiếp với 1 vài cặp bản rõ-bản mã khác ta sẽ tìm được  $K_1 = 101110$  và từ đó tính được  $K = 1001.1010$

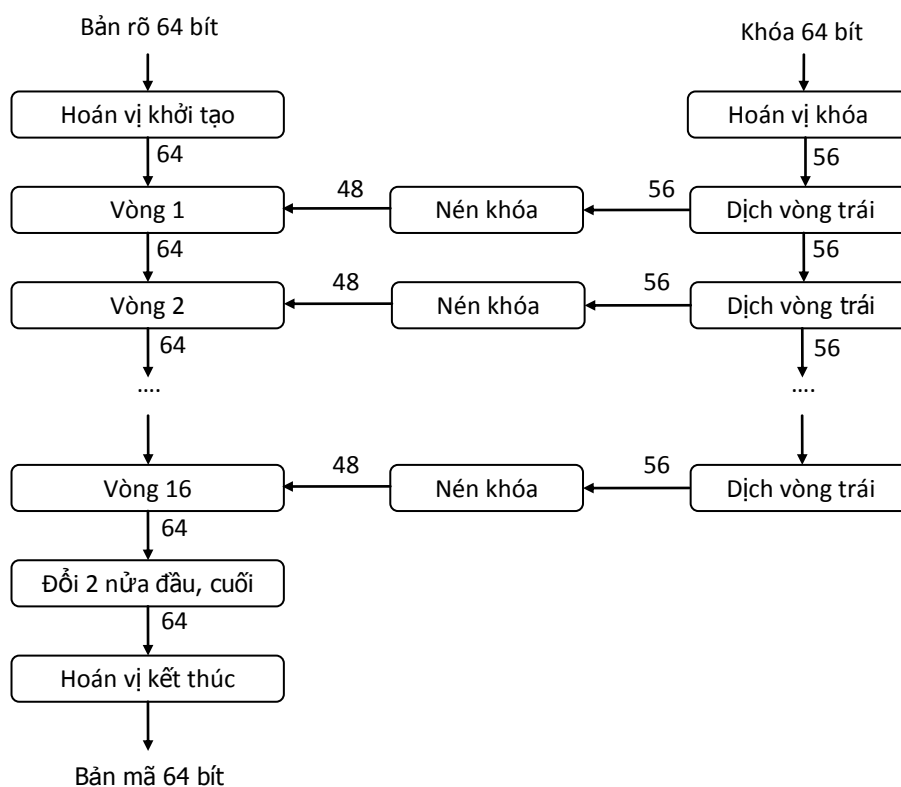
Tuy nhiên với mã TinyDES ba vòng, việc phá mã không còn đơn giản như vậy, người phá mã chỉ biết được input của vòng đầu là  $P$  và output của vòng cuối là  $C$ , giá trị trung gian  $L_1R_1, L_2R_2$  bị ẩn giấu nên không thể giới hạn miền tìm kiếm của các khóa  $K_1, K_2, K_3$  theo phương pháp trên. Dưới tác động của  $S\text{-box}$ , việc thay đổi 1 bit trong bản rõ hoặc khóa  $K$  sẽ ảnh hưởng đến nhiều bit khác nhau trong các giá trị trung gian  $L_1R_1, L_2R_2$  (trong phần mã DES ta sẽ gọi là hiệu ứng lan truyền), nên khó phân tích mối liên quan giữa bản rõ, bản mã và khóa. Việc phá mã còn khó khăn hơn nữa trong trường hợp mã DES gồm 16 vòng và kích thước khối là 64 bit.

### 3.4 Mã DES (Data Encryption Standard)

Mã DES có các tính chất sau:

- Là mã thuộc hệ mã Feistel gồm 16 vòng, ngoài ra DES có thêm một hoán vị khởi tạo trước khi vào vòng 1 và một hoán vị khởi tạo sau vòng 16
- Kích thước của khối là 64 bit: ví dụ bản tin '*meetmeafterthetogaparty*' biểu diễn theo mã ASCII thì mã DES sẽ mã hóa làm 3 lần, mỗi lần 8 chữ cái (64 bit): *meetmeaf - tertheto - gaparty*.
- Kích thước khóa là 56 bit
- Mỗi vòng của DES dùng khóa con có kích thước 48 bit được trích ra từ khóa chính.

Hình dưới đây minh họa các vòng của mã DES



**Hình 3-6. Các vòng Feistel của mã DES**

Sơ đồ mã DES trên gồm ba phần, phần thứ nhất là các hoán vị khởi tạo và hoán vị kết thúc. Phần thứ hai là các vòng Feistel, phần thứ ba là thuật toán sinh khóa con. Chúng ta sẽ lần lượt đi vào chi tiết của từng phần.

### 3.4.1 Hoán vị khởi tạo và hoán vị kết thúc:

Ta đánh số các bit của khối 64 bit theo thứ tự từ trái sang phải là 0, 1, ..., 62, 63:  $b_0b_1b_2 \dots b_{62}b_{63}$

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

$$(b_0b_1b_2 \dots b_{62}b_{63} \rightarrow b_{57}b_{49}b_{41} \dots b_{14}b_6)$$

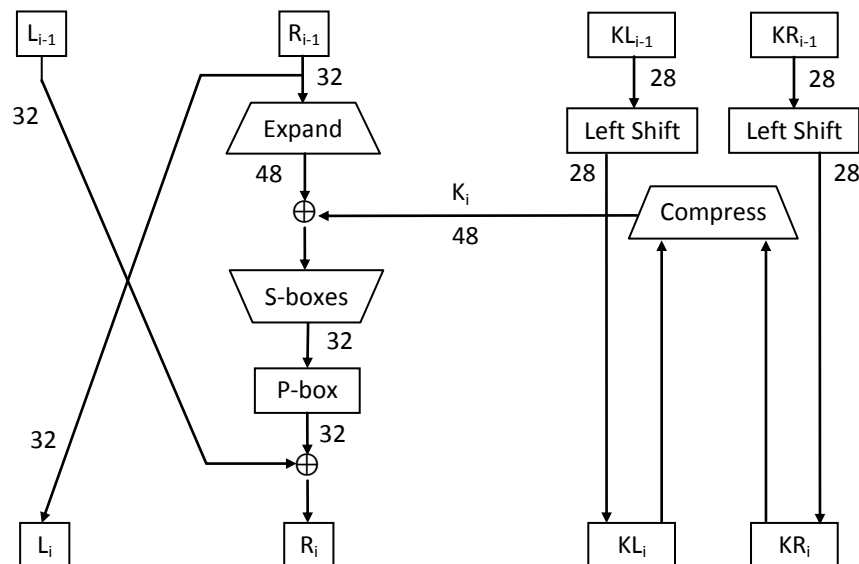
Hoán vị kết thúc hoán đổi các bit theo quy tắc sau:

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

Hoán vị kết thúc chính là hoán vị nghịch đảo của hoán vị khởi tạo. Đối với known-plaintext hay chosen-plaintext attack, hoán vị khởi tạo và hoán vị kết thúc không có ý nghĩa bảo mật, sự tồn tại của hai hoán vị trên được nhận định là do yếu tố lịch sử.

### 3.4.2 Các vòng của DES

Hình sau minh họa một vòng Feistel của DES



**Hình 3-7. Cấu trúc một vòng của mã DES**

Trong DES, hàm  $F$  của Feistel là:

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

Trong đó hàm  $Expand$  vừa mở rộng vừa hoán vị  $R_{i-1}$  từ 32 bit lên 48 bit. Hàm  $S\text{-boxes}$  nén 48 bit lại còn 32 bit. Hàm  $P\text{-box}$  là một hoán vị 32 bit. Mô tả của các hàm trên là như sau:

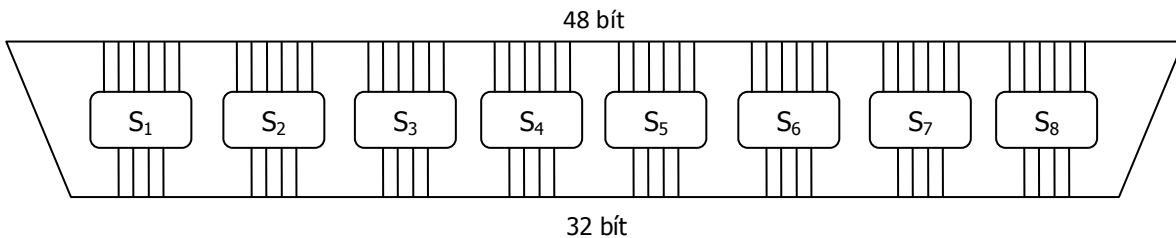
- $Expand$ : đánh số các bit của  $R_{i-1}$  theo thứ tự từ trái sang phải là 0, 1, 2, ..., 31. Hàm  $Expand$  thực hiện vừa hoán vị vừa mở rộng 32 bit thành 48 bit theo quy tắc:

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

- *S-boxes*:

Hàm *S-boxes* của DES biến đổi một số 48 bit thành một số 32 bit. Tuy nhiên, nếu chỉ lập một bảng tra cứu như ở TinyDES thì bảng này phải có  $2^{16}$  dòng và  $2^{32}$  cột, dẫn đến số phần tử của bảng rất lớn. Để giảm kích thước của bảng tra cứu, người ta chia hàm *S-boxes* thành 8 hàm *S-box* con, mỗi hàm biến đổi số 6 bit thành số 4 bit (hình dưới)



Hàm *S-box* đầu tiên, hộp  $S_1$ , giống hoàn toàn như *S-box* của TinyDES, tức có nội dung như sau:

		$b_1b_2b_3b_4$															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
$b_0b_5$	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

Chi tiết các hộp còn lại được trình bày trong Phụ lục 1. Có thể thấy, mỗi hàm *S-box* con là một phép thay thế Substitution. Các hàm *S-box* con không khả nghịch, do đó hàm *S-boxes* cũng không khả nghịch. Sự phức tạp này của *S-boxes* là yếu tố chính làm cho DES có độ an toàn cao.

- *P-box*: hàm *P-box* cũng 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

### 3.4.3 Thuật toán sinh khóa con của DES

Khóa  $K$  64 bit ban đầu được rút trích và hoán vị thành một khóa 56 bit (tức chỉ sử dụng 56 bit) theo quy tắc:

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

Khóa 56 bit này được chia thành 2 nửa trái phải  $KL_0$  và  $KR_0$ , mỗi nửa có kích thước 28 bit. Tại vòng thứ  $i$  ( $i = 1, 2, 3, \dots, 16$ ),  $KL_{i-1}$  và  $KR_{i-1}$  được dịch vòng trái  $r_i$  bit để có được  $KL_i$  và  $KR_i$ , với  $r_i$  được định nghĩa:

$$r_i = \begin{cases} 1 & \text{nếu } i \in \{1, 2, 9, 16\} \\ 2 & \text{với những } i \text{ khác} \end{cases}$$

Cuối cùng khóa  $K_i$  của mỗi vòng được tạo ra bằng cách hoán vị và nén 56 bit của  $KL_i$  và  $KR_i$  thành 48 bit theo quy tắc:

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

#### 3.4.4 Hiệu ứng lan truyền (Avalanche Effect)

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 bit trong bản rõ hay khóa thì dẫn đến sự thay đổi của nhiều bit bản mã. Tính chất này được gọi là hiệu ứng lan truyền. Nhờ có tính chất này mà người 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 vét cạn khóa.

DES là một phương pháp mã hóa có hiệu ứng lan truyền này. Xét hai bản rõ sau (64 bit):

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 2-1.a cho biết số bit khác nhau của bản mã tương ứng với  $P1$  và  $P2$  qua các vòng của DES:

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)

**Bảng 3-1. Hiệu ứng lan truyền**

Chỉ cần đến vòng thứ 2, số bit khác nhau giữa hai bản mã đã là 21 bit, sau 16 vòng số bit khác nhau là 34 bit (khoảng 1/2 tổng số bit của bản rõ)

Xét bản rõ sau (64 bit):

P: 01101000 10000101 00101111 01111010 00010011 01110110 11101011 10100100

Dùng hai khóa sau đây để mã hóa bản rõ trên (hai khóa này chỉ khác nhau 1 bit):

K1: 1110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

K2: 0110010 1111011 1101111 0011000 0011101 0000100 0110001 11011100

Bảng 2-1.b cho biết số bit khác nhau của bản mã tương ứng với K1 và K2 qua các vòng của DES. Sau 16 vòng, số bit khác nhau là 35 bit, cũng khoảng 1/2 tổng số bit của bản rõ.

### 3.4.5 Độ an toàn của DES

Ta hãy xem xét tính an toàn của DES trước một vài phương pháp tấn công phá mã.

1) Tấn công vét cạn khóa (Brute Force Attack):

Vì khóa của mã DES có chiều dài là 56 bit nên để tiến hành brute-force attack, cần kiểm tra  $2^{56}$  khóa khác nhau. Hiện nay với những thiết bị phổ dụng, thời gian gian để thử khóa là rất lớn nên việc phá mã là không khả thi (xem bảng). Tuy nhiên vào năm 1998, tổ chức Electronic Frontier Foundation (EFF) thông báo đã xây dựng được một thiết bị phá mã DES gồm nhiều máy tính chạy song song, trị giá khoảng 250.000\$. Thời gian thử khóa là 3 ngày. Hiện nay mã DES vẫn còn được sử dụng trong thương mại, tuy nhiên người ta đã bắt đầu áp dụng những phương pháp mã hóa khác có chiều dài khóa lớn hơn (128 bit hay 256 bit) như TripleDES hoặc AES.

2) Phá mã DES theo phương pháp vi sai (differential cryptanalysis):

Năm 1990 Biham và Shamir đã giới thiệu phương pháp phá mã vi sai. Phương pháp vi sai tìm khóa ít tốn thời gian hơn brute-force. Tuy nhiên phương pháp phá mã này lại đòi hỏi phải có  $2^{47}$  cặp bản rõ - bản mã được lựa chọn (chosen-plaintext). Vì vậy phương pháp này là bất khả thi dù rằng số lần thử có thể ít hơn phương pháp brute-force.

3) Phá mã DES theo phương pháp thử tuyến tính (linear cryptanalysis)

Năm 1997 Matsui đưa ra phương pháp phá mã tuyến tính. Trong phương pháp này, cần phải biết trước  $2^{43}$  cặp bản rõ-bản mã (known-plaintext). Tuy nhiên  $2^{43}$  cũng là một con số lớn nên phá mã tuyến tính cũng không phải là một phương pháp khả thi.

### 3.5 Một số phương pháp mã khối khác

#### 3.5.1 Triple DES

Một trong những cách để khắc phục yếu điểm kích thước khóa ngắn của mã hóa DES là sử dụng mã hóa DES nhiều lần với các khóa khác nhau cho cùng một bản tin. Đơn giản nhất là dùng DES hai lần với hai khóa khác nhau, cách thức này được gọi là Double DES:

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

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 gặp-nhau-ở-giữa (meet-in-the-middle). Đây là một phương pháp tấn công chosen-plaintext.

Vì vậy người ta chọn dùng DES ba lần với ba khóa khác nhau, cách thức này được gọi là Triple DES:

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

Chiều dài khóa là 168 bit sẽ gây phức tạp hơn nhiều cho việc phá mã bằng phương pháp tấn công gặp-nhau-ở-giữa. Trong thực tế người ta chỉ dùng Triple DES với hai khóa  $K_1, K_2$  mà vẫn đảm bảo độ an toàn cần thiết. Công thức như sau:

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

Nguyên nhân của việc dùng  $EDE$  thay cho  $EEE$  là nếu với  $K_1 = K_2 = K$  thì

$$C = E(D(E(P, K), K), K) = E(P, K)$$

Nghĩa là Triple DES suy giảm thành một DES đơn.

#### 3.5.2 Advanced Encryption Standard (AES)

Vào những năm 1990, nhận thấy nguy cơ của 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 mã hóa mới. Cuối cùng một thuật toán có tên là Rijndael được chọn và đổi tên thành Advanced Encryption Standard hay AES. Có thể nói rằng mã hóa AES với khóa có kích thước 256 bit là “an toàn mãi mãi” bất kể những tiến bộ trong ngành kỹ thuật máy tính.

Giống như DES, mã hóa AES là một mã khối gồm nhiều vòng. Khác với DES, mã hóa AES không phải là một mã hóa Feistel. Thuật toán AES khá phức tạp, ở đây chúng ta chỉ nêu ra một số đặc điểm chính của AES:

- Cho phép lựa chọn kích thước khối mã hóa là 128, 192 hay 256 bit.
- Cho phép lựa chọn kích thước của khóa một cách độc lập với kích thước khối: là 128, 192 hay 256 bit.
- Số lượng vòng có thể thay đổi từ 10 đến 14 vòng tùy thuộc vào kích thước khóa.

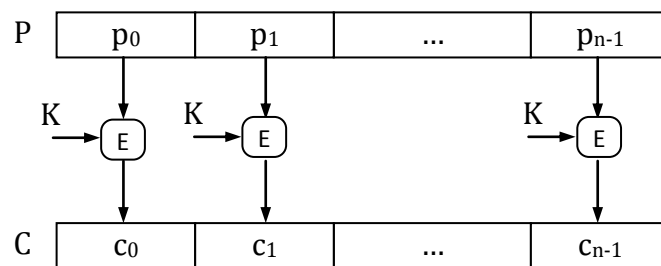
Độ an toàn của AES làm cho AES được sử dụng ngày càng nhiều và trong tương lai sẽ chiếm vai trò của DES và Triple DES.

### 3.6 Các mô hình ứng dụng mã khối

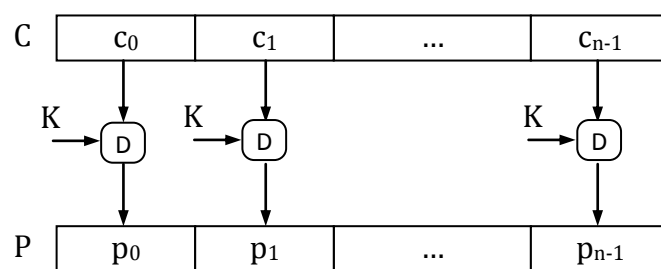
Mã khối (như mã DES) được áp dụng để mã hóa một khối dữ liệu có kích thước xác định. Để mã hóa một bản tin dài, bản tin được chia ra thành nhiều khối ( $P = P_0P_1P_2 \dots P_{n-1}$ ) và áp dụng mã khối cho từng khối một. Có nhiều mô hình áp dụng mã khối là ECB, CBC, CTR, OFB và CFB.

#### 3.6.1 Electronic Codebook – ECB

Trong mô hình ECB, mỗi khối được mã hóa một cách riêng rẽ, dùng chung một khóa  $K$ .



a) Quá trình mã hóa



b) Quá trình giải mã

**Hình 3-8. Mô hình ECB của mã khối**

Trong mã hóa ECB, nếu  $P_i = P_j$  thì  $C_i = C_j$  và ngược lại. Có thể thấy rằng mã ECB tương tự như mã hóa đơn bảng cổ điển, trong đó  $P_i$  và  $C_i$  giống như là các chữ cái, còn khóa  $K$  cùng với mã khối giống như là một phép hoán vị. Do đó, người phá mã có thể dựa vào một số đặc tính thống kê của dữ liệu để tiến hành phá mã, giống như dùng thống kê tần suất chữ cái để phá mã mã hóa đơn bảng (dù rằng  $P_i$  có kích thước lớn nên đặc tính thống



kê cũng khó phát hiện hơn). Vì vậy mã hóa ECB chỉ thích hợp để mã hóa những bản tin ngắn.



**Hình 3-9. Mã hóa ECB không che dấu hết thông tin (nguồn: trích từ [3])**

Để minh họa đặc tính thống kê của dữ liệu, hình trên thể hiện một tấm ảnh được mã hóa bằng ECB. Dù rằng mỗi khối đã được biến đổi qua phép mã hóa, tuy nhiên nhìn tổng thể thì vẫn tồn tại một sự liên hệ nào đó giữa các khối.

### 3.6.2 Cipher Block Chaining – CBC

Trong mô hình CBC, bản mã của một lần mã hóa được sử dụng cho lần mã hóa tiếp theo:

$$C_i = E(P_i \oplus C_{i-1}, K) \quad \text{với } i = 1, 2, 3 \dots n-1$$

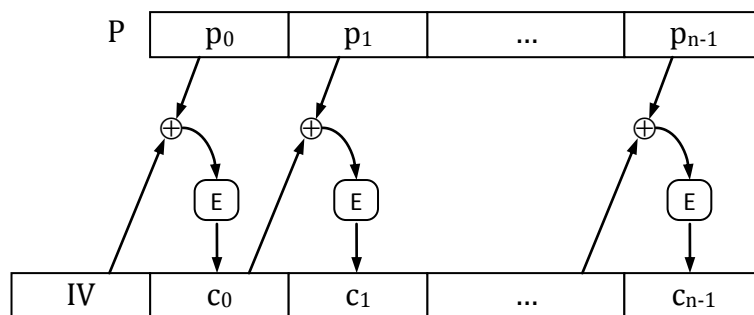
Do đó để mã hóa khối đầu tiên, người ta dùng một khối dữ liệu giả được gọi là vector khởi tạo (initialization vector – IV) và được chọn ngẫu nhiên:

$$C_0 = E(P_0 \oplus IV, K)$$

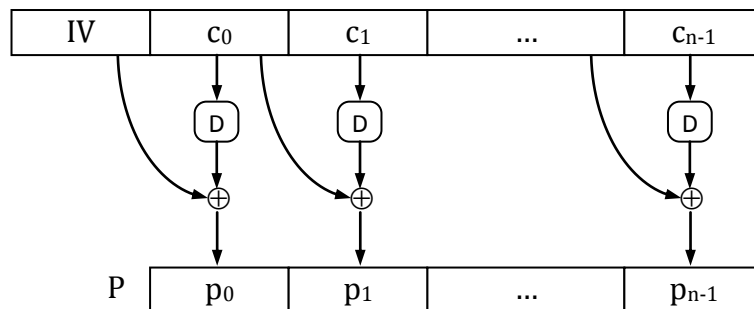
Để giải mã, tiến hành ngược lại:

$$P_0 = D(C_0, K) \oplus IV$$

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



a) Quá trình mã hóa



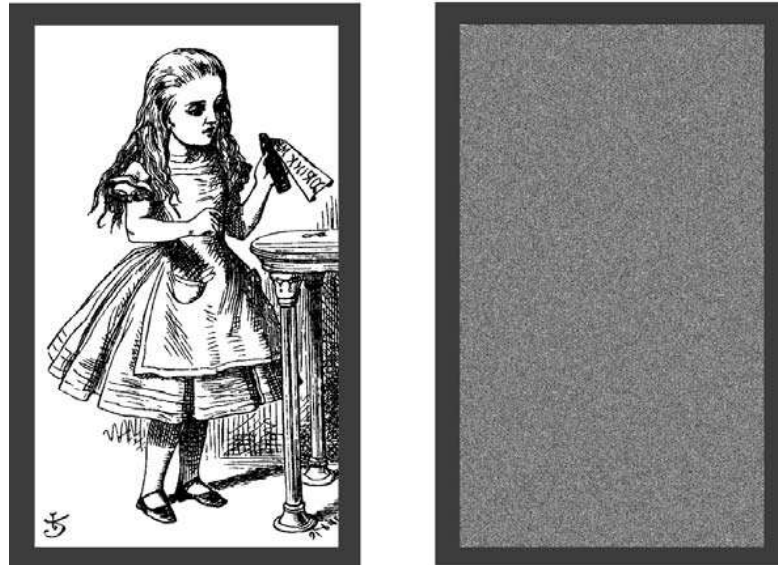
b) Quá trình giải mã

**Hình 3-10. Mô hình CBC của mã khối**

Người mã hóa và người giải 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 ( $IVC_0C_1C_2 \dots C_{n-1}$ ).

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 người phá mã không thể phát hiện ra những đặc tính thống kê của dữ liệu.

Ngược lại, đối với việc giải 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 bit bị hỏng thì dẫn đến không thể giải mã được bản mã đó và bản mã tiếp theo sau.

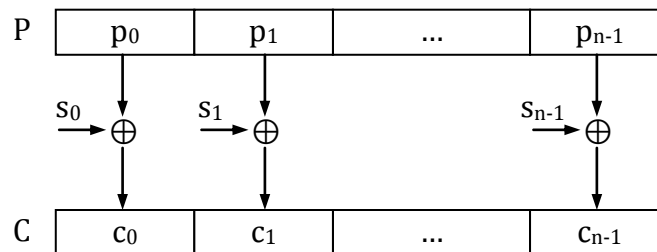


Hình 3-11. Bức ảnh sau khi mã hóa dùng mô hình CBC (nguồn: trích từ [3])

### 3.6.3 Counter – CTR

Đến thời điểm này, chúng ta đã tìm hiểu hai cách tiếp cận để chống lại việc phá mã dựa trên thống kê tần suất. Cách tiếp cận thứ nhất là theo mô hình của mã dòng, dùng một bộ sinh khóa ngẫu nhiên (confusion – làm rối). Cách tiếp cận thứ hai là theo mô hình CBC của mã khối, dùng các khối phía trước tác động đến bản mã của các khối đi sau (diffusion – khuếch tán).

Xét lại mô hình mã dòng:



Trong đó  $s_0, s_1, s_2, \dots$  được sinh ra từ bộ sinh số ngẫu nhiên.

CTR thực ra là một phương pháp mã hóa thuộc loại mã dòng, tuy nhiên bộ sinh khóa ngẫu nhiên có dùng đến mã khối để sinh số. Kích thước của đơn vị mã hóa là kích thước mã khối (ví dụ nếu dùng mã DES thì đơn vị mã hóa là 64 bit). Với một vector khởi tạo ban đầu, công thức sinh số như sau:

$$s_i = E(IV + i, K)$$

Do hiệu ứng lan truyền (Avalanche Effect) của mã khối nên có thể xem bộ sinh khóa trên sinh ra một dãy số ngẫu nhiên theo nguyên tắc thiết kế của mã dòng.

### 3.6.4 Output Feedback – OFB

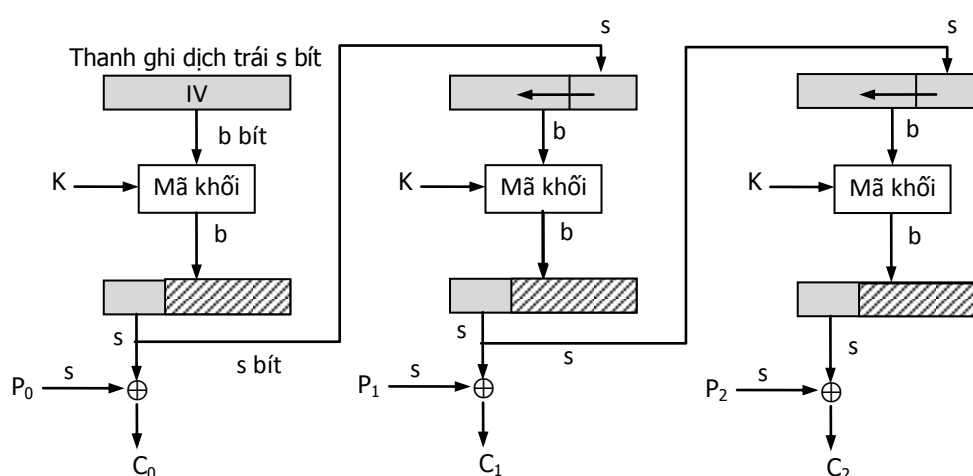
Mô hình CTR là một mã dòng trong đó đơn vị mã hóa có kích thước cố định là  $b$  bit, với  $b$  là kích thước mã khối. Để mã hóa với đơn vị mã hóa có kích thước bất kỳ, mô hình OFB được đề xuất. Mô hình này có hai điểm khác so với mô hình CTR:

- Chỉ dùng  $s$  bit đầu tiên của khóa sinh ra bởi bộ sinh khóa, với  $s$  là kích thước đơn vị mã hóa dùng trong phép XOR.
- Để tăng thêm tính ngẫu nhiên của bộ sinh khóa,  $s$  bit này của khóa được ghép vào vector khởi tạo IV cho lần mã hóa tiếp theo. Phép ghép được thực hiện bằng cách đẩy trái IV  $s$  bit và đưa  $s$  bit của khóa vào  $s$  bit thấp của IV.

```

Register = IV;
for i = 0 to n-1 do
    Ti = E(Register, K);
    Ti = Ti SHR (b-s);           // lấy s bit đầu của Ti
    Ci = Pi XOR Ti;
    Register = Register SHL s;
    Register = Register OR Ti;
next i

```

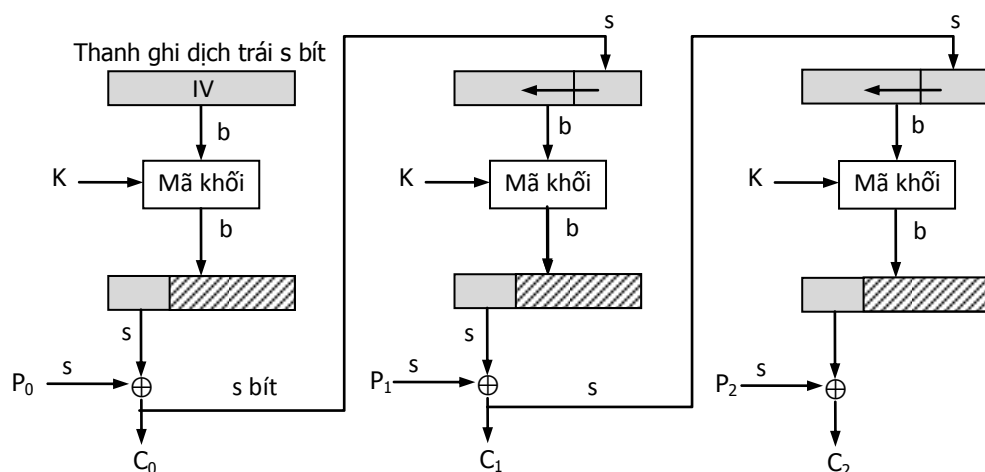


**Hình 3-12. Mô hình OFB của mã khối**

Giống như mô hình ECB, trong mô hình CTR và OFB, mỗi khối được mã hóa một cách riêng rẽ, không phụ thuộc vào nhau.

### 3.6.5 Cipher Feedback – CFB

Mô hình CFB có thay đổi một chút so với mô hình OFB. Mô hình OFB dùng  $s$  bit của khóa do bộ sinh khóa tạo ra để ghép với IV cho lần mã hóa tiếp theo. Còn mô hình CFB dùng  $s$  bit của bản mã để ghép với IV.



**Hình 3-13. Mô hình CFB của mã khối**

Do đó giống như mô hình CBC, 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. Ngược lại, đối với việc giải 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.

### 3.7 Tính chứng thực (authentication) của mã hóa đối xứng.

Trong phần lớn chương này, chúng ta đã tìm hiểu về cách thức mã hóa đối xứng thực hiện tính bảo mật. Vậy còn tính chứng thực thì sao? Mã hóa đối xứng có thể chống lại các hình thức tấn công sửa đổi thông điệp, mạo danh và phát lại thông điệp được hay không? Câu trả lời là có.

Trước tiên là vấn đề mạo danh. Trudy có thể gửi thông điệp cho Bob mà Bob nghĩ rằng thông điệp đó là từ Alice không? Xét tình huống sau:

- Alice và Bob quyết định dùng mã Vigenere để trao đổi dữ liệu, với khóa bí mật  $K_{AB}$  là 'DECEPTIVE'
- Khi Alice gửi cho Bob một bản mã  $C$ , Bob dùng  $K_{AB}$  để giải mã cho ra bản rõ. Ví dụ, Alice gửi bản mã: 'ZICVTWQNGRZGVTWAVZHCQYGLMGJ'. Bob giải mã có được bản rõ: 'wearediscoveredsaveyourself'. Đây là một bản tin tiếng Anh có ý nghĩa.
- Trudy muốn mạo danh Alice nên tìm một bản mã  $C_T$  và gửi  $C_T$  cho Bob. Bob nghĩ rằng  $C_T$  là từ Alice nên giải mã bằng  $K_{AB}$  và có được bản rõ  $P_T$ . Vấn đề ở đây là làm sao Bob biết được  $P_T$  là của Trudy chứ không phải của Alice?
- Vì Trudy không biết  $K_{AB}$  nên Trudy không thể chọn  $P_T$  trước rồi mới có  $C_T$ . Do đó Trudy phải chọn ngẫu nhiên một  $C_T$  nào đó. Ví dụ Trudy chọn  $C_T$  là 'WDTAXRLKY'. Như vậy Bob giải mã có được  $P_T$  là 'tzrwiypdu'. Tuy nhiên  $P_T$  này không phải là văn bản có nghĩa trong tiếng Anh.
- Có thể thấy rằng việc Trudy chọn được một  $C_T$  nào đó, sao cho sau khi Bob giải mã cho ra  $P_T$  là văn bản có nghĩa, thì có xác suất rất bé. Trong trường hợp  $P_T$  có nghĩa theo ý muốn của Trudy thì coi như là không thể xảy ra.

- Do đó có thể chắc chắn rằng nếu Trudy mạo danh thì  $P_T$  sẽ là văn bản vô nghĩa, từ đó Bob biết được  $C_T$  là không phải từ Alice.

Tương tự như vậy với vấn đề sửa nội dung thông điệp, nếu Trudy chặn được bản mã  $C$  của Alice và sửa  $C$  thành  $C_T$ , thì xác suất để  $P_T$  là văn bản có nghĩa cũng rất bé. Và Bob biết được  $C$  đã bị sửa đổi.

Đối với mã hóa hiện đại cũng vậy, nếu Trudy chọn  $C_T$  là một dãy bit bất kỳ thì bản rõ  $P_T$  cũng là một dãy bit lộn xộn, không có cấu trúc ý nghĩa.

Tuy nhiên, trong thực tế, việc xác định như thế nào là dãy bit vô nghĩa là một công việc khó khăn đối với máy tính. Ngoài ra, có những loại dữ liệu hoàn toàn là một dãy bit ngẫu nhiên. Trong thực tế, chúng ta phải chấp nhận rằng bất cứ dãy bit  $P$  nào cũng có thể là có ý nghĩa. Do đó, để đảm bảo tính chứng thực, người ta dùng khái niệm mã chứng thực thông điệp – MAC để biến dãy bit ngẫu nhiên thành dãy bit có cấu trúc. Chúng ta sẽ tìm hiểu về MAC trong chương 5. Còn tại thời điểm này, chúng ta chấp nhận rằng một thông điệp có ý nghĩa thì là một dãy bit có cấu trúc.

Đối với tấn công phát lại thông điệp (replay attack). Alice gửi bản mã  $C$  cho Bob, Bob nhận được và giải mã để có bản rõ  $P$ . Tuy nhiên Trudy chặn được bản mã  $C$  và sau đó mạo danh Alice gửi  $C$  cho Bob thêm một lần nữa. Bob giải mã và cũng có được  $P$ . Như vậy Bob nhận được cùng một thông điệp  $P$  hai lần. Tại lần thứ 2, Bob không có cơ sở xác định là Alice muốn gửi lại hay là do Trudy gửi. Chương 6 sẽ trình bày các phương pháp chống lại hình thức tấn công phát lại thông điệp.

### 3.8 Tính không từ chối (non-repudiation) của mã hóa đối xứng.

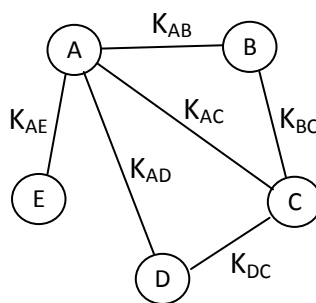
Dù rằng mã hóa đối xứng đảm bảo tính bảo mật của hệ truyền tin, tuy nhiên mã hóa đối xứng lại không thực hiện được tính không từ chối. Nguyên nhân ở đây là tính bí mật của khóa. Vì khóa  $K$  bí mật có hai người biết, nên nếu  $K$  bị tiết lộ thì không có cơ sở để quy trách nhiệm cho Alice hay Bob làm lộ khóa. Do đó Alice có thể từ chối là đã gửi bản tin.

Lấy lại ví dụ về chứng khoán, giả sử Bob là nhân viên môi giới chứng khoán của Alice. Alice gửi thông điệp yêu cầu Bob mua cổ phiếu của công ty Z. Thông điệp này được mã hóa. Ngày hôm sau, giá cổ phiếu công ty này giảm hơn 50%. Thấy bị thiệt hại, Alice nói rằng Bob đã làm lộ khóa, Trudy có được khóa và gửi thông điệp chứ không phải là Alice. Bob không thể nào bác bỏ lập luận này.

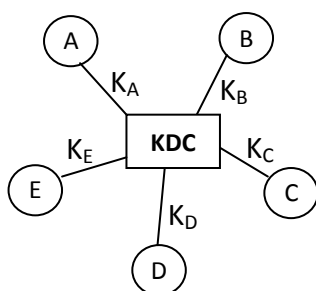
Vì vậy các nhà nghiên cứu bắt đầu tìm kiếm các phương án mã hóa khác, sao cho khóa bí mật chỉ có một người biết mà thôi. Đó là phương pháp mã hóa khóa công khai, được trình bày trong chương tiếp theo.

### 3.9 Trao đổi khóa bí mật bằng trung tâm phân phối khóa

Giả sử có  $N$  người sử dụng, trao đổi dữ liệu bằng mã hóa đối xứng, mỗi cặp người sử dụng cần có một khóa bí mật riêng, dẫn đến cần có  $N(N-1)/2$  khóa bí mật. Việc thiết lập các khóa bí mật này sẽ gây ra khó khăn cho các người sử dụng vì mỗi người cần thiết lập  $N-1$  khóa.

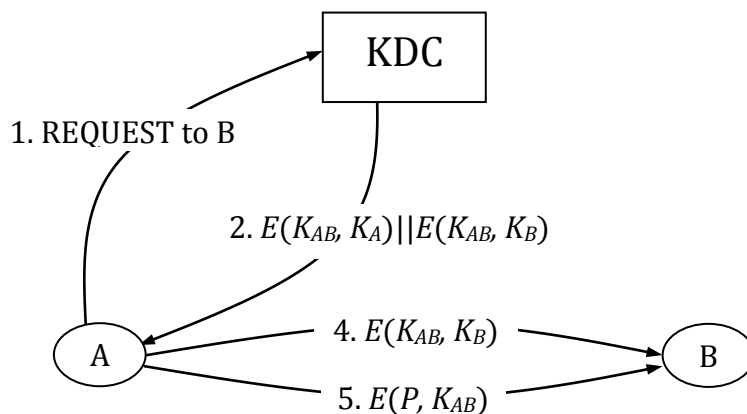


Phương pháp trao đổi khóa bằng trung tâm phân phối khóa (Key Distribution Center – KDC) giúp đơn giản hóa vấn đề này. Trong mô hình sử dụng KDC, mỗi người sử dụng chỉ cần có một khóa bí mật với KDC. Còn khóa dùng để trao đổi dữ liệu giữa các người sử dụng sẽ do KDC cung cấp.



Giả sử Alice có khóa bí mật  $K_A$  với KDC và Bob có khóa bí mật  $K_B$  với KDC. Bây giờ Alice muốn trao đổi dữ liệu với Bob. Quá trình thiết lập khóa chung  $K_{AB}$  giữa Alice và Bob gồm các bước:

- 1) Alice gửi yêu cầu muốn trao đổi dữ liệu với Bob cho KDC.
- 2) KDC tạo một khóa bí mật  $K_{AB}$  và mã hóa thành hai bản mã. Một bản mã được mã hóa bằng khóa bí mật của Alice  $E(K_{AB}, K_A)$  và một bản mã được mã hóa bằng khóa bí mật của Bob  $E(K_{AB}, K_B)$ .
- 3) Alice giải mã  $E(K_{AB}, K_A)$  để có  $K_{AB}$
- 4) Alice gửi  $E(K_{AB}, K_B)$  cho Bob, Bob giải mã để có được  $K_{AB}$
- 5) Alice và Bob trao đổi dữ liệu qua khóa bí mật  $K_{AB}$



**Hình 3-14. Trao đổi khóa bí mật dùng KDC**

Như vậy, khóa  $K_{AB}$  chỉ có KDC, Alice và Bob biết. Trách nhiệm của KDC là giữ bí mật khóa này. Alice và Bob dùng khóa  $K_{AB}$  để mã hóa dữ liệu. Khi kết thúc quá trình

truyền dữ liệu,  $K_{AB}$  được hủy bỏ. Lần sau nếu Alice lại truyền số liệu với Bob thì KDC sẽ cung cấp khóa  $K_{AB}$  khác. Như vậy chỉ cần Alice có thiết lập khóa bí mật  $K_A$  với KDC thì Alice có thể truyền số liệu không chỉ với Bob mà còn với những người khác.

Một khái niệm quan trọng khác có thể rút ra từ mô hình dùng KDC là khái niệm khóa chủ và khóa phiên (master key và session key). Trong ví dụ trên các khóa  $K_A$ ,  $K_B$  không được sử dụng trực tiếp để mã hóa dữ liệu, chúng chỉ được dùng để mã hóa các khóa tạm  $K_{AB}$ . Các khóa  $K_{AB}$  này mới trực tiếp mã hóa dữ liệu và bị hủy bỏ khi sau quá trình truyền dữ liệu kết thúc. Vì vậy  $K_A$ ,  $K_B$  được gọi là khóa chủ, chúng ít được sử dụng nên người phá mã khó có cơ hội thu thập bản mã để phá mã. Khóa  $K_A$ ,  $K_B$  được sử dụng lâu dài. Còn  $K_{AB}$  được gọi là khóa phiên,  $K_{AB}$  chỉ tồn tại trong một phiên truyền dữ liệu duy nhất mà thôi.

Chương 7 trình bày giao thức Keberos, là một giao thức dựa trên khái niệm trung tâm phân phối khóa. Keberos được sử dụng trong các hệ điều hành ngày nay, để mã hóa dữ liệu trong mạng cục bộ LAN.

### 3.10 Câu hỏi ôn tập

- 1) Mã hóa đối xứng hiện đại và mã hóa đối xứng cổ điển khác nhau ở điểm nào.
- 2) Mã dòng hoạt động dựa trên nguyên tắc thay thế hay hoán vị?
- 3) Từ nguyên tắc sinh số của mã hóa A5/1 và RC4, hãy cho biết lý do mã dòng lại dùng bộ sinh số để sinh ra dãy bit? Tại sao không dùng trực tiếp khóa K để thực hiện phép XOR ?
- 4) Hệ mã Feistel có thuận lợi gì trong việc thực hiện mã khối?
- 5) Tại sao mã hóa DES lại dùng các phép biến đổi phức tạp chỉ để mã hóa một khối 64 bit?
- 6) Xét mô hình ECB, để mã hóa một bản tin dài bằng mã DES, chúng ta phải lần lượt mã hóa từng khối 64 bit. Việc thực hiện như vậy giống và khác với mã dòng ở những điểm nào?
- 7) Mô hình CBC có đặc tính gì mà các phương pháp mã hóa theo nguyên tắc thay thế (như ECB) không có?
- 8) Tại sao nói mô hình CTR, OFB và CFB thực ra là mã dòng?
- 9) Một bản rõ phải có đặc điểm gì thì mới có thể nói phương pháp mã hóa đối xứng có tính chứng thực? Nếu Trudy không biết khóa bí mật của Alice và Bob, Trudy có thể mạo danh Alice gửi thông điệp mà Trudy muốn cho Bob được không?
- 10) Trong mã hóa đối xứng, việc hai người cùng biết khóa dẫn đến nhược điểm gì của phương pháp mã hóa này?
- 11) Hãy nêu lợi ích của việc dùng khóa chủ và khóa phiên.

### 3.11 Bài tập

1. Xét thuật toán TinyA5/1, giả sử ban đầu  $X=21$ ,  $Y = 55$ ,  $Z=60$ . Tính bit thứ 1, 2, 3 được sinh ra bởi bộ sinh khóa.
2. Trong bước khởi tạo của thuật toán RC4, đầu tiên S là dãy các giá trị tăng dần từ 1 đến 255. Tìm khóa K để sau khi hoàn tất khởi tạo, S không đổi (vẫn là dãy tăng dần từ 1 đến 255).



3. Alice và Bob trao đổi dữ liệu bằng thuật toán A5/1, tuy nhiên họ muốn tránh việc dùng một khóa mới cho mỗi lần truyền dữ liệu. Alice và Bob bí mật chia sẻ một khóa  $k$  ban đầu gồm 128 bit. Để mã hóa thông điệp  $m$ , Alice tiến hành như sau:
  - Chọn một giá trị  $v$  bất kỳ gồm 80 bit.
  - Mã hóa bằng RC4:  $C = A51(v||k) \oplus m$
  - Gửi đi dãy bit  $v||C$
  - a. Mô tả các bước thực hiện của Bob để giải mã thông điệp
  - b. Giả sử Trudy quan sát thấy dãy  $(v_1||C_1), (v_2||C_2), (v_3||C_3), \dots$  gửi đi giữa Alice và Bob, nêu giải pháp để Trudy có thể phá mã.
4. Chứng minh rằng sau một số bước thực hiện, khóa sinh ra bởi thuật toán A5/1 sẽ lặp lại.
5. Chứng minh rằng sau một số bước thực hiện, khóa sinh ra bởi thuật toán RC4 sẽ lặp lại.
6. Xét một mã khối thuộc hệ Feistel gồm 4 vòng và  $P = (L_0, R_0)$ . Cho biết bảng mã  $C$  ứng với các trường hợp sau của hàm  $F$ :
  - a.  $F(R_{i-1}, K_i) = 0$ .
  - b.  $F(R_{i-1}, K_i) = R_{i-1}$ .
  - c.  $F(R_{i-1}, K_i) = K_i$
  - d.  $F(R_{i-1}, K_i) = R_{i-1} \oplus K_i$ .
7. Xét một mã khối thuộc hệ Feistel gồm 2 vòng với kích thước khối và kích thước khóa là 128 bit. Thuật toán sinh khóa con sinh ra khóa cho 2 vòng là như nhau  $k_1 = k_2$ .  
 Giả sử chúng ta được lựa chọn một (và chỉ một) bản rõ và có bản mã tương ứng (chosen-plaintext attack). Hãy nêu phương thức để phá mã một bản mã  $C$  nào đó.
8. Xét mã TinyDES trong đó khóa  $K$  là 10100100. Hãy tính bản mã trong trường hợp bản rõ là  $P = 01001011$
9. Công thức mã hóa cho mô hình ứng dụng mã khối CTR là:
 
$$C_i = P_i \oplus E(IV + i, K)$$
 Giả sử thay vì sử dụng công thức trên ta dùng công thức:
 
$$C_i = P_i \oplus E(K, IV + i)$$
 Thực hiện như vậy có an toàn không? Tại sao?
10. Xét một mô hình ứng dụng mã khối sau:
 
$$C_0 = IV \oplus E(P_0, K), C_1 = C_0 \oplus E(P_1, K), C_2 = C_1 \oplus E(P_2, K), \dots$$
 Hãy cho biết công thức giải mã. Trình bày một điểm yếu của mô hình này so với mô hình CBC.
11. Trong mô hình CBC, nếu Alice dùng một  $IV$  duy nhất cho tất cả các lần truyền dữ liệu thì có an toàn không? (các lần truyền đều dùng cùng khóa)
12. Trong mô hình CBC áp dụng mã khối 64 bit, nếu có một bit của bản mã bị hỏng trong quá trình truyền dữ liệu, tính số bit bị hỏng của bản giải mã.

### 3.12 Bài tập thực hành

1. Viết chương trình mã hóa và giải mã file bằng thuật toán A5/1, khóa là X, Y, Z nhập từ bàn phím.
2. Viết chương trình mã hóa và giải mã file bằng thuật toán RC4, khóa là dãy N byte nhập từ bàn phím.
3. Viết chương trình mã hóa và giải mã file bằng thuật toán DES và mô hình mã khối CBC. Khóa K được lưu trong 1 file text riêng dưới dạng chữ số thập lục phân.
4. Tìm hiểu về thư viện mã hóa của môi trường lập trình .NET (namespace System.Security.Cryptography). Viết chương trình mã hóa và giải mã một file dùng thuật toán DES, TripleDES, Rijndael và AES trong thư viện mã hóa của .NET. Khóa K được lưu trong 1 file text riêng dưới dạng chữ số thập lục phân.

Mã DES

Chọn thuật toán mã hóa: AES Thông tin khóa: 256 bit Thông tin block: 128 bit

Khóa (dạng hexa) 29864d678d930fd21080091fc67b0df7b7f685f2abc2bd366e8cf6efbedc68bf

IV (dạng hexa) ed233a0163fbff1918623557f972a8fc

Nạp Khóa&IV Lưu Khóa&IV Sinh Khóa&IV

File ban đầu D:\Books\Cac chuyen de lap trinh.pdf Chọn file...

File mã hóa D:\Books\Cac chuyen de lap trinh.pdf.aes Chọn file...

Mã hóa Giải mã

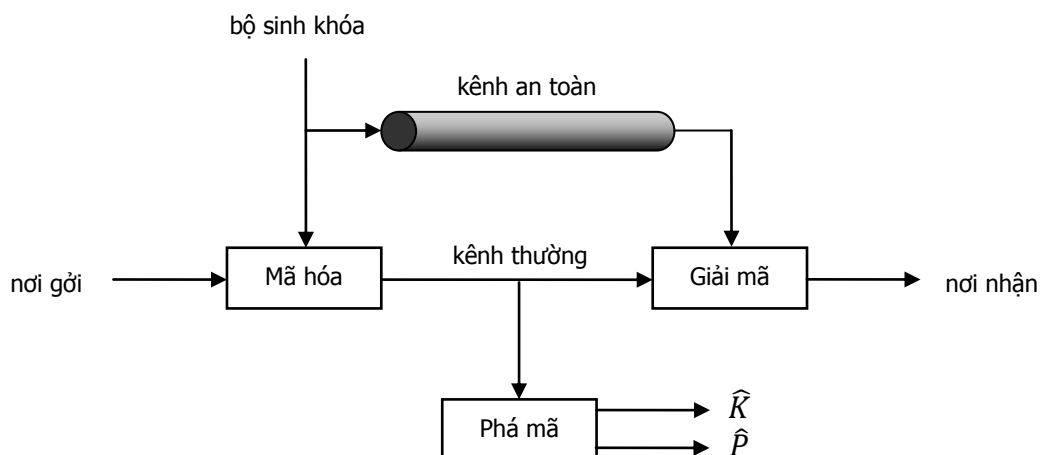
## CHƯƠNG 4. MÃ HÓA KHÓA CÔNG KHAI

Mã hóa đối xứng dù rằng đã phát triển từ cổ điển đến hiện đại, vẫn tồn tại hai điểm yếu sau:

- Vấn đề trao đổi khóa giữa người gửi và người nhận: Cần phải có một kênh an toàn để trao đổi khóa sao cho khóa phải được giữ bí mật chỉ có người gửi và người nhận biết. Điều này tỏ ra không hợp lý khi mà ngày nay, khối lượng thông tin luân chuyển trên khắp thế giới là rất lớn. Việc thiết lập một kênh an toàn như vậy sẽ tốn kém về mặt chi phí và chậm trễ về mặt thời gian.
- Tính bí mật của khóa: không có cơ sở quy trách nhiệm nếu khóa bị tiết lộ.

Vào năm 1976 Whitfield Diffie và Martin Hellman đã tìm ra một phương pháp mã hóa khác mà có thể giải quyết được hai vấn đề trên, đó là mã hóa khóa công khai (public key cryptography) hay còn gọi là mã hóa bất đối xứng (asymmetric cryptography). Đây có thể xem là một bước đột phá quan trọng nhất trong lĩnh vực mã hóa.

Xét lại mô hình mã hóa đối xứng:



Để khắc phục điểm yếu của mã hóa đối xứng người ta tập trung vào nghiên cứu theo hướng: có phương pháp nào để việc *mã hóa và giải mã dùng hai khóa khác nhau*? Có nghĩa là  $C = E(P, K_1)$  và  $P = D(C, K_2)$ . Nếu thực hiện được như vậy thì chúng ta sẽ có 2 phương án áp dụng:

Phương án 1: người nhận (Bob) giữ bí mật khóa  $K_2$ , còn khóa  $K_1$  thì công khai cho tất cả mọi người biết. Alice muốn gửi dữ liệu cho Bob thì dùng khóa  $K_1$  để mã hóa. Bob dùng  $K_2$  để giải mã. Ở đây Trudy cũng biết khóa  $K_1$ , tuy nhiên không thể dùng chính  $K_1$  để giải mã mà phải dùng  $K_2$ . Do đó chỉ có duy nhất Bob mới có thể giải mã được. Điều này bảo đảm *tính bảo mật* của quá trình truyền dữ liệu. Ưu điểm của phương án này là không cần phải truyền khóa  $K_1$  trên kênh an toàn.

$$\cancel{P = D(C, K_1)}$$

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

Phương án 2: người gửi (Alice) giữ bí mật khóa  $K_1$ , còn khóa  $K_2$  thì công khai cho tất cả mọi người biết. Alice muốn gửi dữ liệu cho Bob thì dùng khóa  $K_1$  để mã hóa. Bob

dùng  $K_2$  để giải mã. Ở đây Trudy cũng biết khóa  $K_2$  nên Trudy cũng có thể giải mã được. Do đó phương án này *không đảm bảo tính bảo mật*. Tuy nhiên lại có tính chất quan trọng là *đảm bảo tính chứng thực và tính không từ chối*. Vì chỉ có duy nhất Alice biết được khóa  $K_1$ , nên nếu Bob dùng  $K_2$  để giải mã ra bản tin, thì điều đó có nghĩa là Alice là người gửi bản mã. Nếu Trudy cũng có khóa  $K_1$  để gửi bản mã thì Alice sẽ bị quy trách nhiệm làm lộ khóa  $K_1$ . Trong phương án này cũng không cần phải truyền  $K_2$  trên kênh an toàn.

Vì vậy nếu kết hợp phương án 1 và phương án 2, thì mô hình đề xuất của chúng ta khắc phục được các nhược điểm của mã hóa đối xứng.

Trong cả hai phương án, một khóa được giữ bí mật chỉ một người biết, còn khóa kia được công khai. Do đó mô hình mã hóa trên được gọi là mã hóa khóa công khai (hay mã hóa bất đối xứng). Để thuận tiện ta quy ước lại các ký hiệu như sau:

- Để tránh nhầm lẫn với khóa bí mật của mã đối xứng, khóa bí mật trong mô hình trên được gọi là khóa riêng (private key) và ký hiệu là  $K_R$ .
- Khóa công khai (public key) được ký hiệu là  $K_U$ .
- Bản rõ được ký hiệu là  $M$ , còn bản mã giữ nguyên ký hiệu là  $C$
- Phương án 1 viết lại thành:

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

$$M = D(C, K_R)$$

- Phương án 2 viết lại thành:

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

$$M = D(C, K_U)$$

Vấn đề còn lại ở đây là liệu có tồn tại một mô hình mã hóa và giải mã dùng hai khóa khác nhau như vậy không? Dĩ nhiên là hai khóa  $K_U$  và  $K_R$  không thể nào hoàn toàn độc lập với nhau. Phải có một mối quan hệ nào đó giữa  $K_U$  và  $K_R$  thì mới có thể tiến hành giải mã hóa và giải mã được. Có nghĩa là  $K_R = f(K_U)$ . Tuy nhiên một yêu cầu rất quan trọng là việc tính  $K_R = f(K_U)$  phải là bất khả thi về mặt thời gian. Nếu nguyên tắc này bị vi phạm thì việc giữ bí mật khóa  $K_R$  không còn ý nghĩa vì từ khóa công khai  $K_U$  có thể tính được  $K_R$ .

Để có được cặp khóa  $K_R$  và  $K_U$  như trên, người ta thường dùng các *hàm một chiều* (oneway function). Các hàm một chiều có tính chất là hàm nghịch đảo của chúng rất khó thực hiện. Sau đây là ví dụ về hàm một chiều: việc sinh ra hai số nguyên tố lớn  $p, q$  và tính tích  $N = pq$  thì thực hiện dễ dàng. Tuy nhiên nếu chỉ cho trước  $N$  và thực hiện phân tích  $N$  để tìm lại hai số nguyên tố  $p, q$  là việc hoàn toàn bất khả thi về mặt thời gian. Chúng ta sẽ xem cách thức áp dụng hàm một chiều này để tạo khóa  $K_R$  và  $K_U$  trong phần mã hóa RSA.

Có nhiều phương pháp mã hóa thuộc loại mã hóa khóa công khai. Đó là các phương pháp Knapsack, RSA, Elgaman, và phương pháp đường cong elliptic ECC.... Mỗi phương pháp có cách thức ứng dụng hàm một chiều khác nhau. Trong tài liệu này, chúng ta chỉ tập trung vào tìm hiểu phương pháp RSA. Bên cạnh đó, chúng ta cũng đề cập đến phương pháp trao đổi khóa Diffie-Hellman, một cách áp dụng hàm một chiều nhưng không phải để mã hóa. Tuy nhiên trước tiên chúng ta sẽ tìm hiểu sơ lược về *lý thuyết số*, đây là nền tảng toán học của phương pháp mã hóa khóa công khai.

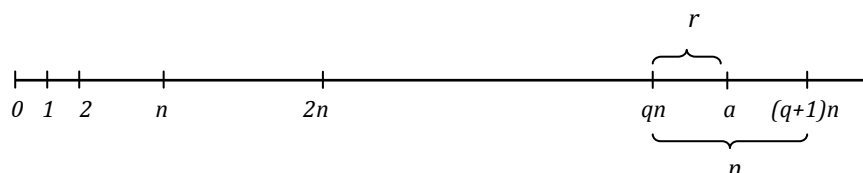
## 4.1 Lý thuyết số

### 4.1.1 Một số khái niệm

#### 1. Phép chia modulo:

Phép chia modulo là phép chia lấy phần dư. Ví dụ:  $27 \bmod 8 = 3$ ,  $35 \bmod 9 = 8$ .  
Một cách tổng quát:

$$a \bmod n = r \text{ với } a \geq 0; n > 0; 0 \leq r \leq n - 1$$



Nếu hai số  $a, b$  có cùng số dư trong phép chia cho  $n$  thì ta nói rằng  $a$  và  $b$  là đồng dư trong phép chia modulo cho  $n$ , phép so sánh đồng dư được ký hiệu bằng dấu  $\equiv$ :

$$a \equiv b \pmod{n} \text{ hay viết tắt là } a \equiv b \bmod n$$

Có thể thấy, phép toán modulo phân hoạch tập số tự nhiên  $\mathbb{N}$  thành  $n$  lớp tương đương đồng dư ứng với các giá trị của  $r$  trong tập  $\{0, 1, 2, 3, \dots, n-1\}$ . Ví dụ với  $n = 4$  ta có 4 lớp tương đương sau:

$$\{0, 4, 8, 12, 16, \dots\}$$

$$\{1, 5, 9, 13, 17, \dots\}$$

$$\{2, 6, 10, 14, 18, \dots\}$$

$$\{3, 7, 11, 15, 19, \dots\}$$

#### 2. Một số tính chất của phép modulo:

Cho  $a, b$  và  $n$  là các số nguyên, phép modulo có các tính chất:

$$a) (a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$$

$$b) (a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$$

$$c) (a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$$

#### 3. Ước số:

Nếu  $a \bmod n = 0$  (viết cách khác  $a \equiv 0 \pmod{n}$ ) thì có nghĩa là  $a$  chia hết cho  $n$ , hay  $n$  là ước số của  $a$ .

Ước số chung lớn nhất của hai số: ký hiệu  $\gcd(a, b)$ . Để tìm USCLN của hai số  $a, b$ , chúng ta có thể dùng thuật toán Euclid (xem Phụ lục 2).

#### 4. Số nguyên tố

Một số  $p$  được gọi là số nguyên tố nếu  $p$  chỉ chia hết cho 1 và chính nó, ngoài ra không chia hết cho số nào khác từ 2 đến  $p - 1$ .

#### 5. Số nguyên tố cùng nhau

Hai số nguyên  $a, b$  được gọi là nguyên tố cùng nhau nếu USCLN của  $a$  và  $b$  là 1. Ký hiệu:  $a \perp b$ . Ví dụ:  $3 \perp 8$ ,  $7 \perp 9$ ,  $4 \perp 15$ . Hai số 20 và 15 không nguyên tố cùng nhau vì có USCLN là 5.

#### 6. Phần tử nghịch đảo của phép nhân modulo:

Nếu hai số nguyên  $a$  và  $n$  nguyên tố cùng nhau, thì tồn tại số nguyên  $w$  sao cho:

$$a \cdot w \equiv 1 \pmod{n}$$

Ta gọi  $w$  là phần tử nghịch đảo của  $a$  trong phép modulo cho  $n$  và ký hiệu là  $a^{-1}$

Ví dụ:

- $n = 10, a = 7$  là hai số nguyên tố cùng nhau, do đó tìm được  $a^{-1} = 3$  ( $21 \equiv 1 \pmod{10}$ )

$a^{-1}$	0	1	2	3	4	5	6	7	8	9
$a^{-1} \times 7$	0	7	4	1	8	5	2	9	6	3

- $n = 10, a = 2$  không phải là hai số nguyên tố cùng nhau, ta có bảng phép nhân sau:

$a^{-1}$	0	1	2	3	4	5	6	7	8	9
$a^{-1} \times 2$	0	2	4	6	8	0	2	4	6	8

Trong bảng trên không tồn tại số  $a^{-1}$  nào sao cho  $a \cdot a^{-1} \equiv 1 \pmod{10}$ . Vậy không tồn tại phần tử nghịch đảo.

Để tính  $a^{-1}$  chúng ta dùng thuật toán Euclid mở rộng (xem Phụ lục 2)

#### 4.1.2 Định lý Fermat

Định lý:

Nếu  $p$  là số nguyên tố và  $a$  là số nguyên không chia hết cho  $p$  thì  $a^{p-1} \equiv 1 \pmod{p}$

Chứng minh:

Xét tập  $X$  gồm  $p - 1$  phần tử sau:

$$X = \{ a \pmod{p}, 2a \pmod{p}, \dots, (p-1)a \pmod{p} \}$$

Ta có hai nhận xét sau:

- Không có phần tử nào của tập  $X$  bằng 0 vì  $a$  nguyên tố cùng nhau với  $p$ .
- Không tồn tại hai phần tử thứ  $i$  và thứ  $j$  ( $i \neq j$ ) sao cho:  $ia \pmod{p} = ja \pmod{p}$ .  
Vì  $a$  nguyên tố cùng nhau với  $p$  nên tồn tại  $a^{-1}$  trong phép modulo  $p$ . Do đó nếu  $ia \equiv ja \pmod{p}$  thì  $iaa^{-1} \equiv jaa^{-1} \pmod{p}$  nghĩa là  $i \equiv j \pmod{p}$ . Điều này trái với giả thiết  $i \neq j$ .

Từ hai nhận xét trên ta suy ra các phần tử của  $X$  sẽ là một hoán vị của các giá trị  $\{1, 2, \dots, p-1\}$ . Do đó:

$$\begin{aligned} a \times 2a \times \dots \times (p-1)a &\equiv [1 \times 2 \times \dots \times (p-1)] \pmod{p} \\ \Rightarrow a \times a \times \dots \times a &= a^{p-1} \equiv 1 \pmod{p} \quad (\text{đpcm}) \end{aligned}$$

Sau đây là một số ví dụ của định lý Fermat:

- $p = 5, a = 7 \Rightarrow 7^4 = 49 \cdot 49 = 2401, 2401 \equiv 1 \pmod{5}$
- $p = 7, a = 4 \Rightarrow 4^6 = 64 \cdot 64 = 4096, 4096 \equiv 1 \pmod{7}$

#### 4.1.3 Phép logarit rời rạc

Ta định nghĩa phép lũy thừa modulo như sau, để tính  $y$  từ  $a, x$  và  $n$  là các số nguyên:

$$y = a^x \pmod{n} = (a \cdot a \dots a) \pmod{n} \quad \text{với } x \text{ số } a \text{ nhân với nhau}$$

Ta chỉ xét trường hợp  $n$  là số nguyên tố. Bảng sau minh họa các giá trị của phép lũy thừa modulo với  $n = 19$ ,  $a$  và  $x$  từ 1 đến 18.

$a$	$a^2$	$a^3$	$a^4$	$a^5$	$a^6$	$a^7$	$a^8$	$a^9$	$a^{10}$	$a^{11}$	$a^{12}$	$a^{13}$	$a^{14}$	$a^{15}$	$a^{16}$	$a^{17}$	$a^{18}$
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

**Bảng 4-1. Bảng giá trị lũy thừa modulo với  $n = 19$**

Nhìn vào bảng trên, ta thấy rằng không phải hàng nào cũng có đầy đủ các giá trị từ 1 đến 18. Xét hàng  $a = 11$ , ta có:

- $11^1 \equiv 11 \pmod{19}$  (\*)
- $11^2 = 121 \equiv 7 \pmod{19}$
- $11^3 = 1331 \equiv 1 \pmod{19}$
- $11^4 \equiv 11^3 \cdot 11 \equiv 11 \pmod{19}$  (giống như hàng (\*))
- $11^5 \equiv 11^2 \pmod{19}$
- ....

Do đó hàng  $a = 11$  (tương ứng với dãy  $11^1, 11^2, \dots, 11^{18}$ ) chỉ có ba giá trị 11, 7, 1 được lặp lại theo chu kỳ.

Trong bảng trên chỉ có các giá trị  $a = 2, 3, 10, 13, 14, 15$  là làm cho dãy  $a^1, a^2, \dots, a^{18}$  có đầy đủ các giá trị từ 1 đến 18 với phép modulo 19. Như vậy chỉ có  $a = 2, 3, 10, 13, 14, 15$  thì phép lũy thừa modulo trên mới *khả nghịch*.

Trong trường hợp tổng quát với mỗi  $n$  chỉ có một số trường hợp của  $a$  thì phép lũy thừa là khả nghịch. Lúc này  $a$  được gọi là *primitive root* của  $n$ .

Và cũng tương tự như số thực, nếu biết  $y$ ,  $a$  và  $n$ , muốn tìm lại  $x$  thì ta cũng dùng hàm logarith, được gọi là logarith rời rạc.

$$x = d\log_{a,n} y$$

Tuy nhiên không giống như số thực, việc tính logarith rời rạc đã được chứng minh là rất tốn kém về mặt thời gian. Và được xem như là bất khả thi nếu  $a$  và  $n$  là các số lớn. Do đó

phép lũy thừa modulo cũng được xem là *hàm một chiều* và được ứng dụng trong phương pháp trao đổi khóa Diffie – Hellman.

## 4.2 RSA

Phương pháp RSA là một phương pháp mã hóa khóa công khai. RSA được xây dựng bởi các tác giả Ron Rivest, Adi Shamir và Len Adleman tại học viện MIT vào năm 1977, và ngày nay đang được sử dụng rộng rãi. Về mặt tổng quát RSA là một phương pháp mã hóa theo khối. Trong đó bản rõ  $M$  và bản mã  $C$  là các số nguyên từ 0 đến  $2^i$  với  $i$  số bit của khối. Kích thước thường dùng của  $i$  là 1024 bit. RSA sử dụng hàm một chiều là vấn đề phân tích một số thành thừa số nguyên tố.

### 4.2.1 Nguyên tắc thực hiện của RSA

Để thực hiện mã hóa và giải mã, RSA dùng phép lũy thừa modulo của lý thuyết số. Các bước thực hiện như sau:

- 1) Chọn hai số nguyên tố lớn  $p$  và  $q$  và tính  $N = pq$ . Cần chọn  $p$  và  $q$  sao cho:  
 $M < 2^{i-1} < N < 2^i$ . Với  $i = 1024$  thì  $N$  là một số nguyên dài khoảng 309 chữ số.
- 2) Tính  $n = (p - 1)(q - 1)$
- 3) Tìm một số  $e$  sao cho  $e$  nguyên tố cùng nhau với  $n$
- 4) Tìm một số  $d$  sao cho  $e \cdot d \equiv 1 \pmod{n}$  ( $d$  là nghịch đảo của  $e$  trong phép modulo  $n$ )
- 5) Hủy bỏ  $n, p$  và  $q$ . Chọn khóa công khai  $K_U$  là cặp  $(e, N)$ , khóa riêng  $K_R$  là cặp  $(d, N)$
- 6) Việc mã hóa thực hiện theo công thức:
  - Theo phương án 1, mã hóa bảo mật:  $C = E(M, K_U) = M^e \pmod{N}$
  - Theo phương án 2, mã hóa chứng thực:  $C = E(M, K_R) = M^d \pmod{N}$
- 7) Việc giải mã thực hiện theo công thức:
  - Theo phương án 1, mã hóa bảo mật:  $\bar{M} = D(C, K_R) = C^d \pmod{N}$
  - Theo phương án 2, mã hóa chứng thực:  $\bar{M} = D(C, K_U) = C^e \pmod{N}$

Bản rõ  $M$  có kích thước  $i-1$  bit, bản mã  $C$  có kích thước  $i$  bit.

Để đảm bảo rằng RSA thực hiện đúng theo nguyên tắc của mã hóa khóa công khai, ta phải chứng minh hai điều sau:

- a) Bản giải mã chính là bản rõ ban đầu:  $\bar{M} = M$ , xét phương án 1:

Từ bước 4 ta suy ra:

$$ed = kn + 1 \quad \text{với } k \text{ là một số nguyên nào đó}$$

$$\begin{aligned} \text{Vậy: } \bar{M} &= C^d \pmod{N} \\ &= M^{ed} \pmod{N} \\ &= M^{kn+1} \pmod{N} \\ &= M^{k(p-1)(q-1)+1} \pmod{N} \end{aligned}$$

Trước tiên ta chứng minh:  $M^{k(p-1)(q-1)+1} \equiv M \pmod{p}$ . Xét hai trường hợp của  $M$ :

- $M$  chia hết cho  $p$ :  $M \pmod{p} = 0$  do đó  $M^{k(p-1)(q-1)+1} \equiv M \equiv 0 \pmod{p}$



- $M$  không chia hết cho  $p$ , vì  $p$  là số nguyên tố nên suy ra  $M$  nguyên tố cùng nhau với  $p$ . Vậy:

$$\begin{aligned} M^{k(p-1)(q-1)+1} \bmod p &= M \cdot (M^{p-1})^{k(q-1)} \bmod p \\ &= M \cdot 1^{k(q-1)} \bmod p \quad (\text{theo định lý Fermat}) \\ &= M \bmod p \end{aligned}$$

Vậy:  $M^{k(p-1)(q-1)+1} - M \equiv 0 \bmod p$  với mọi  $M$ . Hay nói cách khác  $M^{k(p-1)(q-1)+1} - M$  chia hết cho  $p$ . Chứng minh tương tự ta có  $M^{k(p-1)(q-1)+1} - M$  chia hết cho  $q$ . Vì  $p, q$  là hai số nguyên tố nên suy ra  $M^{k(p-1)(q-1)+1} - M$  chia hết cho  $N = pq$ . Tóm lại:

$$\begin{aligned} M^{k(p-1)(q-1)+1} &\equiv M \bmod N \\ \Rightarrow \bar{M} &= M^{k(p-1)(q-1)+1} \bmod N = M \quad (\text{do } M < N) \quad (\text{đpcm}). \end{aligned}$$

Vì  $e$  và  $d$  đối xứng nên có thể thấy trong phương án 2, ta cũng có  $\bar{M} = M$ .

- b) Không thể suy ra  $K_R$  từ  $K_U$ , nghĩa là tìm cặp  $(d, N)$  từ cặp  $(e, N)$ :

Có  $e$  và  $N$ , muốn tìm  $d$ , ta phải dựa vào công thức:  $e \cdot d \equiv 1 \bmod n$ . Do đó phải tính được  $n$ . Vì  $n = (p-1)(q-1)$  nên suy ra phải tính được  $p$  và  $q$ . Vì  $N = pq$  nên ta chỉ có thể tính được  $p$  và  $q$  từ  $N$ . Tuy nhiên điều này là bất khả thi vì  $N = pq$  là hàm một chiều. Vậy không thể tính được  $K_R$  từ  $K_U$ .

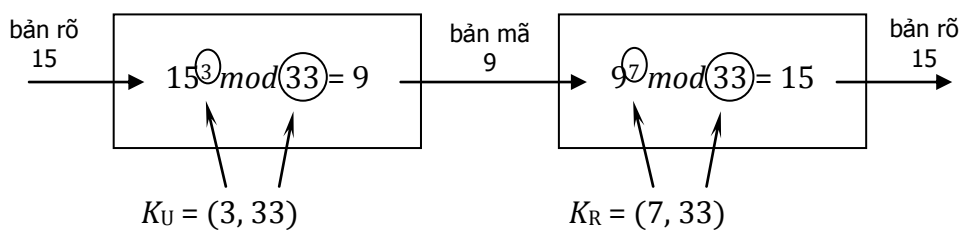
#### 4.2.2 Ví dụ RSA

Để minh họa ta sẽ thực hiện một ví dụ về mã hóa RSA với kích thước khóa là 6 bit.

- 1) Chọn  $p = 11$  và  $q = 3$ , do đó  $N = pq = 33$  ( $2^5 = 32 < 33 < 64 = 2^6$ )
- 2)  $n = (p-1)(q-1) = 20$
- 3) Chọn  $e = 3$  nguyên tố cùng nhau với  $n$
- 4) Tính nghịch đảo của  $e$  trong phép modulo  $n$  được  $d = 7$  ( $3 \times 7 = 21$ )
- 5) Khóa công khai  $K_U = (e, N) = (3, 33)$ . Khóa bí mật  $K_R = (d, N) = (7, 33)$

Theo phương án 1 (mã hóa bảo mật):

- 6) Mã hóa bản rõ  $M = 15$ :  
 $C = M^e \bmod N = 15^3 \bmod 33 = 9$  (vì  $15^3 = 3375 = 102 \times 33 + 9$ )
- 7) Giải mã bản mã  $C = 9$ :  
 $\bar{M} = C^d \bmod N = 9^7 \bmod 33 = 15 = M$  (vì  $9^7 = 4.782.696 = 144.938 \times 33 + 15$ )



Theo phương án 2 (mã hóa chứng thực):

- 6) Mã hóa bản rõ  $M = 15$ :  
 $C = M^d \bmod N = 15^7 \bmod 33 = 27$  (vì  $15^7 = 170.859.375 = 5177.556 \times 33 + 27$ )

7) Giải mã bản mã  $C = 9$ :

$$\bar{M} = C^e \bmod N = 27^3 \bmod 33 = 15 = M \text{ (vì } 27^3 = 19.683 = 596 \times 33 + 15 \text{)}$$

### 4.3 Độ phức tạp tính toán trong RSA

Có hai vấn đề về độ phức tạp tính toán trong phương pháp RSA. Đó là các phép tính sinh khóa và các phép tính mã hóa/giải mã

#### 4.3.1 Phép tính mã hóa/giải mã

Phép tính mã hóa và giải mã dùng phép lũy thừa modulo. Để an toàn, thường phải chọn  $N, e, d, M$  lớn. Nếu thực hiện bằng cách tính phép lũy thừa trước sau đó rút gọn modulo, thì giá trị của phép lũy thừa là quá lớn để có thể lưu trữ và tính toán. Tuy nhiên số học modulo có một tính chất sau:

$$(a \times b) \equiv [(a \bmod n) \times (b \bmod n)] \bmod n$$

Chúng ta có thể sử dụng tính chất này để đơn giản phép tính lũy thừa modulo thông qua một phương pháp gọi là “bình phương liên tiếp”. Ví dụ cần tính  $x^{16} \bmod n$ , đầu tiên sẽ tính  $a = x \bmod n$ , tiếp theo là  $b = x^2 \bmod n = a^2 \bmod n$ , tiếp theo là  $c = x^4 \bmod n = b^2 \bmod n$ , tiếp theo là  $d = x^8 \bmod n = c^2 \bmod n$ , và cuối cùng  $x^{16} \bmod n = d^2 \bmod n$ . Các số  $a, b, c, d$  luôn nhỏ hơn  $n$  do đó tránh được việc tính số lũy thừa lớn đồng thời nâng cao tốc độ tính toán.

Trong trường hợp tổng quát để tính  $x^b \bmod n$ , ta viết  $b$  dưới dạng số nhị phân.

$$b = b_k b_{k-1} \dots b_2 b_1 b_0 \quad \text{trong đó } b_i \text{ là các bit } 0, 1$$

Như vậy:  $b = \sum_{b_i \neq 0} 2^i$

Do đó  $x^b = \prod_{b_i \neq 0} x^{2^i} \bmod n$

Như vậy ứng với các  $b_i = 1$ , ta dùng phương pháp bình phương liên tiếp để tính các  $x^{2^i} \bmod n$ . Sau đó nhân các kết quả với nhau và rút gọn modulo để có kết quả cuối cùng

Ví dụ, tính  $y = 5^{20} \bmod 11$ : số 20 viết dưới dạng nhị phân là 10100, có bit 2 và bit 4 bằng 1. (nghĩa là  $20 = 16 + 4$ )

$$5^1 = 5 \bmod 11$$

$$5^2 = (5^1)^2 = 25 \equiv 3 \bmod 11$$

$$5^4 = (5^2)^2 = 3^2 \equiv 9 \bmod 11$$

$$5^8 = (5^4)^2 = 9^2 = 81 \equiv 4 \bmod 11$$

$$5^{16} = (5^8)^2 = 4^2 = 16 \equiv 5 \bmod 11$$

$$\text{Kết quả cuối cùng } y = 5^{20} \bmod 11 = (5^4 \bmod 11)(5^{16} \bmod 11) = 9 \times 5 \bmod 11 = 1$$

/\* Thuật toán tính lũy thừa modulo  $x^b \bmod n$  \*/

---

```

a = x; y = 1;
do
    if (b mod 2 <> 0)
        y = (y*a) mod n;
    end if
    b = b shr 1;
    a = (a*a) mod n;

```

```
while (b>0);
return y;
```

---

Để tăng tốc quá trình mã hóa dữ liệu, người ta thường chọn một khóa công khai  $e$  cụ thể nào đó. Thường thì  $e$  là 65537 ( $2^{16} + 1$ ). Ngoài ra còn có thể chọn  $e$  là 3 hoặc 17. Lý do để chọn các con số này là lúc đó  $e$  chỉ có 2 bit 1. Do đó giảm được phép các phép nhân trong phép tính lũy thừa modulo (một điểm cần chú ý là phải chọn  $p$  và  $q$  sao cho  $e$  nguyên tố cùng nhau với  $n$ ). Tuy nhiên nếu chọn  $e$  quá nhỏ thì RSA sẽ bị phá mã bằng cách sử dụng định lý số dư Trung Hoa (xem Phụ Lục 2-3).

Giả sử chọn  $e = 3$  và Alice gửi cùng một thông điệp  $M$  cho ba người khác có khóa công khai lần lượt là  $(3, N_1)$ ,  $(3, N_2)$ ,  $(3, N_3)$ . Ba bản mã sẽ là  $C_1 = M^3 \bmod N_1$ ,  $C_2 = M^3 \bmod N_2$ ,  $C_3 = M^3 \bmod N_3$ . Thường thì  $N_1, N_2, N_3$  sẽ nguyên tố cùng nhau theo từng cặp. Theo nguyên tắc của RSA,  $M$  đều nhỏ hơn  $N_1, N_2, N_3$  nên  $M^3 < N_1 N_2 N_3$ . Vậy xét dưới góc độ của định lý số dư Trung Hoa:

$$N_1 N_2 N_3 \Leftrightarrow T$$

$$M^3 \Leftrightarrow A$$

$$N_1 \Leftrightarrow t_1, N_2 \Leftrightarrow t_2, N_3 \Leftrightarrow t_3$$

$$C_1 \Leftrightarrow a_1, C_2 \Leftrightarrow a_2, C_3 \Leftrightarrow a_3$$

Theo định lý số dư Trung Hoa, nếu biết  $a_1, a_2, a_3, t_1, t_2, t_3$  ta có thể khôi phục được giá trị  $A$ . Vậy nếu Trudy có được  $C_1, C_2, C_3$  của Alice, Trudy sẽ tìm được  $M^3$  và từ đó tính được  $M$ .

Vì ta đã chọn  $e$  nhỏ nên  $d$  tìm được sẽ khá lớn. Do đó ngoài việc áp dụng phương pháp bình phương liên tiếp để tính  $M = C^d \bmod N$ , người ta còn áp dụng định lý số dư Trung Hoa để tăng tốc độ tính toán lên 4 lần. Cách thực hiện như sau:

Bổ sung vào khóa riêng các tham số  $p, q, dP, dQ, qInv$ , với:

- $p, q$  là hai số nguyên tố để tính  $N$
- $dP.e \equiv 1 \bmod p$
- $dQ.e \equiv 1 \bmod q$
- $qInv.q \equiv 1 \bmod p$  (với giả định  $p > q$ )

Áp dụng định lý số dư Trung Hoa theo phương án Garner, thay vì tính  $M = C^d \bmod N$ , chúng ta có thể tính  $M$  như sau:

- $m_1 = C^{dP} \bmod p$
- $m_2 = C^{dQ} \bmod q$
- $h = qInv(m_1 - m_2) \bmod p$
- $M = m_2 + qh$

Với cách thực hiện trên, chúng ta chỉ tính phép modulo trên các số  $p, q$  có kích thước nhỏ hơn  $N$ .

Chứng minh:

Tương tự như cách chứng minh RSA, đặt  $e.dP = u(p - 1) + 1$ , ta có:

$$m_1 = C^{dP} \bmod p = (M^e \bmod N)^{dP} \bmod p = M^{e.dP} \bmod p \quad (\text{vì } N \text{ chia hết cho } p)$$

$$= M^{u(p-1)+1} \bmod p$$

Từ đó áp dụng định lý Fermat bé, ta suy ra:  $m_1 = M \bmod p$

Tương tự, ta cũng có  $m_2 = M \bmod q$

Do đó, xét theo định lý số dư Trung Hoa theo phương án Garner thì  $M$  trong  $Z_N$  tương ứng với cặp  $(m_1, m_2)$  trong  $Z_p \times Z_q$ :

$$N \Leftrightarrow T, M \Leftrightarrow A, p \Leftrightarrow t_2, q \Leftrightarrow t_1, m_1 \Leftrightarrow a_2, m_2 \Leftrightarrow a_1, qInv \Leftrightarrow c_{12}$$

Ta có:

$$b_1 = a_1 \bmod t_1 = m_2$$

$$b_2 = (a_2 - b_1)c_{12} \bmod t_2 = qInv(m_1 - m_2) \bmod p = h$$

Vậy:

$$M = A = b_2 t_1 + b_1 = m_2 + qh \text{ (đpcm).}$$

#### 4.3.2 Phép tính sinh khóa

Phép tính sinh khóa là chọn  $p$  và  $q$  nguyên tố để tính  $N$ . Để phân tích số  $N$  thành tích hai thừa số nguyên tố  $p, q$ , chỉ có một cách duy nhất là thử từng số  $p$  và  $q$ . Do đó phải chọn  $p, q$  đủ lớn để việc thử là không khả thi. Hiện nay chưa có phương pháp nào để sinh ra số nguyên tố lớn tùy ý. Chỉ có cách là chọn một số lẻ ngẫu nhiên nào đó và kiểm tra số đó có phải là số nguyên tố không. Việc kiểm tra tính nguyên tố cũng gặp nhiều khó khăn. Thuật toán kiểm tra số nguyên tố hiệu quả hiện nay là thuật toán Miller-Rabin (xem Phụ lục 2), dù rằng không hoàn toàn chính xác 100%, tuy nhiên có thể đạt sai số nhỏ không đáng kể.

Dựa vào lý thuyết số nguyên tố, người ta ước tính rằng cần thử trung bình khoảng 70 số lẻ để tìm ra một số nguyên tố lớn chừng  $2^{200}$ .

Vì chúng ta đã chọn  $e$  trước là 65537 (hay 3 hoặc 17 ...), do đó cần kiểm tra xem  $e$  có nguyên tố cùng nhau với  $n = (p-1)(q-1)$  hay không. Nếu không ta phải thử lại với  $p$  và  $q$  khác. Sau khi đã tìm  $p$  và  $q$  thích hợp, cần tìm  $d$  sao cho  $e \cdot d \equiv 1 \bmod n$ . Bằng cách dùng thuật toán Euclid mở rộng, chúng ta có thể kết hợp việc kiểm tra tính nguyên tố cùng nhau của  $e$  và  $n$ , đồng thời nếu  $e$  nguyên tố cùng nhau với  $n$  thì thuật toán cũng cho biết  $d$ . Vì vậy không cần tiến hành bước tìm  $d$  riêng.

#### 4.4 Độ an toàn của RSA

Sau đây ta sẽ xem xét một số các tấn công phương pháp RSA.

- 1) Vết cạn khóa: cách tấn công này thử tất cả các khóa  $d$  có thể có để tìm ra bản giải mã có ý nghĩa, tương tự như cách thử khóa  $K$  của mã hóa đối xứng. Với  $N$  lớn, việc tấn công là bất khả thi.
- 2) Phân tích  $N$  thành thừa số nguyên tố  $N = pq$ : Chúng ta đã nói rằng việc phân tích phải là bất khả thi thì mới là hàm một chiều, là nguyên tắc hoạt động của RSA. Tuy nhiên, nhiều thuật toán phân tích mới đã được đề xuất, cùng với tốc độ xử lý của máy tính ngày càng nhanh, đã làm cho việc phân tích  $N$  không còn quá khó khăn như trước đây. Năm 1977, các tác giả của RSA đã treo giải thưởng cho ai phá được RSA có kích thước của  $N$  vào khoảng 428 bit, tức 129 chữ số. Các tác giả này ước đoán phải mất 40 nghìn triệu triệu năm mới có thể giải được. Tuy

nhiên vào năm 1994, câu đố này đã được giải chỉ trong vòng 8 tháng. Bảng sau liệt kê kích thước  $N$  của các RSA đã phá mã được cho đến hiện nay

<i>Số chữ số của <math>N</math></i>	<i>Số bit</i>	<i>Năm phá mã</i>	<i>Thuật toán</i>
100	322	1991	Quadratic sieve
110	365	1992	Quadratic sieve
120	398	1993	Quadratic sieve
129	428	1994	Quadratic sieve
130	431	1996	GNFS
140	465	1999	GNFS
155	512	1999	GNFS
160	530	2003	Lattice sieve
174	576	2003	Lattice sieve
200	633	2005	Lattice sieve

**Bảng 4-2. Bảng liệt kê các mốc phá mã RSA**

Dĩ nhiên là việc phá mã trên chỉ được thực hiện trong phòng thí nghiệm. Tuy nhiên người ta cho rằng kích thước của  $N$  phải khoảng 1024 bit (309 chữ số) thì mới bảo đảm an toàn thật sự.

- 3) Đo thời gian: Đây là một phương pháp phá mã không dựa vào mặt toán học của thuật toán RSA, mà dựa vào một “hiệu ứng lẻ” sinh ra bởi quá trình giải mã RSA. Hiệu ứng lẻ đó là thời gian thực hiện giải mã. Giả sử người phá mã có thể đo được thời gian giải mã  $M = C^d \bmod N$  dùng thuật toán bình phương liên tiếp. Trong thuật toán bình phương liên tiếp, nếu một bit của  $d$  là 1 thì xảy ra hai phép modulo, nếu bit đó là 0 thì chỉ có một phép modulo, do đó thời gian thực hiện giải mã là khác nhau. Bằng một số phép thử chosen-plaintext, người phá mã có thể biết được các bit của  $d$  là 0 hay 1 và từ đó biết được  $d$ .

Phương pháp phá mã này là một ví dụ cho thấy việc thiết kế một hệ mã an toàn rất phức tạp. Người thiết kế phải lường trước được hết các tình huống có thể xảy ra.

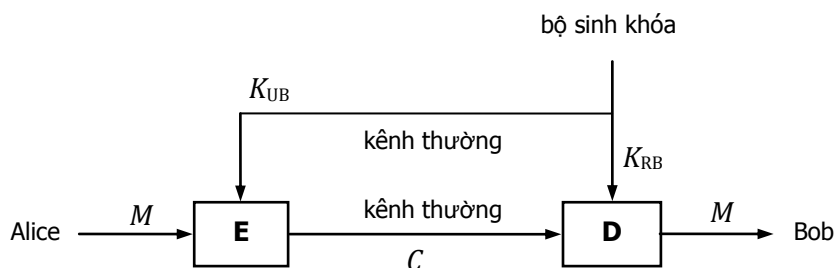
#### **4.5 Bảo mật, chứng thực và không từ chối với mã hóa khóa công khai**

Giả sử Alice muốn gửi dữ liệu cho Bob dùng mã hóa khóa công khai, trước tiên Alice và Bob sẽ chọn cặp khóa riêng-khóa công khai. Ký hiệu khóa riêng-khóa công khai của Alice là  $K_{RA}$  và  $K_{UA}$ , của Bob là  $K_{RB}$  và  $K_{UB}$ ,

Như vậy để gửi dữ liệu bảo mật cho Bob, Alice sẽ dùng phương án 1: mã hóa dữ liệu bằng khóa công khai  $K_{UB}$  của Bob, và Bob dùng khóa riêng  $K_{RB}$  để giải mã.

$$C = E(M, K_{UB})$$

$$M = D(C, K_{RB})$$

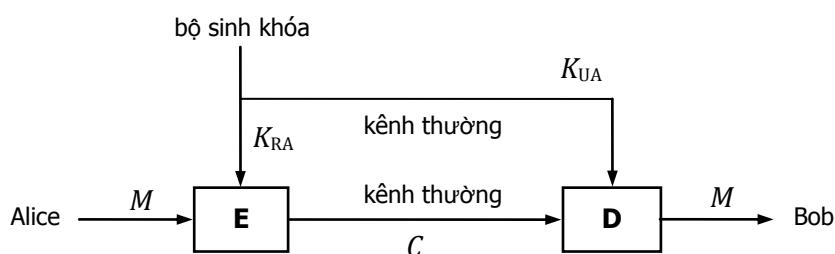


**Hình 4-1. Mô hình bảo mật với mã hóa khóa công khai**

Để đảm bảo tính chứng thực và Alice không từ chối trách nhiệm gửi dữ liệu, Alice sẽ dùng phương án 2: Alice mã hóa dữ liệu bằng khóa riêng  $K_{RA}$ , và Bob dùng khóa công khai  $K_{RA}$  của Alice để giải mã.

$$C = E(M, K_{RA})$$

$$M = D(C, K_{UA})$$



**Hình 4-2. Mô hình không thoái thác với mã hóa khóa công khai**

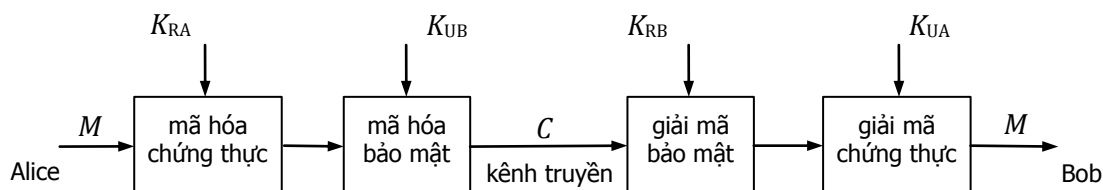
Cũng với giả định rằng thông điệp có ý nghĩa là một dãy bit có cấu trúc, nếu bản giải mã của Bob là hợp lệ thì rõ ràng là Alice là người gửi vì chỉ có Alice mới có khóa riêng  $K_{RA}$ . Giống như mã hóa đối xứng, nếu Trudy can thiệp chỉnh sửa trên bản mã  $C$  thì Bob sẽ giải mã ra bản rõ là một dãy bit vô nghĩa. Còn nếu Trudy có được khóa  $K_{RA}$  thì Alice không thể thoái thác trách nhiệm làm lộ khóa.

Tuy nhiên mô hình như trên lại không đảm bảo tính bảo mật. Vì không chỉ riêng Bob, Trudy cũng biết được khóa công khai  $K_{UA}$  của Alice. Do đó Trudy có thể giải mã bản mã  $C$  và biết được nội dung bản rõ  $M$ .

Để giải quyết vấn đề trên, người ta kết hợp tính bảo mật, chứng thực và không từ chối qua mô hình sau:

$$C = E(E(M, K_{RA}), K_{UB})$$

$$M = D(D(C, K_{RB}), K_{UA})$$

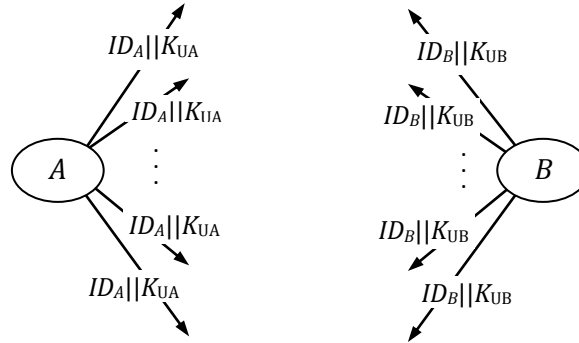


**Hình 4-3. Mô hình kết hợp bảo mật, chứng thực và không từ chối**

## 4.6 Trao đổi khóa

#### 4.6.1 Trao đổi khóa công khai

Khi hai người sử dụng muốn truyền dữ liệu với nhau bằng phương pháp mã hóa khóa công khai, trước tiên họ phải trao đổi khóa công khai cho nhau. Vì đây là khóa công khai nên không cần giữ bí mật việc trao đổi này, khóa có thể truyền công khai trên kênh thường. Alice và Bob, hay bất cứ người nào khác có thể công bố rộng rãi khóa công khai của mình theo mô hình bên dưới:



Hình 4-4. Trao đổi khóa công khai tự phát

Tuy nhiên ở đây chúng ta lại gặp phải vấn đề về chứng thực. Làm như thế nào mà Alice có thể đảm bảo rằng  $K_{UB}$  chính là khóa công khai của Bob? Trudy có thể mạo danh Bob bằng cách lấy khóa  $K_{UT}$  của Trudy và nói rằng đó là khóa công khai của Bob.

Vì vậy, việc trao đổi khóa công khai theo mô hình trên đặt gánh nặng lên vai của từng cá nhân. Alice muốn gửi thông điệp cho Bob hay bất cứ người nào khác thì phải tin tưởng vào khóa công khai của Bob hay của người đó. Tương tự như vậy cho Bob.

Để giảm gánh nặng cho từng cá nhân, một mô hình gọi là ‘chứng chỉ khóa công khai’ (public-key certificate) được sử dụng. Trong mô hình này có một tổ chức làm nhiệm vụ cấp chứng chỉ được gọi là trung tâm chứng thực (Certificate Authority – CA). Các bước thực hiện cấp chứng chỉ cho Alice như sau:

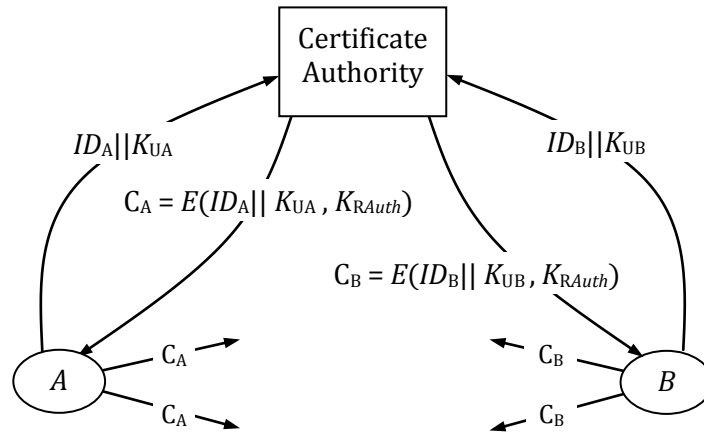
- 1) Alice gửi định danh  $ID_A$  và khóa công khai  $K_{UA}$  của mình đến trung tâm chứng thực.
- 2) Trung tâm chứng nhận kiểm tra tính hợp lệ của Alice, ví dụ nếu  $ID_A$  là ‘Microsoft’, thì Alice phải có bằng chứng chứng tỏ mình thực sự là công ty Microsoft.
- 3) Dựa trên cơ sở đó, trung tâm chứng thực cấp một chứng chỉ  $C_A$  để xác nhận rằng khóa công khai  $K_{UA}$  đó là tương ứng với  $ID_A$ . Chứng chỉ được ký chứng thực bằng khóa riêng của trung tâm để đảm bảo rằng nội dung của chứng chỉ là do trung tâm ban hành.

$$C_A = E(ID_A || K_{UA}, K_{RAuth})$$

(|| là phép nối dãy bit)

- 4) Alice công khai chứng chỉ  $C_A$ .
- 5) Bob muốn trao đổi thông tin với Alice thì sẽ giải mã  $C_A$  bằng khóa công khai của trung tâm chứng thực để có được khóa công khai  $K_{UA}$  của Alice. Do đó nếu Bob

tin tưởng vào trung tâm chứng thực thì Bob sẽ tin tưởng là  $K_{UA}$  là tương ứng với  $ID_A$ , tức tương ứng với Alice.



**Hình 4-5. Trao đổi khóa công khai dùng trung tâm chứng thực**

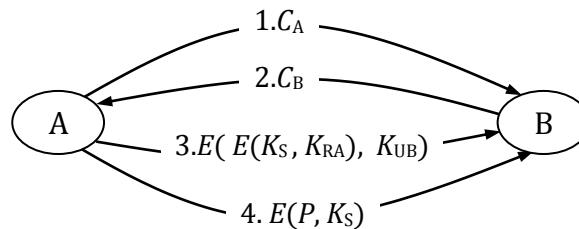
Như vậy có thể thấy rằng nếu Bob muốn gửi thông điệp cho Alice, Cindy, hay Darth..., thì Bob không cần phải tin tưởng vào khóa công khai của Alice, Cindy, hay Darth nữa. Bob chỉ cần tin tưởng vào trung tâm chứng thực và khóa công khai của trung tâm chứng thực là đủ.

Hiện nay mô hình chứng chỉ khóa công khai đang được áp dụng rộng rãi với chuẩn của chứng chỉ là chuẩn X.509. Trên thế giới có khoảng 80 tổ chức chứng thực chứng chỉ khóa công khai. Chúng ta sẽ tìm hiểu chi tiết hơn về chuẩn X.509 trong chương 7.

#### 4.6.2 Dùng mã hóa khóa công khai để trao đổi khóa bí mật

Do đặc điểm toán học của phương pháp mã hóa khóa công khai, thời gian mã hóa và giải mã của phương pháp này chậm hơn so với phương án mã hóa đối xứng. Trong thực tế, đối với vấn đề bảo đảm tính bảo mật, người ta vẫn sử dụng phương pháp mã hóa đối xứng. Mã hóa khóa công khai được dùng để thiết lập khóa bí mật cho mỗi phiên trao đổi dữ liệu. Lúc này khóa bí mật được gọi là khóa phiên (session key), các phiên trao đổi dữ liệu khác nhau sẽ dùng các khóa bí mật khác nhau.

Hình dưới mô tả một mô hình đơn giản để thiết lập khóa phiên  $K_S$  giữa Alice và Bob.



**Hình 4-6. Thiết lập khóa phiên bí mật bằng mã hóa khóa công khai**

Alice tạo một khóa phiên  $K_S$ , mã hóa bằng khóa riêng của Alice, sau đó mã hóa bằng khóa công khai của Bob. Bob giải mã  $K_S$  dùng khóa riêng của Bob và khóa công khai của Alice. Nhờ tính bảo mật, Alice biết chắc rằng ngoài Alice chỉ có Bob mới biết được  $K_S$ . Nhờ tính không từ chối, Bob biết rằng ngoài Bob chỉ có Alice mới biết được  $K_S$  vì Alice dùng khóa riêng để mã hóa  $K_S$ . Do đó  $K_S$  có thể dùng làm khóa bí mật cho mã hóa đối xứng



để trao đổi dữ liệu giữa Alice và Bob. Sau phiên trao đổi dữ liệu,  $K_S$  được hủy bỏ nên khóa bí mật này sẽ ít có khả năng bị lộ. Lúc này vai trò của mã hóa khóa công khai không phải là bảo mật dữ liệu nữa (việc này do mã hóa đối xứng đảm trách) mà là bảo đảm tính bí mật của khóa đối xứng, chỉ có A và B biết khóa  $K_S$ .

#### 4.7 Phương pháp trao đổi khóa Diffie – Hellman

Phương pháp trao đổi khóa Diffie-Hellman dùng để thiết lập một khóa bí mật giữa người gửi và người nhận mà không cần dùng đến mã hóa công khai như ở phần 4.6.2. Phương pháp này dùng hàm một chiều làm hàm logarith rời rạc. Diffie-Hellman không có ý nghĩa về mặt mã hóa giống như RSA.

Trước tiên Alice và Bob sẽ thống nhất sử dụng chung một số nguyên tố  $p$  và một số  $g$  nhỏ hơn  $p$  và là *primitive root* của  $p$  (nghĩa là phép toán  $g^x \bmod p$  khả nghịch). Hai số  $p$  và  $g$  không cần giữ bí mật. Sau đó Alice chọn một số  $a$  và giữ bí mật số  $a$  này. Bob cũng chọn một số  $b$  và giữ bí mật số  $b$ . Tiếp theo Alice tính và gửi  $g^a \bmod p$  cho Bob, Bob tính và gửi  $g^b \bmod p$  cho Bob. Trên cơ sở đó Alice tính:

$$(g^b)^a \bmod p = g^{ab} \bmod p$$

Bob tính:

$$(g^a)^b \bmod p = g^{ab} \bmod p$$

Do đó Alice và Bob có chung giá trị  $g^{ab} \bmod p$ . Giá trị này có thể dùng làm khóa cho phép mã hóa đối xứng.

Như vậy, kẻ phá mã Trudy có thể có được  $g$ ,  $p$ ,  $g^a$  và  $g^b$ . Muốn tính được  $g^{ab} \bmod p$ , Trudy không thể dùng cách:

$$g^a g^b \bmod p = g^{a+b} \bmod p \neq g^{ab} \bmod p$$

Muốn tính được  $g^{ab} \bmod p$ , Trudy phải tính được  $a$  hoặc được  $b$ . Tuy nhiên việc tính  $a$  hay  $b$  theo công thức:

$$a = d\log_{g,p} g^a \text{ hay } b = d\log_{g,p} g^b$$

là không khả thi do tính phức tạp của phép logarith rời rạc. Vậy Trudy không thể nào tính được  $g^{ab} \bmod p$ . Hay nói cách khác, khóa dùng chung được trao đổi bí mật giữa Alice và Bob.

Tuy nhiên, thuật toán Diffie-Hellman lại thất bại đối với cách tấn công kẻ-đứng-giữa. Trong phương pháp tấn công này, Trudy đứng giữa Alice và Bob. Trudy chặn các thông điệp của Alice và Bob, giả mạo các thông điệp mà Alice và Bob không hay biết. Alice vẫn nghĩ là nhận dữ liệu từ Bob và ngược lại.

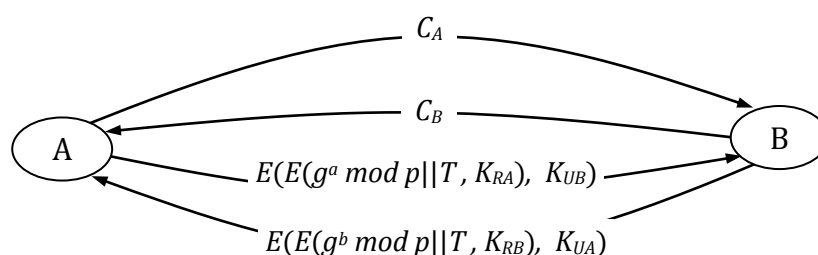
Do đó Trudy có thể thiết lập khóa Diffie-Hellman  $g^{at} \bmod p$  với Alice và khóa  $g^{bt} \bmod p$  với Bob. Khi Alice gửi dữ liệu, Trudy giải mã bằng khóa  $g^{at} \bmod p$ , sau đó mã hóa lại bằng  $g^{bt} \bmod p$  và gửi cho Bob. Như vậy Alice và Bob không hay biết còn Trudy thì xem trộm được dữ liệu.



#### Hình 4-7. Tấn công kẻ-đứng-giữa với phương pháp Diffie--Hellman

Để an toàn, quá trình thiết lập khóa Diffie-Hellman vẫn phải được mã hóa bằng một khóa công khai. Lúc này một câu hỏi được đặt ra là nếu đã được bảo vệ bằng khóa công khai rồi, thì có thể chọn khóa đối xứng bất kỳ, cần gì chọn khóa Diffie-Hellman? Tuy nhiên có một số trường hợp, khi mà cách thức tấn công kẻ-đứng-giữa không thể thực hiện được, thì phương thức Diffie-Hellman tỏ ra rất hữu dụng.

Trong mô hình trong phần 4.6.2, giả sử Trudy ghi nhận lại hết tất cả các thông điệp giữa Alice và Bob. Sau này nếu Trudy phát hiện ra được khóa riêng  $K_{RA}$  và  $K_{RB}$  của Alice và Bob, Trudy có thể khôi phục lại được khóa đối xứng  $K_S$ . Và từ đó Trudy có thể khôi phục lại các bản rõ mà được mã hóa bằng khóa  $K_S$  này. Bây giờ ta xét mô hình sau dùng Diffie-Hellman được bảo vệ bằng mã hóa khóa công khai:



Hình 4-8. Bảo vệ khóa Diffie-Hellman bằng khóa công khai

Trong mô hình trên, dù cho sau này Trudy phát hiện ra được khóa riêng  $K_{RA}$  và  $K_{RB}$  của Alice và Bob, và Trudy tìm ra được  $g^a \bmod p$  và  $g^b \bmod p$ . Tuy vậy, Trudy cũng không thể nào khôi phục lại được khóa bí mật  $g^{ab} \bmod p$ . Và do đó không thể khôi phục lại các bản rõ giữa Alice và Bob. Đây chính là ý nghĩa của phương pháp Diffie-Hellman.

## 4.8 Câu hỏi ôn tập

1. Nêu điểm yếu của mã hóa đối xứng.
2. Hàm một chiều là gì? Cho ví dụ về hàm một chiều.
3. Trong số học modulo  $n$ , khi nào thì một số có số nghịch đảo của phép nhân?
4. Logarit rời rạc khác logarit liên tục ở những điểm nào?
5. Để kiểm tra tính nguyên tố của một số nguyên, thuật toán Miller-Rabin có thể cho kết quả sai, vậy tại sao người ta vẫn sử dụng thuật toán này?
6. Tại sao trong thuật toán RSA cần dùng phương pháp bình phương liên tiếp để tính lũy thừa modulo?
7. Nêu nguyên tắc của mã hóa khóa công khai? Tại sao trong mã hóa khóa công khai không cần dùng đến kênh an toàn để truyền khóa?
8. Trong mã hóa khóa công khai, khóa riêng và khóa công khai có phải là 2 khóa tùy ý, không liên quan? Nếu có liên quan, tại sao không thể tính khóa riêng từ khóa công khai?
9. Ngoài vấn đề truyền khóa, mã hóa khóa công khai còn ưu điểm hơn mã hóa đối xứng ở điểm nào?
10. Nêu nhược điểm của mã hóa khóa công khai.
11. Diffie-Hellman không phải là một phương pháp mã hóa khóa công khai. Vậy Diffie-Hellman là gì?

## 4.9 Bài tập

1. Cho  $a = 13$ ,  $p = 20$ . Tìm giá trị nghịch đảo của  $a$  trong phép modulo  $p$  dùng thuật toán Euclid mở rộng (xem phụ lục 2).
2. Cho  $n = 17$ , lập bảng tương tự như Bảng 4-1. Liệt kê các *primitive root* của  $n$ .
3. Áp dụng thuật toán bình phương liên tiếp tính  $7^{21} \bmod 13$
4. Cho  $p = 5$ ,  $q = 11$ ,  $e = 7$ . Tính khóa riêng ( $d$ ,  $N$ ) trong phương pháp RSA.
5. Thực hiện mã hóa và giải mã bằng phương pháp RSA với  $p = 3$ ,  $q = 11$ ,  $e = 7$ ,  $M = 5$  theo hai trường hợp mã hóa bảo mật và mã hóa chứng thực.
6. Alice chọn  $p = 7$ ,  $q = 11$ ,  $e = 17$ , Bob chọn  $p = 11$ ,  $q = 13$ ,  $e = 11$ :
  - a. Tính khóa riêng  $K_{RA}$  của Alice và  $K_{RB}$  của Bob
  - b. Alice muốn gửi cho Bob bản tin  $M = 9$  vừa áp dụng chứng thực và bảo mật như ở sơ đồ 4-3. Hãy thực hiện quá trình mã hóa và giải mã.
7. Xét thuật toán Miller-Rabin (xem phụ lục 2). Với số 37, cho biết kết quả của thuật toán Miller-Rabin trong các trường hợp sau đây của  $a$ : 9, 17, 28.
8. Dùng thuật toán Miller-Rabin, kiểm tra tính nguyên tố của số 169.

## 4.10 Bài tập thực hành

1. Viết chương trình thể hiện thuật toán Euclid mở rộng áp dụng cho các số nguyên nhỏ 32 bit.
2. Viết chương trình sinh một số nguyên tố nhỏ (32 bit) dùng thuật toán Miller-Rabin.
3. Viết chương trình thể hiện thuật toán bình phương liên tiếp tính  $a^x \bmod p$  trên số nguyên nhỏ
4. Viết chương trình mã hóa file bằng thuật toán RSA trên số nguyên nhỏ.
5. Viết chương trình thực hiện các phép toán  $+$ ,  $-$ ,  $*$ ,  $\bmod$  trên các số nguyên lớn (kích thước tối đa một số nguyên là 1024 bit). Gợi ý: mỗi số nguyên được biểu diễn bằng một mảng các phần tử 32 bit.
6. Áp dụng bài 5, thực hiện lại các bài 1, 2, 3, 4 áp dụng trên số nguyên lớn.
7. Tìm hiểu về thuật toán RSA trong môi trường lập trình .NET (namespace System.Security.Cryptography). Viết chương trình mã hóa và giải mã một file trên máy tính dùng phương pháp RSA trong thư viện mã hóa của .NET. Khóa công khai và khóa riêng được lưu trong 1 file text dưới dạng chữ số thập lục phân.

RSA

Thông tin khóa RSA

P dfaecd918582c6c66513b682c4b6ddbce2ce0c79498ee032d53b16b09a6a1247ef8

Q dd2b1eed6add5ebbc5b370b35da91d03626cd4a6e853088d755011c865e71368b

Modulus c13f94d915fe7dba8ec492cca48d52df1d8cd4c3de52487f8bb40e923eef1e855ea70

D ac4c442a49b1dde7b7e30bffe9b7176c56ac7a0728f1b432041d45e09ebc143a556

DP c536040e3a2d1bfdcd1dfd3afc1ed20ee5a451a6051074409ed4fc4687e2be8e43d5

DQ d0fd3634528eec3db30b64ab990d7c84fe34a89d2f5b86f164b14c9c78ee69c0c1ef

InverseQ cacf3a53823f49dc1aef3c50d0303a0af5d8faf0af5706c9384f072ec57effbe04e8f6b

Exponent 010001

Nạp Lưu Sinh

File ban đầu D:\Books\Cac chuyen de lap trinh.pdf Chọn

File mã hóa D:\Books\Cac chuyen de lap trinh.pdf.rsa Chọn

Mã hóa Giải mã

## CHƯƠNG 5. MÃ CHỨNG THỰC THÔNG ĐIỆN, HÀM BẮM

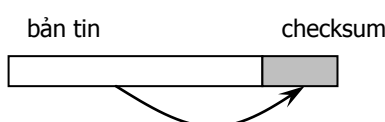
Để tìm hiểu về tính chứng thực của mã hóa đối xứng và mã hóa khóa công khai, trong chương 2, 3 và 4, chúng ta đã giả sử rằng một thông điệp có ý nghĩa thì phải có một cấu trúc nào đó. Chẳng hạn như đối với ngôn ngữ, một câu văn chỉ có ý nghĩa khi chữ cái được kết hợp với nhau theo các quy tắc từ vựng và ngữ pháp của một ngôn ngữ. Do đó nếu Trudy can thiệp sửa đổi bản mã thì bản giải mã sẽ là một chuỗi bits vô nghĩa, và người nhận biết được là dữ liệu đã bị thay đổi. Ta có hai kết luận sau về tính chứng thực của mã hóa đối xứng và mã hóa khóa công khai:

- KL1: Trudy không thể tìm ra một bản mã  $C_T$ , sao cho khi Bob giải mã bằng khóa  $K_{AB}$  (hay khóa  $K_{UA}$  với mã khóa công khai) cho ra bản rõ  $P_T$  có ý nghĩa theo ý muốn của Trudy.
- KL2: Hơn nữa, Trudy cũng không thể tìm ra một bản mã  $C_T$  sao cho  $P_T$  là một bản tin có ý nghĩa, mà chỉ là một dãy bits lộn xộn, không cấu trúc.

Tuy nhiên trong thực tế có nhiều loại dữ liệu mà các bits gần như là ngẫu nhiên. Chẳng hạn như dữ liệu hình ảnh bitmap hay âm thanh. Ngoài ra đối với máy tính, việc nhận dạng ra thế nào là dãy bits có ý nghĩa là một công việc khó khăn. Do đó trong thực tế, chúng ta hầu như chấp nhận rằng bất cứ dãy bits nào cũng có thể có ý nghĩa. Lúc này các phương pháp mã hóa đối xứng và mã hóa công khai không thể bảo đảm tính chứng thực.

Để giải quyết vấn đề này, mã hóa phải vận dụng khái niệm *redundancy* của lĩnh vực truyền số liệu, tức thêm vào một ít dữ liệu (checksum) để biến bản tin, từ *dãy bits ngẫu nhiên*, trở thành *dãy bits có cấu trúc*.

Trong quá trình truyền số liệu, do tác động *nhiều* của môi trường, bản tin lúc đến đích có thể bị sai lệch so với bản tin ban đầu trước khi truyền. Để phát hiện nhiễu, một đoạn bits ngắn gọi là checksum được tính toán từ dãy bits của bản tin, và gắn vào sau bản tin để tạo redundancy, và được truyền cùng với bản tin đến đích.



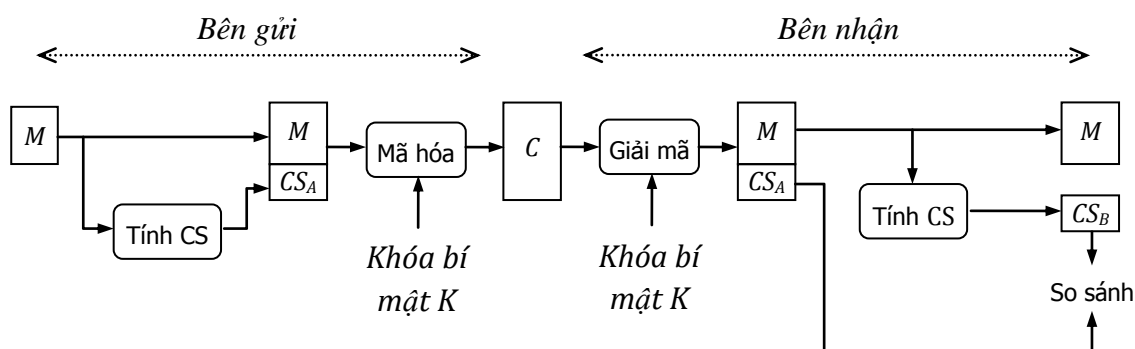
Chúng ta hãy xem xét lại một phương pháp kiểm lỗi checksum phổ biến là CRC (cyclic redundancy check). Trong phương pháp này, một đoạn bits ngắn được chọn làm số chia, lấy dãy bits của thông điệp chia cho số chia này, phần dư còn lại được gọi là giá trị checksum CRC. Phép chia này khác phép chia thường ở chỗ dùng phép XOR thay cho phép trừ. Giả sử thông điệp là 10101011 và số chia là 10011, quá trình tính như sau:

$$\begin{array}{r} \begin{array}{r} 10101011 \\ \oplus 10011 \\ \hline 11001 \\ \oplus 10011 \\ \hline 10101 \\ \oplus 10011 \\ \hline 110 \end{array} \quad \left| \begin{array}{r} 10011 \\ \hline 1011 \end{array} \right. \end{array}$$

Giá trị CRC là phần dư **0110** (ít hơn 1 bit so với số chia). Giá trị này được gửi kèm thông điệp đến người nhận. Người nhận cũng thực hiện phép tính CRC như vậy. Nếu giá trị CRC người nhận tính được trùng khớp với CRC của người gửi thì có nghĩa là thông điệp không bị lỗi trong quá trình truyền dữ liệu. Trong phương pháp CRC không khó để tìm ra hai dãy bit khác nhau mà *có cùng CRC*. Có nghĩa là có thể xảy ra lỗi mà không phát hiện được. Tuy nhiên xác suất ngẫu nhiên xảy ra lỗi trên đường truyền mà làm cho dãy bit truyền và dãy bit nhận có cùng giá trị CRC là rất thấp.

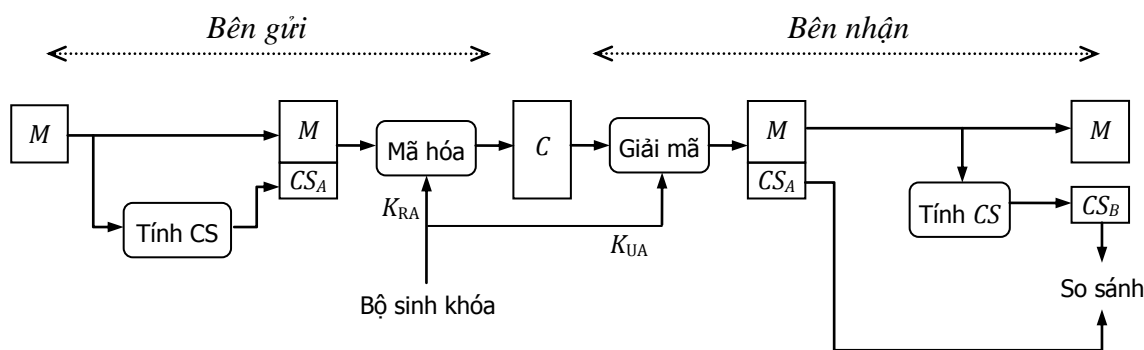
Nếu áp dụng cơ chế checksum vào chứng thực thông điệp, người gửi có thể tính checksum từ dãy bit của thông điệp, sau đó nối checksum này vào dãy bit đó. Như vậy chúng ta được một dãy bit có cấu trúc. Sau đó tiến hành mã hóa đối xứng hay mã hóa công khai trên dãy bit mới. Vì kích thước của checksum là ngắn nên cũng không ảnh hưởng lắm đến tốc độ mã hóa và băng thông sử dụng.

Cụ thể, mô hình mã hóa đối xứng bảo mật và chứng thực được sửa thành như sau:



**Hình 5-1. Mô hình chứng thực mã hóa đối xứng có dùng checksum**

Mô hình chứng thực bằng mã hóa khóa công khai được sửa thành:



**Hình 5-2. Mô hình chứng thực mã hóa khóa công khai có dùng checksum**

Nếu Trudy sửa bản mã  $C$ , thì bản giải mã của Bob, ký hiệu  $M_T$  và  $CS_T$ , sẽ mất đi tính cấu trúc. Nghĩa là checksum  $CS_B$  mà Bob tính được từ  $M_T$  không giống với  $CS_T$ . Và Bob biết được là bản tin bị thay đổi đường truyền. Nếu hàm checksum có độ phức tạp cao thì xác suất để  $CS_B = CS_T$  là rất thấp.

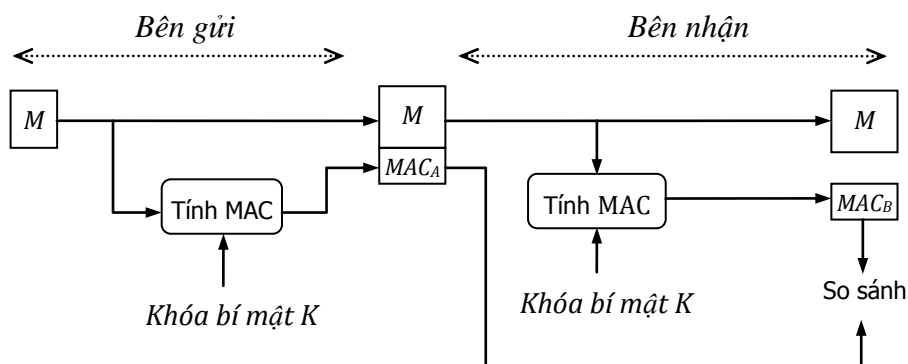
Ngoài ra còn có hai phương thức chứng thực thông điệp khác mà chúng ta sẽ tìm hiểu là mã chứng thực thông điệp MAC và hàm băm (Hash function).

## 5.1 Mã chứng thực thông điệp

Mã chứng thực thông điệp (MAC) có thể coi là một dạng checksum của mã hóa, được tính theo công thức  $MAC = C(M, K)$ , trong đó:

- 1)  $M$  là thông điệp cần tính  $MAC$
- 2)  $K$  là khóa bí mật được chia sẻ giữa người gửi và người nhận
- 3)  $C$  là hàm tính  $MAC$

Vì  $MAC$  có khóa  $K$  bí mật giữa người gửi và người nhận nên chỉ có người gửi và người nhận mới có thể tính được giá trị  $MAC$  tương ứng. Mô hình ứng dụng  $MAC$  để chứng thực thông điệp như sau:



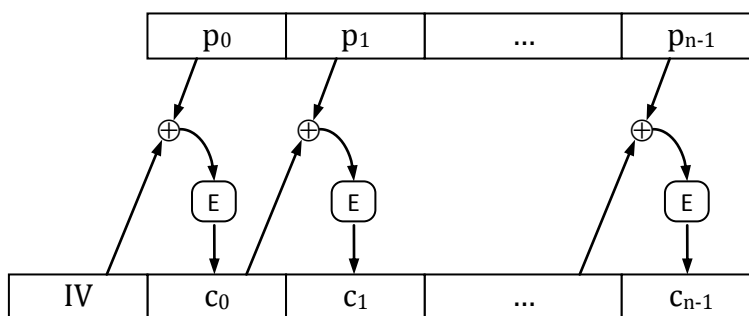
**Hình 5-3. Chứng thực dùng MAC**

Trudy, nếu chỉ sửa  $M$  thành  $M_T$  thì giá trị  $MAC_B$  sẽ khác  $MAC_A$  và Bob phát hiện được. Nếu Trudy muốn sửa thông điệp mà Bob không biết, thì cần sửa luôn  $MAC_A$  thành  $MAC_T$  tính được từ  $M_T$ . Tuy nhiên Trudy không biết khóa  $K$ , do đó không tính được  $MAC_T$  cần thiết.

Mô hình trên không đảm bảo tính bảo mật. Để có tính bảo mật,  $M$  và  $MAC_A$  cần được mã hóa trước khi truyền đi.

Trong phần đầu chương, ta đã thấy mã hóa đối xứng cũng có tính chứng thực, như vậy thì tại sao không dùng mã hóa đối xứng mà cần dùng MAC? Câu trả lời là trong một số trường hợp người ta không cần tính bảo mật mà chỉ cần tính chứng thực, nên sử dụng MAC tiết kiệm được thời gian xử lý hơn.

Trong thực tế, người ta hay dùng mô hình CBC và phương pháp DES của mã hóa đối xứng để tính giá trị MAC. Hình dưới đây trình bày lại mô hình CBC



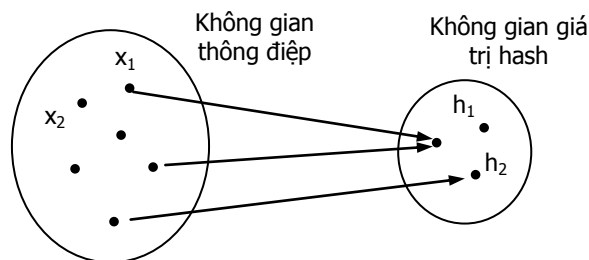
Như vậy thông điệp  $M$  sẽ được chia thành các khối  $(P_0, P_1, \dots, P_{n-1})$ , dùng thêm một vector khởi tạo  $IV$ , thì bản mã  $C_{n-1}$  được chọn làm giá trị  $MAC$  cho  $M$ . Như vậy kích thước của  $MAC$  là 64 bit, kích thước của khóa  $K$  là 56 bit.

## 5.2 Hàm băm – Hash function

Trong khi phương pháp checksum CRC cho phép hai dãy bit có cùng checksum, thì hàm băm  $H(x)$  là một hàm tính checksum mạnh thỏa mãn các yêu cầu sau:

- 1)  $H$  có thể áp dụng cho các thông điệp  $x$  với các độ dài khác nhau
- 2) Kích thước của output  $h = H(x)$  là cố định và nhỏ
- 3) Tính một chiều: với một  $h$  cho trước, không thể tìm lại được  $x$  sao cho  $h = H(x)$  (về mặt thời gian tính toán)
- 4) Tính chống trùng yếu: cho trước một  $x$ , không thể tìm  $y \neq x$  sao cho  $H(x) = H(y)$
- 5) Tính chống trùng mạnh: không thể tìm ra cặp  $x, y$  bất kỳ ( $x \neq y$ ) sao cho  $H(x) = H(y)$ , hay nói cách khác nếu  $H(x) = H(y)$  thì có thể chắc chắn rằng  $x = y$ .

Kích thước của input  $x$  là bất kỳ còn kích thước của  $h$  là nhỏ, ví dụ giả sử kích thước của  $x$  là 512 bit còn kích thước của  $h$  là 128 bit. Như vậy trung bình có khoảng  $2^{384}$  giá trị  $x$  mà có cùng giá trị  $h$ . Việc trùng là không thể loại bỏ. Tính chống trùng của hàm Hash là yêu cầu rằng việc tìm ra hai input  $x$  như vậy thì phải là rất khó về mặt thời gian tính toán.



**Hình 5-4. Ánh xạ giữa thông điệp và giá trị hash không phải là song ánh**

Lấy ví dụ với đối tượng con người. Xét hai hàm sau: hàm lấy khuôn mặt và hàm lấy dấu vân tay. Có thể thấy hàm lấy khuôn mặt không phải là hàm hash vì chúng có thể tìm ra 2 người giống nhau ở khuôn mặt. Còn hàm lấy dấu vân tay là hàm hash vì trên khắp thế giới không tìm ra hai người giống nhau về dấu vân tay.

Một yêu cầu nữa của hàm Hash là kích thước của output  $h$  không được quá lớn. Nếu kích thước  $h$  lớn thì dễ đạt được tính chống trùng tuy nhiên sẽ tốn dung lượng đường truyền trong mô hình Hình 5-1. Vậy kích thước của output  $h$  cần thiết là bao nhiêu để thực hiện chống trùng có hiệu quả? Chúng ta sẽ tìm hiểu vấn đề này qua một lý thuyết gọi là bài toán ngày sinh nhật.

### 5.2.1 Bài toán ngày sinh nhật

Bài toán 1: Giả sử trong phòng có 30 người. Vậy xác suất để có hai người có cùng ngày sinh là bao nhiêu phần trăm?

Nguyên lý chuồng bồ câu Dirichlet phát biểu rằng là cần có  $365+1 = 366$  người để tìm thấy hai người có cùng ngày sinh với xác suất 100% (để đơn giản, chúng ta bỏ qua năm nhuận). Do đó hầu hết chúng ta sẽ nghĩ rằng với 30 người thì xác suất hai người cùng



ngày sinh là nhỏ, chắc chắn nhỏ hơn 50%. Tuy nhiên nếu kiểm tra bằng toán học thì chỉ cần 23 người là đủ để xác suất lớn hơn 50%. Vì vậy bài toán này còn được gọi dưới tên *ngịch lý ngày sinh*. Ta có thể phát biểu lại bài toán và chứng minh như sau.

Giả sử trong phòng có  $M$  người. Hỏi  $M$  tối thiểu phải là bao nhiêu để tồn tại hai người có cùng ngày sinh với xác suất lớn hơn 50%?

Ta đánh số thứ tự của  $M$  người lần lượt là  $0, 1, 2, \dots, M-1$ . Xác suất để người thứ 1 khác ngày sinh với người thứ 0 là  $364/365$ . Tiếp theo, xác suất để người thứ 2 khác ngày sinh với người thứ 0 và thứ 1 là  $363/365$ . Tiếp tục như vậy đến người thứ  $M-1$  thì xác suất để người này khác ngày sinh với tất cả những người trước là  $(365-M+1)/365$ . Vậy xác suất để  $M$  người này đều có ngày sinh khác nhau là:

$$p(M) = \left(\frac{364}{365}\right) \left(\frac{363}{365}\right) \dots \left(\frac{365-M+1}{365}\right) = \left(1 - \frac{1}{365}\right) \left(1 - \frac{2}{365}\right) \dots \left(1 - \frac{M-1}{365}\right)$$

Xét hàm lũy thừa  $e^x$ , chúng ta đã biết một xấp xỉ của  $e^x$  khi  $x$  nhỏ là  $e^x \approx 1+x$ . Do đó  $p(M)$  có thể viết lại thành:

$$p(M) \approx e^{\frac{-1}{365}} \cdot e^{\frac{-2}{365}} \cdot e^{\frac{-3}{365}} \dots e^{\frac{-M+1}{365}} = e^{-\frac{1+2+3+\dots+(M-1)}{365}} = e^{-\frac{M(M-1)}{2 \times 365}}$$

Dẫn đến xác suất để tồn tại ít nhất hai người có ngày sinh giống nhau là

$$1 - p(M) \approx 1 - e^{-\frac{M(M-1)}{2 \times 365}}$$

Để xác suất này lớn hơn 50%, chúng ta cho biểu thức trên lớn hơn 0.5:

$$1 - e^{-\frac{M(M-1)}{2 \times 365}} \geq \frac{1}{2}$$

$$e^{-\frac{M(M-1)}{2 \times 365}} \leq \frac{1}{2}$$

$$M(M-1) \geq 2 \times 365 \times \log_e 2 \quad (*)$$

và giải bất đẳng thức, ta có được  $M \geq 23$ .

**Bài toán 2:** Giả sử bạn đang ở trong một căn phòng với  $M$  người khác. Hỏi  $M$  tối thiểu là bao nhiêu để tồn tại một người có cùng ngày sinh với bạn với xác suất lớn hơn 50%?

Xác suất để một người không có cùng ngày sinh với bạn là  $364/365$ . Như vậy xác suất để  $M$  người đều khác ngày sinh với bạn là  $(364/365)^M$ . Từ đó ta có xác suất để tồn tại ít nhất một người có cùng ngày sinh với bạn là:

$$1 - (364/365)^M$$

Để xác suất này lớn hơn 50% thì suy ra  $M \geq 253$ . Vậy tối thiểu phải có 253 người.

Áp dụng vấn đề ngày sinh nhật vào hàm băm, ta thấy rằng tính chống trùng mạnh giống bài toán 1, còn tính chống trùng yếu giống bài toán 2. Giả sử số bit của kết xuất  $h$  của hàm băm là  $n$  bit, như vậy số lượng giá trị có thể có của  $h$  là  $N = 2^n$ . Giả sử thêm rằng  $2^n$  giá trị băm này đều là ngẫu nhiên, có khả năng xuất hiện như nhau. Thay giá trị 365 của bất phương trình (\*) bằng  $2^n$

$$M(M-1) \geq 2 \times 2^n \times \log_e 2$$

Giải bất phương trình trên, ta có xấp xỉ  $M \geq \sqrt{2^n} = 2^{n/2}$

Giống như vấn đề ngày sinh nhật, kết quả trên cho thấy, đối với hàm băm chúng ta phải thử khoảng  $2^{n/2}$  thông điệp khác nhau để tìm ra hai thông điệp mà có cùng giá trị băm (xác suất lớn hơn 50%). Nếu  $n=128$  thì phải thử khoảng  $2^{64}$  thông điệp, một con số khá lớn, nghĩa là hàm băm này đạt được tính chống trùng mạnh. Do đó việc phá hàm băm cũng khó giống như là việc tấn công vét cạn khóa của mã hóa đối xứng DES.

Tóm lại có thể phát biểu tính chất chống trùng của hàm băm dưới dạng toán học như sau:

$$\nexists x \neq y \text{ sao cho } H(x) = H(y)$$

Nói cách khác:

$$\forall x, y \text{ nếu } H(x) = H(y) \text{ thì } x = y \quad (*)$$

Hai hàm băm được dùng phổ biến hiện nay là MD5 và SHA-1.

### 5.2.2 Hàm băm MD5 và SHA-1

MD5 được phát minh bởi Ron Rivest, người cũng đã tham gia xây dựng RSA. MD5, viết tắt từ chữ ‘Message Digest’, được phát triển lên từ MD4 và trước đó là MD2, do MD2 và MD4 không còn được xem là an toàn. Kích thước giá trị băm của MD5 là 128 bit, mà chúng ta coi như là an toàn (theo nghĩa không tìm được 2 thông điệp có cùng giá trị băm). Tuy nhiên vào năm 1994 và 1998, một phương pháp tấn công MD5 đã được tìm thấy và một số thông điệp có cùng giá trị băm MD5 được chỉ ra (vi phạm tính chống trùng mạnh). Tuy vậy ngày nay MD5 vẫn còn được sử dụng phổ biến.

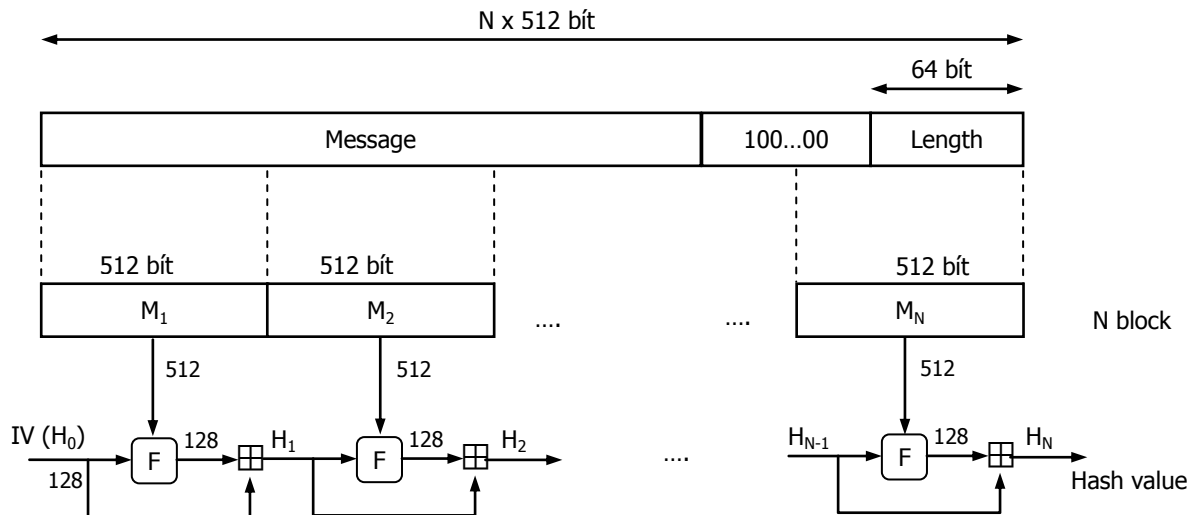
Vì MD5 không còn được xem là an toàn, nên người ta đã xây dựng thuật toán băm khác. Một trong những thuật toán đó là SHA-1 (Secure Hash Algorithm) mà đã được chính phủ Mỹ chọn làm chuẩn quốc gia. SHA-1 có kích thước giá trị băm là 160 bit. Ngày nay còn có ba phiên bản khác của SHA là SHA-256, SHA-384, SHA-512 mà có kích thước giá trị băm tương ứng là 256, 384 và 512 bit.

Tương tự như mã hóa đối xứng, các hàm băm mạnh đều có hiệu ứng lan truyền (*avalanche effect*). Chỉ cần thay đổi 1 bit trong thông điệp đầu vào thì  $\frac{1}{2}$  các bit của giá trị băm sẽ thay đổi theo. Điều này làm cho người phá hàm băm không thể thử sai theo kiểu chosen-plaintext, nghĩa là không tồn tại cách tấn công nào khác được và buộc phải thử vét cạn  $2^{n/2}$  thông điệp khác nhau, mà chúng ta đã chứng minh là bất khả thi về mặt thời gian.

#### a) MD5

Sau đây chúng ta sẽ tìm hiểu hàm băm MD5 với kích thước giá trị băm là 128 bit, được dùng để tính giá trị băm của thông điệp có kích thước tối đa là  $2^{64}$  bit.

Sơ đồ tổng thể:



Trước tiên thông điệp được thêm dãy bit padding 100...00. Sau đó thêm vào chiều dài (trước khi padding) của thông điệp được biểu diễn bằng 64 bit. Như vậy chiều dài của dãy bit padding được chọn sao cho cuối cùng thông điệp có thể chia thành  $N$  block 512 bit  $M_1, M_2, \dots, M_N$ .

Quá trình tính giá trị băm của thông điệp là quá trình lũy tiến. Trước tiên block  $M_1$  kết hợp với giá trị khởi tạo  $H_0$  thông qua hàm  $F$  để tính giá trị hash  $H_1$ . Sau đó block  $M_2$  được kết hợp với  $H_1$  để cho ra giá trị hash là  $H_2$ . Block  $M_3$  kết hợp với  $H_2$  cho ra giá trị  $H_3$ . Cứ như vậy cho đến block  $M_N$  thì ta có giá trị băm của toàn bộ thông điệp là  $H_N$ .

$H_0$  là một dãy 128 bit được chia thành 4 từ 32 bit, ký hiệu 4 từ 32 bit trên là abcd. a, b, c, d là các hằng số như sau (viết dưới dạng thập lục phân):

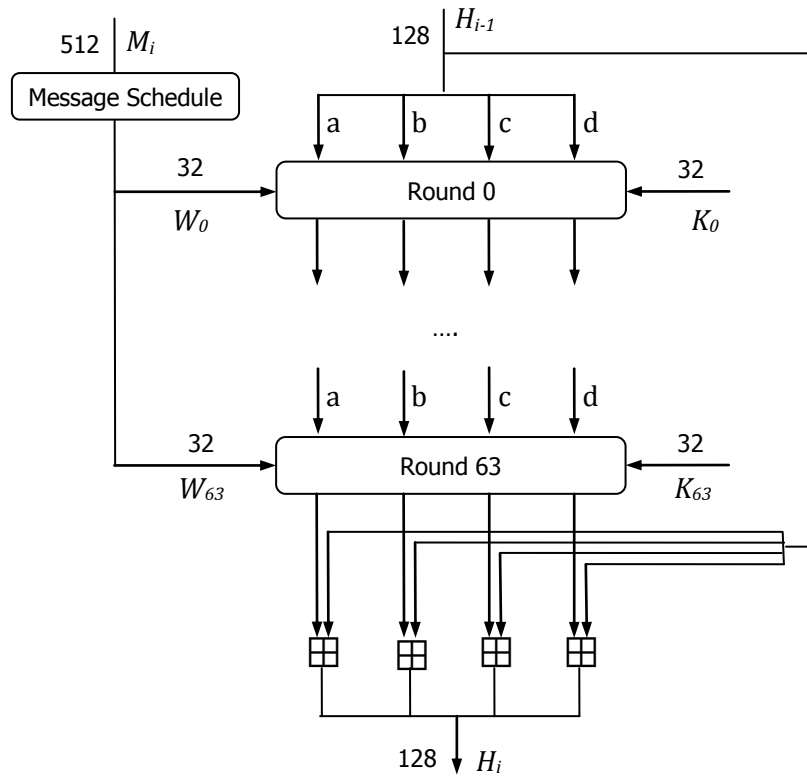
a = 01234567

b = 89abcdef

c = fedbca98

d = 76543210

Tiếp theo ta sẽ tìm hiểu cấu trúc của hàm  $F$ .



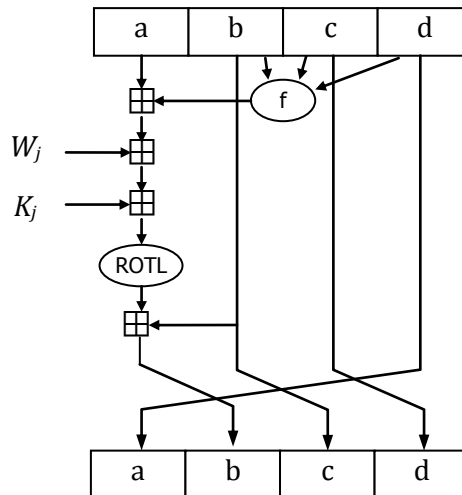
Tại mỗi bước lũy tiến, các giá trị abcd của giá trị hash  $H_{i-1}$  được biến đổi qua 64 vòng từ 0 đến 63. Tại vòng thứ  $j$  sẽ có 2 tham số là  $K_j$  và  $W_j$  đều có kích thước 32 bit. Các hằng số  $K_j$  được tính từ công thức:

$K_j$  là phần nguyên của số  $2^{32} \text{abs}(\sin(i))$  với  $i$  biểu diễn theo radian.

Giá trị block  $M_i$  512 bit được biến đổi qua một hàm message schedule cho ra 64 giá trị  $W_0, W_1, \dots, W_{63}$  mỗi giá trị 32 bit. Block  $M_i$  512 bit được chia thành 16 block 32 bit ứng với các giá trị  $W_0, W_1, \dots, W_{15}$  ( $16 \times 32 = 512$ ). Tiếp theo, 16 giá trị này được lặp lại 3 lần tạo thành dãy 64 giá trị.

Sau vòng cuối cùng, các giá trị abcde được cộng với các giá trị abcd của  $H_{i-1}$  để cho ra các giá trị abcd của  $H_i$ . Phép cộng ở đây là phép cộng modulo  $2^{32}$ .

Tiếp theo ta tìm hiểu cấu trúc của một vòng. Việc biến đổi các giá trị abcd trong vòng thứ  $i$  được thể hiện trong hình bên dưới.



Ở đây  $b \rightarrow c, c \rightarrow d, d \rightarrow a$ . Giá trị  $b$  được tính qua hàm:

$$t = a + f(b, c, d) + W_i + K_i$$

$$b = b + ROTL(t, s)$$

Trong đó:

- Hàm  $f(x, y, z)$ :

$$f(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$$

nếu là vòng 0 đến 15

$$f(x, y, z) = (z \wedge x) \vee (\neg z \wedge y)$$

nếu là vòng 16 đến 31

$$f(x, y, z) = x \oplus y \oplus z$$

nếu là vòng 32 đến 48

$$f(x, y, z) = y \oplus (x \vee \neg z)$$

nếu là vòng 49 đến 63

- Hàm  $ROTL(t, s)$ :  $t$  được dịch vòng trái  $s$  bit, với  $s$  là các hằng số cho vòng thứ  $i$  như sau:

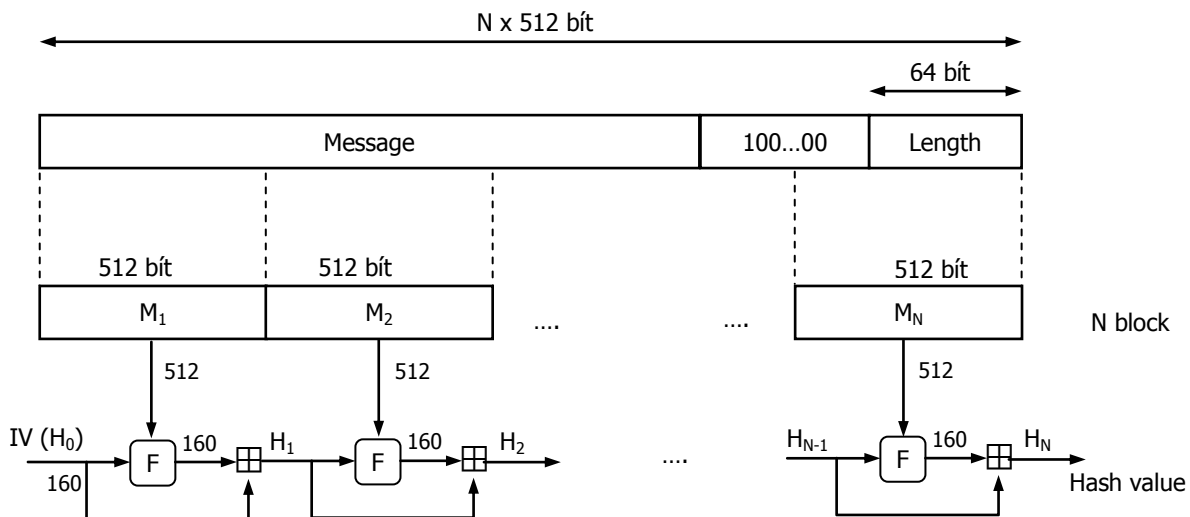
$i$	$s$
0, 4, 8, 12	7
1, 5, 9, 13	12
2, 6, 10, 14	17
3, 7, 11, 15	22
16, 20, 24, 28	5
17, 21, 25, 29	9
18, 22, 26, 30	14
19, 23, 27, 31	20
32, 36, 40, 44	4
33, 37, 41, 45	11
34, 38, 42, 46	16
35, 39, 43, 47	23
48, 52, 56, 60	6
49, 53, 57, 61	10
50, 54, 58, 62	15
51, 55, 59, 63	21

- Phép  $+$  (hay  $\boxplus$ ) là phép cộng modulo  $2^{32}$

b) SHA-1

Hàm băm SHA-1 với giá trị băm có kích thước là 160 bit, được dùng để tính giá trị băm của thông điệp có kích thước tối đa là  $2^{64}$  bit.

Sơ đồ tổng thể của SHA1 cũng giống như của MD5, chỉ có điểm khác là kích thước của giá trị hash tại mỗi bước là 160 bit.



$H_0$  là một dãy 160 bit được chia thành 5 từ 32 bit, ký hiệu 5 từ 32 bit trên là abcde. a, b, c, d, e là các hằng số như sau:

a = 67452301

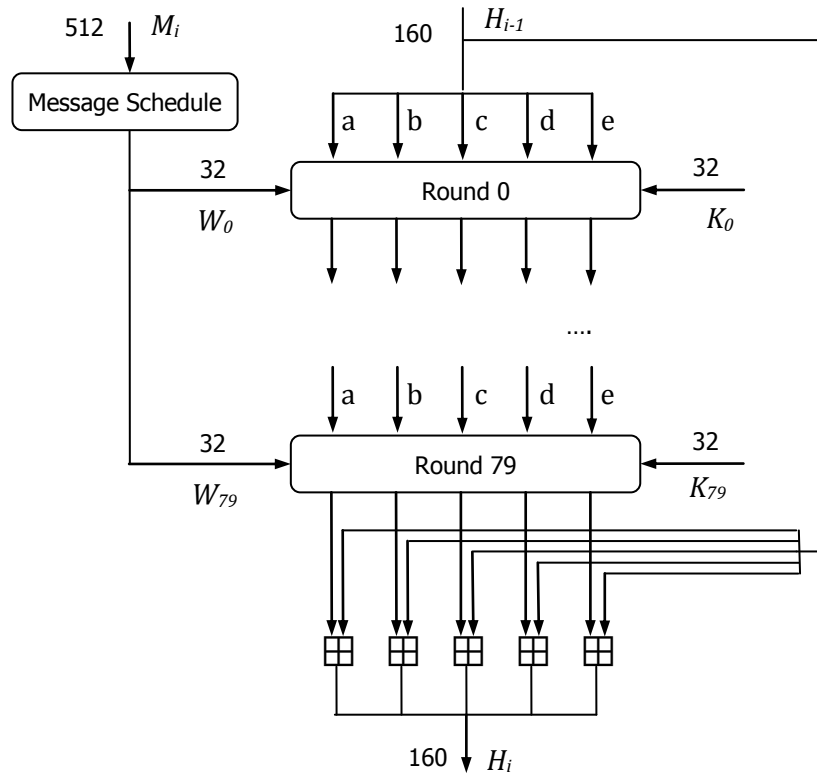
b = efcdab89

c = 98badcfe

d = 10325476

e = c3d2e1f0

Cấu trúc của hàm  $F$  của SHA cũng tương tự như MD5, tuy nhiên được thực hiện trên 80 vòng.



Giá trị  $K_0, K_1, \dots, K_{79}$  là các hằng số sau:

$$\begin{aligned}
 K_i &= 5A827999 & \text{với } 0 \leq i \leq 19 \\
 K_i &= 6ED9EBA1 & \text{với } 20 \leq i \leq 39 \\
 K_i &= 8F1BBCDC & \text{với } 40 \leq i \leq 59 \\
 K_i &= CA62C1D6 & \text{với } 60 \leq i \leq 79
 \end{aligned}$$

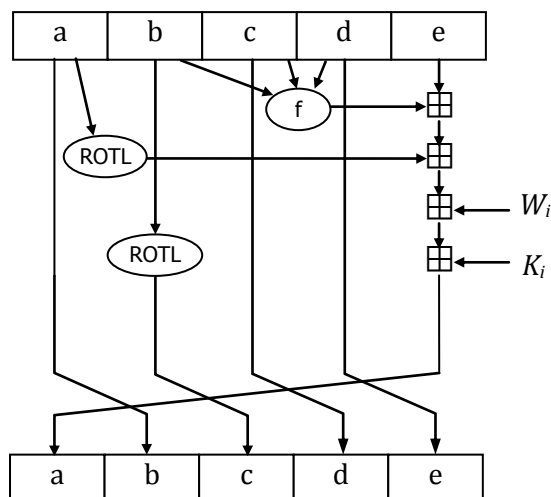
Giá trị block  $M_i$  512 bit được biến đổi qua một hàm message schedule cho ra 80 giá trị  $W_0, W_1, \dots, W_{80}$  mỗi giá trị 32 bit, theo quy tắc:

Trước tiên block  $M_i$  512 bit được chia thành 16 block 32 bit ứng với các giá trị  $W_0, W_1, \dots, W_{15}$  ( $16 \times 32 = 512$ ).

Các giá trị  $W_t$  ( $16 \leq t \leq 79$ ) được tính theo công thức:

$$W_t = ROTL(W_{t-3} + W_{t-8} + W_{t-14} + W_{t-16}, 1) \quad \text{với phép cộng modulo } 2^{32}.$$

Việc biến đổi các giá trị abcde trong vòng thứ  $i$  được thể hiện trong hình bên dưới.



Ở đây  $a \rightarrow b, c \rightarrow d, d \rightarrow e$ . Giá trị  $a$  và  $c$  được tính qua các hàm:

$$a = ROTL(a, 5) + f(b, c, d) + e + W_i + K_i$$

$$c = ROTL(b, 30)$$

Trong đó, hàm  $f(x, y, z)$ :

$$f(x, y, z) = Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \quad \text{nếu là vòng 0 đến 19}$$

$$f(x, y, z) = Parity(x, y, z) = x \oplus y \oplus z \quad \text{nếu là vòng 20 đến 39}$$

$$f(x, y, z) = Maj(x, y, z) = (x \wedge y) \oplus (y \wedge z) \oplus (z \wedge x) \quad \text{nếu là vòng 40 đến 59}$$

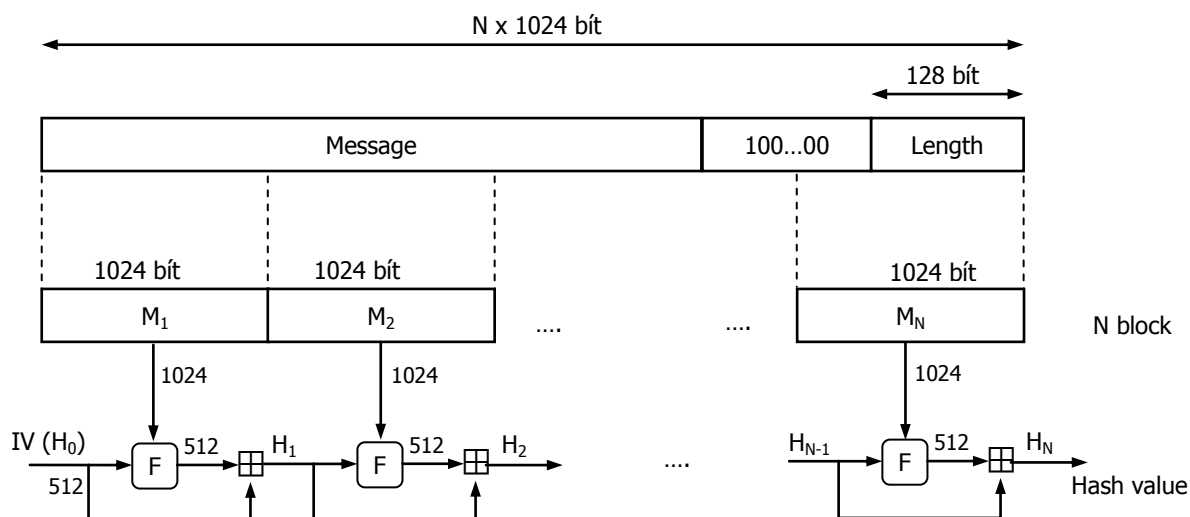
$$f(x, y, z) = Parity(x, y, z) = x \oplus y \oplus z \quad \text{nếu là vòng 60 đến 79}$$

Ý nghĩa của hàm Maj và hàm Ch:

- Hàm Maj: giả sử  $x_i, y_i, z_i$  là bit thứ  $i$  của  $x, y, z$ , thì bit thứ  $i$  của hàm Maj là giá trị nào chiếm đa số, 0 hay 1 (giống như hàm maj được định nghĩa trong phần thuật toán A5/1).
- Hàm Ch: bit thứ  $i$  của hàm Ch là phép chọn: if  $x_i$  then  $y_i$  else  $z_i$ .

### c) SHA-512

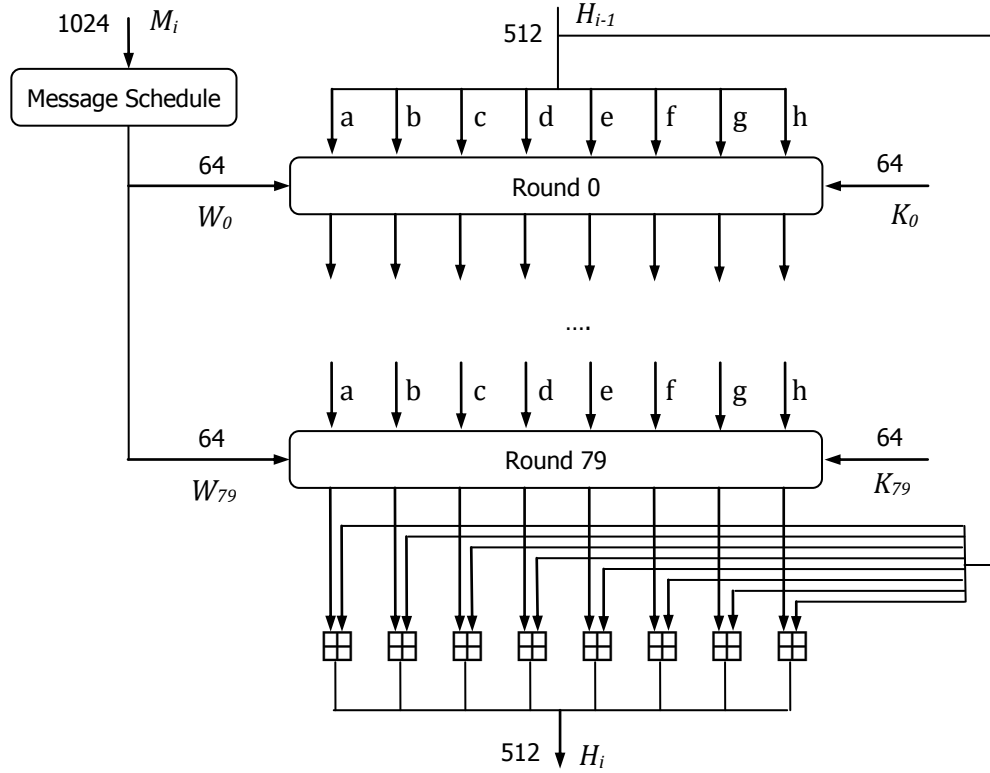
Phương pháp SHA-512 có cấu trúc cũng gần giống như SHA-1, tuy nhiên các khối tính toán có số bit lớn hơn. Bên dưới là sơ đồ tổng thể của SHA-512





Thông điệp được padding có thể chia thành các khối 1024 bit. Giá trị hash tại mỗi bước có kích thước 512 bit.  $H_0$  được chia thành 8 từ 64 bit abcdefgh. a, b, c, d, e, f, g, h được lấy từ phần thập phân của căn bậc 2 của 8 số nguyên tố đầu tiên (ví dụ a có giá trị hexa là 6A09E667F3BCC908).

Cấu trúc của hàm  $F$  cũng giống như hàm  $F$  của SHA-1.



Hai tham số là  $K_i$  và  $W_i$  đều có kích thước 64 bit. Giá trị  $K_0, K_1, \dots, K_{80}$  được lấy từ phần thập phân của căn bậc 3 của 80 số nguyên tố đầu tiên. Còn  $W_0, W_1, \dots, W_{79}$  được tính từ  $M_i$  như sau:

Trước tiên block  $M_i$  1024 bit được chia thành 16 block 64 bit ứng với các giá trị  $W_0, W_1, \dots, W_{15}$  ( $16 \times 64 = 1024$ ).

Các giá trị  $W_t$  ( $16 \leq t \leq 79$ ) được tính theo công thức:

$$W_t = W_{t-16} + \delta_0(W_{t-15}) + W_{t-7} + \delta_1(W_{t-2})$$

Với:

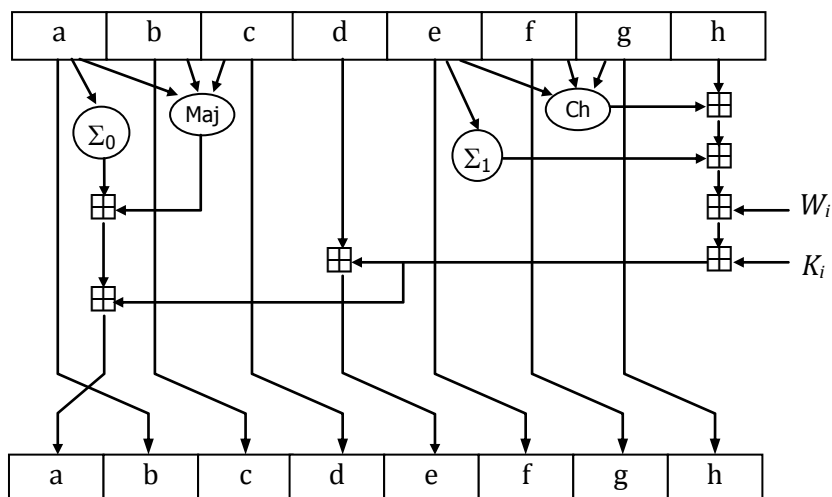
$$\delta_0(x) = \text{ROTR}(x, 1) \oplus \text{ROTR}(x, 8) \oplus \text{SHR}(x, 7)$$

$$\delta_1(x) = \text{ROTR}(x, 19) \oplus \text{ROTR}(x, 61) \oplus \text{SHR}(x, 6)$$

Trong đó:

- $\text{SHR}(x, i)$ : là hàm dịch phải  $i$  bit của một số  $x$  64 bit
- $\text{ROTR}(x, i)$ : là hàm dịch vòng phải  $i$  bit của một số  $x$  64 bit
- Phép cộng là phép modulo  $2^{64}$

Cấu trúc của một vòng:



Ở đây  $g \rightarrow h, f \rightarrow g, e \rightarrow f, c \rightarrow d, b \rightarrow c, a \rightarrow b$ . Giá trị  $a$  và  $e$  được tính qua các hàm:

$$T_0 = \Sigma_0(a) + Maj(a, b, c)$$

$$T_1 = h + Ch(e, f, g) + \Sigma_1(e) + W_i + K_i$$

$$a = T_0 + T_1$$

$$e = d + T_1$$

Trong đó, hàm  $\Sigma_0(a)$  và  $\Sigma_1(e)$ :

$$\Sigma_0(a) = \text{ROTR}(a, 28) \oplus \text{ROTR}(a, 34) \oplus \text{ROTR}(a, 39)$$

$$\Sigma_1(e) = \text{ROTR}(e, 14) \oplus \text{ROTR}(e, 18) \oplus \text{ROTR}(e, 41)$$

### 5.2.3 HMAC

Hàm băm cũng có thể dùng để tính MAC bằng cách truyền thêm khóa bí mật  $K$  vào hàm băm. Lúc này, giá trị kết xuất được gọi là HMAC.

$$HMAC = H(M||K)$$

## 5.3 Một số ứng dụng của hàm băm

### 5.3.1 Lưu trữ mật khẩu

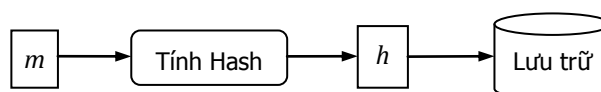
Hầu hết các ứng dụng phần mềm ngày nay, dù trên máy đơn hay trên web, đều có chứng thực người sử dụng. Nghĩa là để sử dụng ứng dụng, người sử dụng phải qua một cơ chế chứng thực username và mật khẩu, và từ đó được cung cấp các quyền sử dụng phần mềm khác nhau. Do đó vấn đề bảo mật mật khẩu là vấn đề quan trọng đối với mọi phần mềm.

Mật khẩu người sử dụng thường gồm các chữ cái thường và hoa, cộng thêm các chữ số. Giả sử mật khẩu được lưu trữ dưới dạng thường, không mã hóa, tại một nơi nào đó trên máy tính cá nhân hay máy chủ, trong một file dữ liệu hay trong hệ quản trị cơ sở dữ liệu. Như vậy sẽ xuất hiện một nguy cơ là có một người khác, hoặc là người quản trị administrator, hoặc là hacker, có thể mở được file dữ liệu hoặc cơ sở dữ liệu, và xem trộm được mật khẩu. Như vậy mật khẩu không thể được giữ bí mật tuyệt đối.

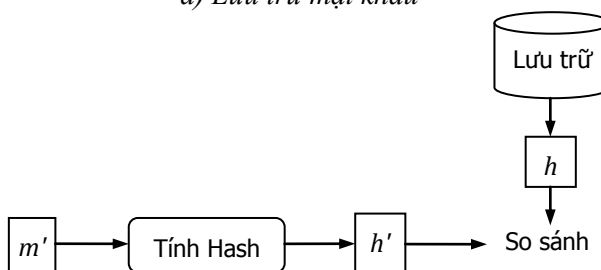
Một phương pháp để bảo vệ mật khẩu là dùng mã hóa, chương trình phần mềm sẽ dùng một khóa bí mật để mã hóa mật khẩu trước khi lưu mật khẩu xuống file hay cơ sở dữ liệu. Do đó tránh được vấn đề xem trộm mật khẩu. Tuy nhiên phương pháp này có yếu

điểm là lại phải lo bảo vệ khóa bí mật này. Nếu khóa bí mật bị lộ thì việc mã hóa không còn ý nghĩa.

Phương pháp bảo vệ mật khẩu hiệu quả nhất là dùng hàm băm. Khi người sử dụng đăng ký mật khẩu, giá trị băm của mật khẩu được tính bằng một hàm băm nào đó (MD5 hay SHA-1,...) Giá trị băm được lưu trữ vào file hay cơ sở dữ liệu. Vì hàm băm là một chiều, nên dù biết được giá trị băm và loại hàm băm, hacker cũng không thể suy ra được mật khẩu. Khi người sử dụng đăng nhập, mật khẩu đăng nhập được tính giá trị băm và so sánh với giá trị băm đang được lưu trữ. Do tính chống trùng, chỉ có một mật khẩu duy nhất có giá trị băm tương ứng, nên không ai khác ngoài người sử dụng có mật khẩu đó mới có thể đăng nhập ứng dụng.



a) Lưu trữ mật khẩu



b) Chứng thực mật khẩu, theo tính chống trùng, nếu  $h' = h$  thì  $m' = m$

**Hình 5-5. Dùng hàm Hash để lưu trữ mật khẩu**

	username	password	email
1	admin	nhx64312	nguyen@yahoo.com
2	devil	kin32xz	nam@hotmail.com
3	vampire	62ntt34	hung@gmail.com

Lưu trữ password không mã hóa

	username	password	Salt	email
1	admin	23dacd8cd768c95ad2e63cdc399c0535	64f84a9bdec1999e43c97ab12f8d9a36	nguyen@yahoo.com
2	devil	d400c472ab7a09ba87bf5c9715bbe118	66436db921568ae452f1e76a44e40aac	nam@hotmail.com
3	vampire	0d7b12ce9cf7ef3534aa5fee204eb0f5	4c798886f04910bad5a85fbec1c4bfea	hung@gmail.com

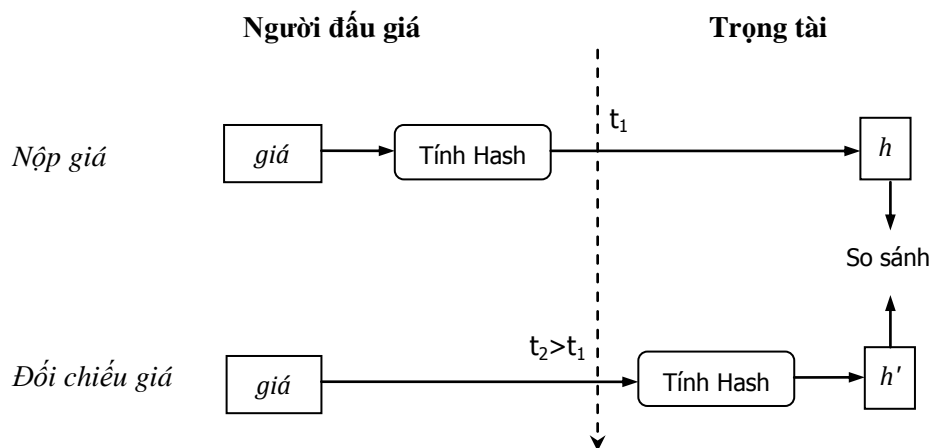
Lưu trữ password mã hóa bằng hàm hash MD5

### 5.3.2 Đấu giá trực tuyến

Phương pháp lưu trữ mật khẩu bằng giá trị Hash cũng được áp dụng tương tự cho việc đấu giá trực tuyến bằng hình thức đấu giá bí mật. Giả sử Alice, Bob và Trudy cùng tham gia đấu giá, họ sẽ cung cấp mức giá của mình cho trọng tài. Các mức giá này được giữ bí mật cho đến khi cả ba đều nộp xong. Nếu ai là người đưa ra mức giá cao nhất thì thắng thầu. Điểm quan trọng của phương pháp đấu giá này là giá của Alice, Bob, và Trudy phải được giữ bí mật trước khi công bố. Giả sử mức giá của Alice là 100, mức giá của Bob

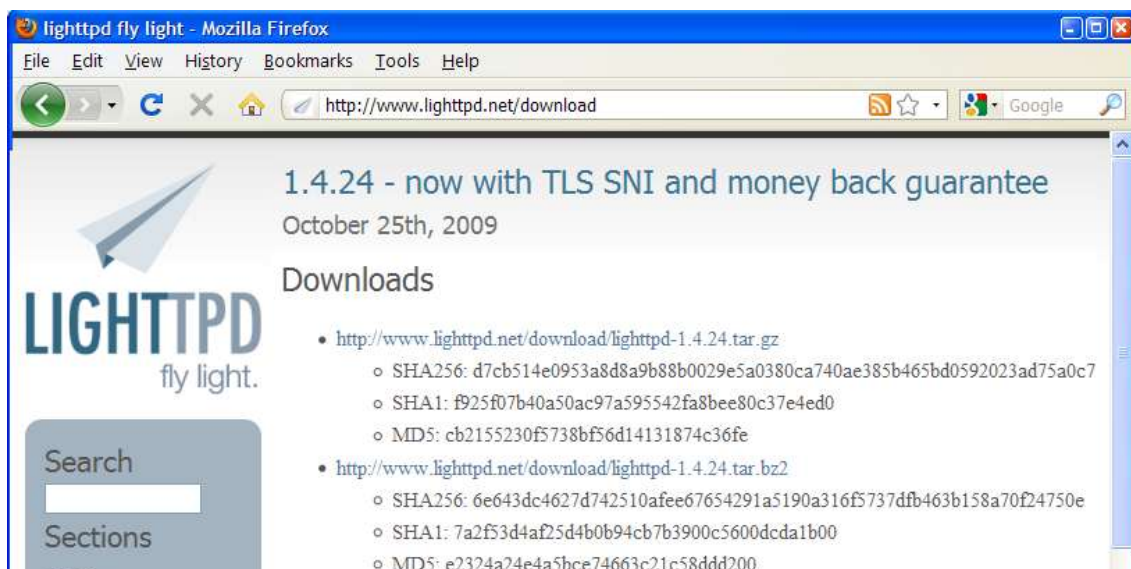
là 110, nếu Trudy thông đồng với trọng tài và biết được giá của Alice và Bob, Trudy có thể đưa ra mức giá 111 và thắng thầu.

Có thể tránh những hình thức lừa đảo như vậy bằng cách sử dụng hàm băm. Từ mức giá bỏ thầu, Alice và Bob sẽ tính các giá trị băm tương ứng và chỉ cung cấp cho trọng tài các giá trị băm này. Vì hàm băm là một chiều, nếu trọng tài và Trudy bắt tay nhau thì cũng không thể biết được giá của Alice và Bob là bao nhiêu. Đến khi công bố, Alice, Bob và Trudy sẽ đưa ra mức giá của mình. Trọng tài sẽ tính các giá trị băm tương ứng và so sánh với các giá trị băm đã nộp để bảo đảm rằng mức giá mà Alice, Bob và Trudy là đúng với ý định ban đầu của họ. Vì tính chống trùng của hàm băm nên Alice, Bob và Trudy không thể thay đổi giá so với ý định ban đầu.



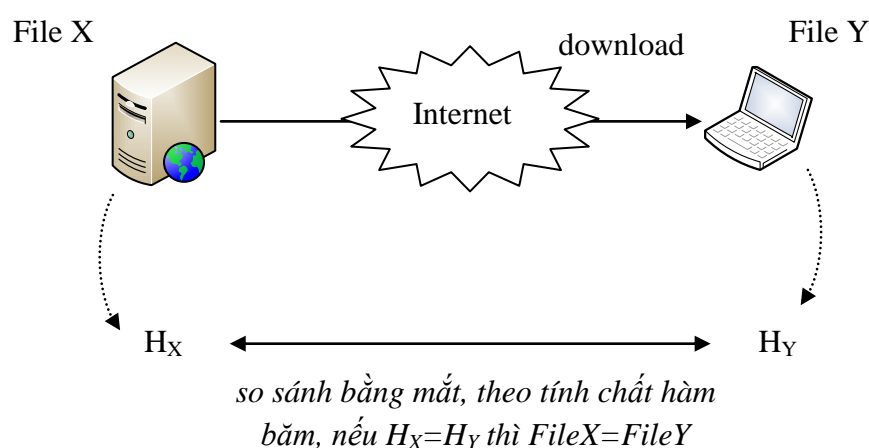
**Hình 5-6. Đấu giá bí mật**

### 5.3.3 Download file



Khi chúng ta download file từ mạng internet, nếu chất lượng mạng không tốt thì có thể xảy ra lỗi trong quá trình download làm cho file tại máy client khác với file trên server. Hàm băm có thể giúp chúng ta phát hiện ra những trường hợp bị lỗi như vậy.

Gọi file cần download trên server là  $X$ , và giá trị hash theo MD5 của file  $X$  mà server đã tính sẵn và cung cấp trên trang web là  $H_X$  (có thể xem bằng mắt). Gọi  $Y$  là file mà người sử dụng download được tại máy. Người sử dụng sẽ tính giá trị MD5  $H_Y$  cho file  $Y$ . Như vậy nếu  $H_X = H_Y$  thì theo tính chống trùng của hàm hash, file  $Y$  hoàn toàn giống file  $X$  và quá trình download không xảy ra lỗi.

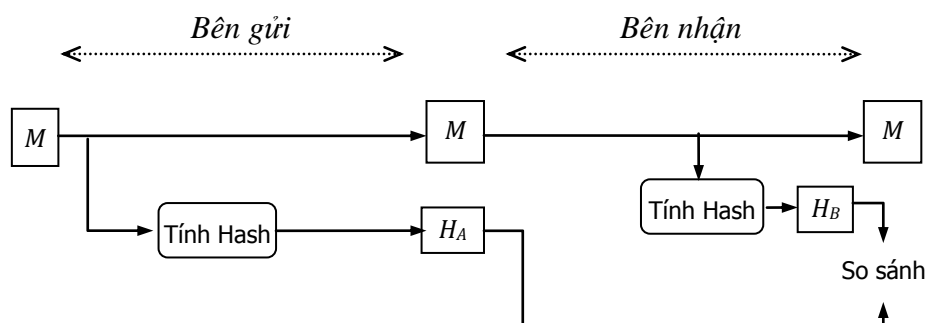


## 5.4 Hàm băm và chữ ký điện tử

Trong phần này chúng ta tìm hiểu cách thức ứng dụng hàm băm vào vấn đề chứng thực mà ta gọi là chữ ký điện tử.

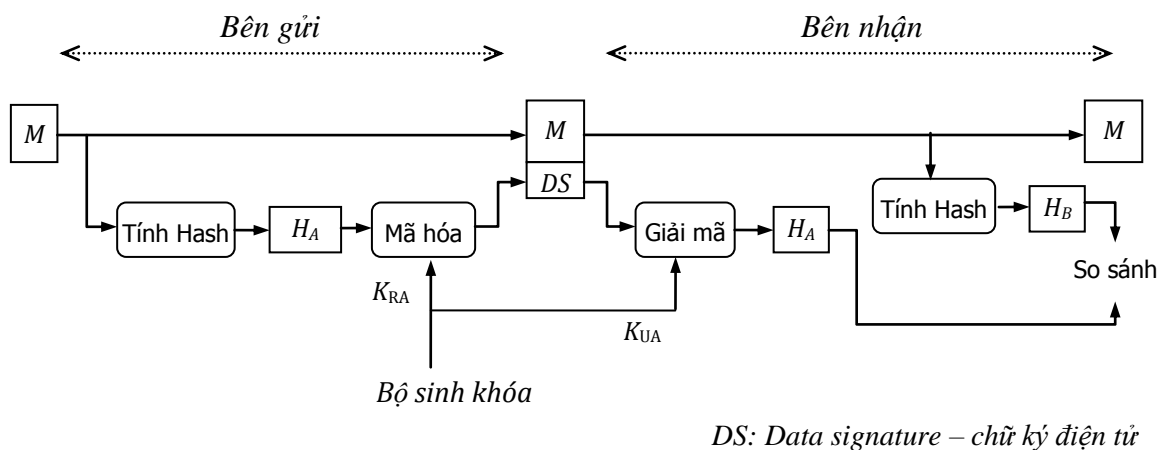
Việc sử dụng khóa bí mật chung cho người gửi và người nhận trong mã chứng thực thông điệp MAC sẽ gặp phải vấn đề tính không từ chối tương tự như mã hóa đối xứng. Dùng hàm băm và mã hóa khóa công khai khắc phục được vấn đề này.

Trước tiên xét một mô hình đơn giản:



Trong mô hình này Alice tính giá trị băm của thông điệp cần gửi và gửi kèm cho Bob. Bob tính lại giá trị băm của thông điệp nhận được và so sánh với giá trị băm của Alice. Tương tự như vấn đề download file, nếu Trudy sửa thông điệp  $M$  thì  $H_B \neq H_A$  và Bob sẽ phát hiện.

Tuy nhiên, Trudy cũng có thể sửa luôn giá trị băm  $H_A$  do Alice gửi và Bob không thể phát hiện. Để tránh vấn đề này cần sử dụng mã hóa khóa công khai để chứng thực  $H_A$  theo mô hình sau:



**Hình 5-6. Mô hình chữ ký điện tử**

Trong mô hình này, Alice sau khi tính giá trị hash  $H_A$  cho thông điệp  $M$  thì sẽ mã hóa  $H_A$  bằng khóa riêng của Alice để tạo thành chữ ký điện tử  $DS$ . Alice gửi kèm  $DS$  theo  $M$  cho Bob. Bob dùng khóa công khai của Alice để giải mã chữ ký điện tử  $DS$  và có được giá trị hash  $H_A$  của Alice. Vì Trudy không có  $K_{RA}$  nên không thể sửa được  $H_A$ .

Ngoài ra, vì Alice là người duy nhất có  $K_{RA}$ , nên chỉ có Alice mới có thể tạo  $DS$  từ  $M$ . Do đó Alice không thể từ chối là đã gửi bản tin.

Vậy dùng chữ ký điện tử thì có ưu điểm gì hơn so với cách dùng checksum trong mô hình ở hình 5-2? Chữ ký điện tử chỉ cần mã hóa giá trị hash mà không cần mã hóa toàn bộ thông điệp  $M$ . Vì phương pháp mã hóa khóa công khai tốn kém thời gian nên nếu  $M$  là một thông điệp dài, thì việc không mã hóa  $M$  giúp tiết kiệm được nhiều thời gian.

## 5.5 Câu hỏi ôn tập

1. Để bảo đảm tính chứng thực dùng mã hóa đối xứng hay mã hóa khóa công khai, bản rõ phải có tính chất gì? Tại sao?
2. Nếu bản rõ là một dãy bit ngẫu nhiên, cần làm gì để bản rõ trở thành có cấu trúc?
3. Sử dụng MAC để chứng thực có ưu điểm gì so với chứng thực bằng mã hóa đối xứng?
4. Về mặt lý thuyết, giá trị Hash có thể trùng không? Vậy tại sao nói giá trị Hash có thể xem là “dấu vân tay của thông điệp”?
5. Tại sao để chứng thực một thông điệp  $M$ , người ta chỉ cần mã hóa khóa công khai giá trị Hash của  $M$  là đủ? Thực hiện như vậy có lợi ích gì hơn so với cách thức mã hóa toàn bộ  $M$ .

## 5.6 Bài tập

1. Với số chia trong phép tính checksum CRC là 11001, bạn hãy tìm một số mà có CRC giống với số 11101101.
2. Hãy xem xét hàm hash sau. Thông điệp có dạng là một dãy các số thập phân  $M = (a_1, a_2, \dots, a_n)$ . Hàm hash được tính bằng công thức:  $h = (\sum_{i=1}^n a_i) \bmod n$ . Hàm hash trên có thỏa mãn các tính chất của một hàm hash như đã nêu trong phần 5.2 hay không? Giải thích lý do.
3. Thực hiện tương tự câu 2 với hàm hash  $h = (\sum_{i=1}^n a_i^2) \bmod n$
4. Giả sử Alice và Bob muốn tung đồng xu qua mạng (Alice tung và Bob đoán). Giao thức thực hiện như sau:
  - i. Alice chọn giá trị  $X=0$  hay 1.
  - ii. Alice sinh một khóa  $K$  ngẫu nhiên gồm 256 bit
  - iii. Dùng AES, Alice tính  $Y = E(X||R, K)$  trong đó  $R$  gồm 255 bit bất kỳ
  - iv. Alice gửi  $Y$  cho Bob
  - v. Bob đoán  $Z$  là 0 hay 1 và gửi  $Z$  cho Alice
  - vi. Alice gửi khóa  $K$  cho Bob để Bob tính  $X||R = D(Y, K)$
  - vii. Nếu  $X=Z$ , Bob đoán trúng. Nếu không Bob đoán sai.

Chúng tôi chứng tỏ rằng Alice có thể lừa Bob (chẳng hạn, Alice chọn  $X=1$ , thấy Bob đoán  $Z=1$  thì Alice sẽ lừa như thế nào để Bob giải mã  $Y$  thì có  $X=0$ ). Dùng hàm hash, hãy sửa đoạn giao thức trên để Alice không thể lừa được.

## 5.7 Bài tập thực hành

1. Tìm hiểu về phương pháp sử dụng hàm hash MD5 và SHA trong thư viện mã hóa của .NET. Áp dụng viết chương trình mã hóa password lưu trữ và kiểm tra password như đã trình bày trong phần 5.3.1
2. Gần đây, người ta đã phát hiện điểm yếu của hàm hash MD5, tức tìm ra hai thông điệp có cùng giá trị hash MD5. Bạn hãy tìm những bit khác nhau của 2 thông điệp bên dưới và dùng thư viện của .NET hoặc Java để tính giá trị hash MD5 của chúng.

Thông điệp 1 (dạng số thập lục phân):

```
d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c
2f ca b5 87 12 46 7e ab 40 04 58 3e b8 fb 7f 89
55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 71 41 5a
08 51 25 e8 f7 cd c9 9f d9 1d bd f2 80 37 3c 5b
96 0b 1d d1 dc 41 7b 9c e4 d8 97 f4 5a 65 55 d5
35 73 9a c7 f0 eb fd 0c 30 29 f1 66 d1 09 b1 8f
75 27 7f 79 30 d5 5c eb 22 e8 ad ba 79 cc 15 5c
ed 74 cb dd 5f c5 d3 6d b1 9b 0a d8 35 cc a7 e3
```

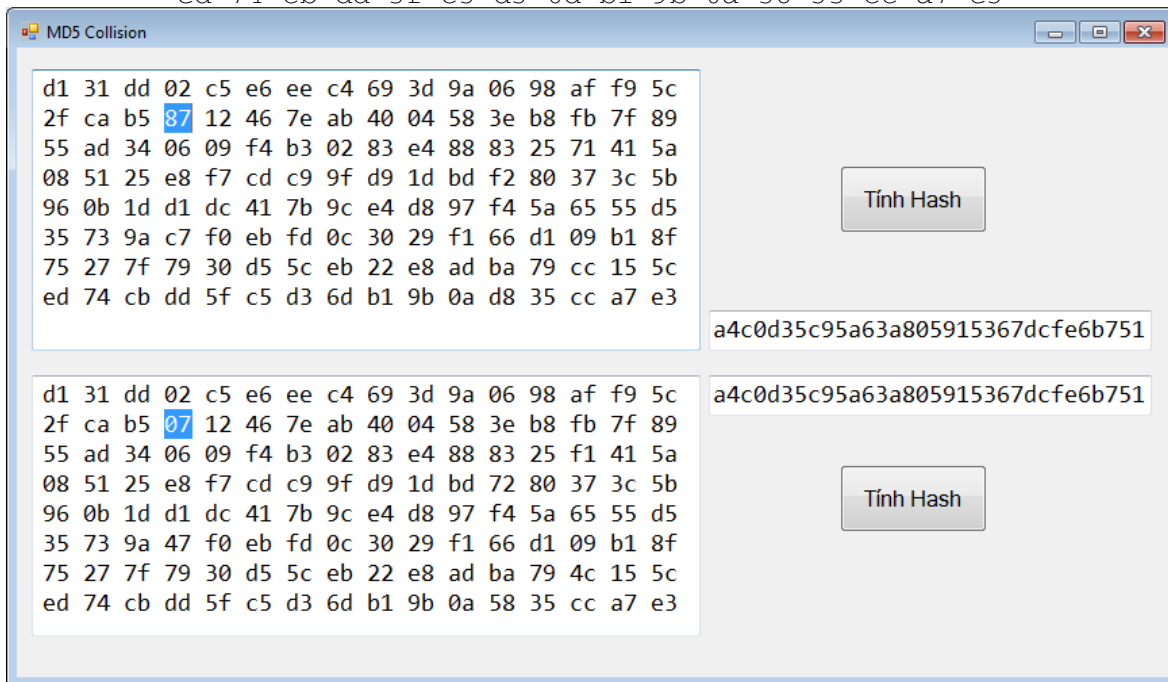
Thông điệp 2 (dạng số thập lục phân):

```
d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c
2f ca b5 07 12 46 7e ab 40 04 58 3e b8 fb 7f 89
55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 f1 41 5a
08 51 25 e8 f7 cd c9 9f d9 1d bd 72 80 37 3c 5b
96 0b 1d d1 dc 41 7b 9c e4 d8 97 f4 5a 65 55 d5
```

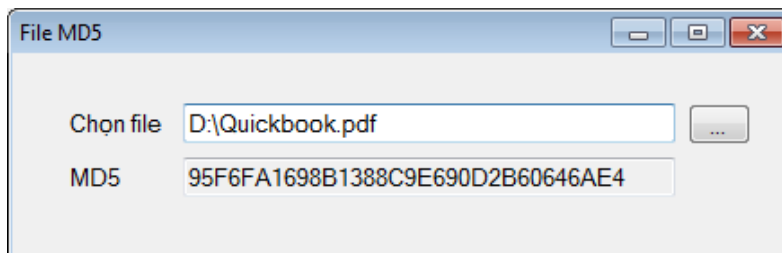
```

35 73 9a 47 f0 eb fd 0c 30 29 f1 66 d1 09 b1 8f
75 27 7f 79 30 d5 5c eb 22 e8 ad ba 79 4c 15 5c
ed 74 cb dd 5f c5 d3 6d b1 9b 0a 58 35 cc a7 e3

```



3. Viết chương trình tính giá trị MD5 cho một file trên máy tính tương tự như hình dưới đây:



4. Một giải pháp dùng để chống lại tình trạng vi phạm bản quyền, sao chép phần mềm mà không được sự đồng ý của tác giả, được thực hiện như sau:
- Sau khi cài đặt, phần mềm sẽ lấy thông tin về ID của CPU (hay ID của đĩa cứng) trên máy người mua phần mềm và gửi về cho nhà cung cấp phần mềm.
  - Dùng chữ ký điện tử, nhà cung cấp phần mềm ký vào ID của CPU (hay ID của đĩa cứng) của người mua, sau đó gửi lại nội dung đã ký cho người mua.
  - Mỗi khi chạy chương trình, phần mềm sẽ giải mã chữ ký của nhà cung cấp để lấy ID CPU được ký, đồng thời lấy lại thông tin về ID CPU của máy đang chạy. Nếu hai ID này không khớp, thì nghĩa là phần mềm đã bị sao chép vào một máy tính khác không có bản quyền.

Dùng chữ ký điện tử RSA (hoặc chữ ký điện tử DSS – xem chương 10), hãy viết chương trình thực hiện cơ chế chống vi phạm bản quyền nói trên cho một phần mềm nào đó của bạn.





## CHƯƠNG 6. GIAO THỨC

Trong các chương trước, chúng ta đã tìm hiểu về cách thức thực hiện tính bảo mật, tính chứng thực và tính không thoái thác của các phương pháp mã hóa đối xứng và mã hóa khóa công khai. Chương này trước tiên tìm hiểu cơ chế chống lại hình thức tấn công phát lại thông điệp (replay attack). Tiếp theo trình bày về các giao thức bảo mật, là các nguyên tắc áp dụng các kỹ thuật mã hóa nhằm đảm bảo việc truyền dữ liệu là an toàn trước những hình thức tấn công đã được đề cập trong chương một. Chương này trình bày các giao thức dưới dạng nguyên tắc lý thuyết, chương tiếp theo trình bày một số giao thức ứng dụng thực tiễn.

### 6.1 Phát lại thông điệp (Replay Attack)

Trong hình thức tấn công phát lại thông điệp, Trudy chặn thông điệp của Alice gửi cho Bob, và sau đó một thời gian gửi lại thông điệp này cho Bob. Như vậy Bob sẽ nghĩ rằng Alice gửi thông điệp hai lần khác nhau. Tuy nhiên thực sự thì Alice chỉ gửi một lần. Chỉ sử dụng mã hóa đối xứng và mã hóa khóa công khai thì không thể ngăn cản hình thức tấn công này. Để chống lại replay attack có 3 phương pháp:

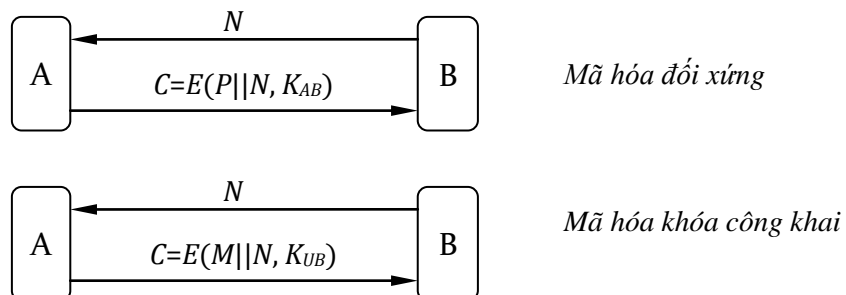
- 1) *Dùng số định danh*: trong mỗi thông điệp gửi cho Bob, Alice nhúng vào đó một con số định danh thông điệp  $S$ . Mỗi thông điệp ứng với một  $S$  khác nhau.

$$C = E(P||S, K_{AB}) \quad || \text{ là phép nối dây bít}$$

Do đó nếu Trudy phát lại thông điệp, Bob biết được hai thông điệp có cùng số định danh và loại bỏ thông điệp thứ hai. Tuy nhiên, phương pháp này có hạn chế là Bob phải lưu trữ số định danh của Alice để có cơ sở so sánh. Do đó phương pháp này thường chỉ sử dụng cho một phiên làm việc (connection oriented).

- 2) *Dùng timestamp*: trong mỗi thông điệp gửi cho Bob, Alice nhúng vào một timestamp  $T$  xác định thời điểm gửi. Bob chỉ chấp nhận thông điệp nếu nó đến được Bob trong một giới hạn thời gian nào đó kể từ lúc gửi. Tuy nhiên phương pháp này yêu cầu đồng hồ của Alice và của Bob phải đồng bộ, không được sai lệch đáng kể. Ngoài ra độ trễ của việc truyền tin trên mạng cũng là một trở ngại đối với phương pháp này.

- 3) *Dùng cơ chế challenge/response*: để bảo đảm thông điệp từ Alice không phải là replay, Bob gửi 1 số ngẫu nhiên  $N$  cho Alice (gọi là nonce). Alice sẽ nhúng  $N$  trong thông điệp gửi cho Bob.



Khi Bob giải mã thì sẽ kiểm tra  $N$  mà Bob nhận được xem có trùng khớp với  $N$  Bob gửi đi không. Như vậy Trudy không thể replay thông điệp  $E(P||N, K_{AB})$  được vì mỗi lần Bob sẽ gửi một số  $N$  khác nhau. Tuy nhiên phương pháp này đòi hỏi thêm một bước là Bob phải gửi  $N$  trước cho Alice. Vì vậy trong thực tế tùy trường hợp mà người ta sẽ sử dụng một trong 3 kỹ thuật trên cho hợp lý.

## 6.2 Giao thức bảo mật

Trong thực tế, khi hai người bất kỳ chưa biết trước muốn trao đổi dữ liệu với nhau, họ phải xác định người kia là ai, sau đó thống nhất với nhau là phải dùng phương pháp mã hóa nào, khóa là gì,... Để làm được điều đó họ phải tiến hành thông qua giao thức bảo mật.

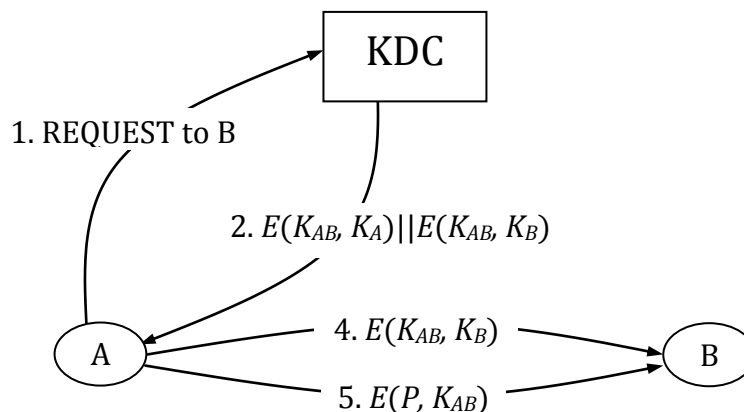
Như vậy có thể định nghĩa giao thức bảo mật là các quy định mà nếu hai cá thể tuân theo các quy định đó, thì họ có thể trao đổi dữ liệu với nhau một cách an toàn bảo mật. Một giao thức bảo mật thường nhằm xác định các yếu tố sau:

- Định danh hai cá thể trao đổi dữ liệu, chống replay attack.
- Trao đổi khóa phiên bí mật để mã hóa dữ liệu. Vì mã đối xứng thực hiện nhanh hơn mã hóa công khai nên ngày nay người ta dùng mã đối xứng để mã hóa dữ liệu, còn việc trao đổi khóa phiên bí mật thì có thể dùng mã hóa đối xứng hay mã hóa khóa công khai.

Trong phần 3.9 hay phần 4.6.2 và 4.7 chúng ta đã xem một số giao thức tập trung vào việc trao đổi khóa phiên. Trong phần này, ta sẽ mở rộng các giao thức trên nhằm định danh cá thể trao đổi dữ liệu và chống replay attack.

### 6.2.1 Định danh và trao đổi khóa phiên dùng mã hóa đối xứng với KDC

Xét lại mô hình phần 3.9 trao đổi khóa phiên



Mô hình trên có thể bị tấn công replay attack. Ví dụ, Trudy có thể replay bước 4 mà B vẫn nghĩ là A gửi và B tiếp tục dùng  $K_{AB}$  này làm khóa phiên. Dựa trên cơ sở đó Trudy tiếp tục replay bước 5. (việc replay dữ liệu tại bước 5 sẽ gây ra hậu quả không mong muốn như chúng ta đã đề cập trong chương 1).

Needham and Schroeder đã đề xuất sửa đổi mô hình trên như sau:

- 1)  $A \rightarrow KDC: ID_A || ID_B || N_1$
- 2)  $KDC \rightarrow A: E(K_S || ID_B || N_1 || E(K_S || ID_A, K_B), K_A)$  //  $K_S$  là khóa phiên,  $ID_B$  để A biết khóa phiên này dùng với B

- 3) A giải mã có được  $K_S$  và  $E(K_S || ID_A, K_B)$
- 4)  $A \rightarrow B: E(K_S || ID_A, K_B)$  //  $ID_A$  để B biết khóa phiên này dùng với A
- 5)  $B \rightarrow A: E(N_2, K_S)$
- 6)  $A \rightarrow B: E(f(N_2), K_S)$  // f là hàm bất kỳ
- 7)  $A \rightarrow B: E(P, K_S)$

Tại bước 1, A gửi cho KDC nonce  $N_1$  và KDC nhúng  $N_1$  vào trong bản rõ ở bước 2. Do đó bước 2 không thể bị replay attack (theo phương pháp challenge/response).

Tại bước 5, B gửi cho A giá trị nonce  $N_2$ , và chờ A gửi lại giá trị  $f(N_2)$ , f là một hàm được chọn trước. Do đó nếu Trudy replay attack tại bước 4 thì Trudy không thể thực hiện bước 6 vì Trudy không có  $K_S$  để tính  $N_2$  và  $f(N_2)$ . Bob nhận biết Trudy là giả mạo và Trudy không thể replay dữ liệu tiếp tại bước 7.

Như vậy có thể thấy các bước 4, 5, 6 cũng là một hình thức challenge/response để chống replay attack. Để phòng Trudy replay bước 4 để sử dụng lại một  $K_S$  cũ. Bob challenge tại bước 5 và yêu cầu được response tại bước 6 xem người gửi có biết  $K_S$  không (chỉ có Alice mới biết  $K_S$ )

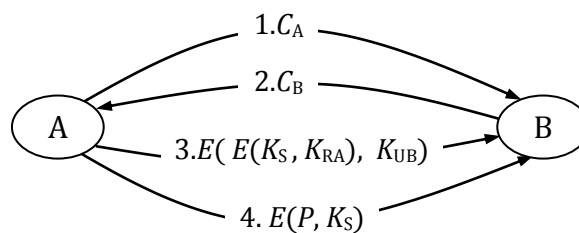
Tuy nhiên giao thức này chưa hoàn toàn chắc chắn, có một khuyết điểm là nếu sau này Trudy biết được  $K_S$  và  $E(K_S || ID_A, K_B)$  tương ứng thì Trudy có thể replay attack bước 4, sau đó dựa trên  $K_S$  tính được  $N_2$  và phản hồi  $N_2$  cho Bob. Như vậy Bob không biết được là Trudy đã mạo danh Alice và tiếp tục dùng khóa phiên  $K_S$  đã bị lộ này. Do đó giao thức Needham/Schroeder tiếp tục được sửa lại như sau:

- 1)  $A \rightarrow B: ID_A || N_A$
- 2)  $B \rightarrow KDC: ID_B || N_B || E(ID_A || N_A, K_B)$
- 3)  $KDC \rightarrow A: E(ID_B || N_A || K_S, K_A) || E(ID_A || K_S, K_B) || N_B$
- 4)  $A \rightarrow B: E(ID_A || K_S, K_B) || E(N_B, K_S)$
- 5)  $A \rightarrow B: E(P, K_S)$

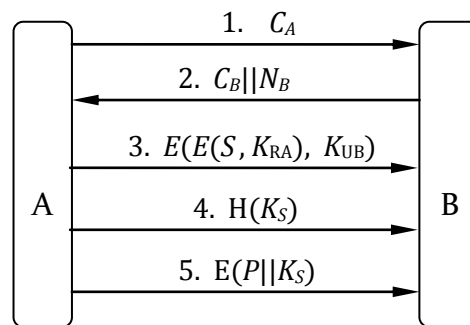
Trong giao thức trên A gửi  $N_A$  cho Bob, Bob gửi tiếp cho KDC, KDC nhúng  $N_A$  vào bản rõ gửi cho A. Do đó nếu A nhận được  $N_A$  thì có nghĩa là bản mã  $E(ID_B || N_A || K_S, K_A)$  trong bước 3 không bị replay attack. B gửi  $N_B$  cho KDC, KDC gửi lại cho A, A gửi lại  $N_B$  cho B dưới dạng mã hóa. Do đó nếu B nhận được  $N_B$  thì có nghĩa  $E(ID_A || K_S, K_B)$  trong bước 4 không bị replay attack. Do đó  $K_S$  mà Alice và Bob nhận được là khóa phiên mới. Trudy không thể replay lại các bản mã  $E(P, K_S)$  cũ trong các lần trước tại bước 5.

### 6.2.2 Định danh và trao đổi khóa phiên dùng mã hóa khóa công khai

Xét lại mô hình phần 4.6.2



Trong mô hình trên, Trudy có thể replay bước 3 mà B vẫn nghĩ là A gửi và B tiếp tục dùng  $K_S$  này làm khóa phiên. Dựa trên cơ sở đó Trudy tiếp tục replay bước 4. Ở đây áp dụng một cơ chế challenge/response khác để chống replay như sau:



Mô tả:

- Bước 1: A gửi chứng chỉ  $C_A$  cho B.
- Bước 2: B gửi chứng chỉ  $C_B$  và nonce  $N_B$  cho A.
- Bước 3: A chọn một *tiền khóa phiên*  $S$  và tính được khóa phiên  $K_S = H(S || N_B)$ . A gửi chứng thực và bảo mật  $S$  cho B. B cũng tính khóa phiên  $K_S$ .
- Bước 4: A gửi giá trị hash  $H(K_S)$  cho B, B kiểm tra giá trị hash này với giá trị hash do B tự tính. Nếu khớp, B biết được rằng bước 3 không thể bị replay attack.

Giả sử Trudy replay bước 3 nhưng không biết  $S$ , vậy Trudy không tính được  $K_S$  tương ứng với  $N_B$  mới của Bob, từ đó Trudy cũng không thể tính được  $H(K_S)$ . Do đó Trudy không thể replay bước 4 mà không bị Bob phát hiện.

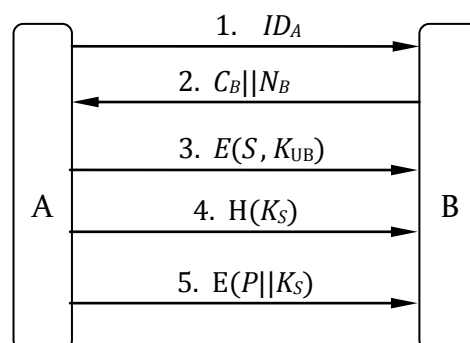
- Bước 5: A và B tiến hành trao đổi dữ liệu.

### 6.3 Câu hỏi ôn tập

- 1) Tấn công phát lại thông điệp là gì? Nêu tác hại của thao tác tấn công này và so sánh với việc sửa đổi thông điệp vào mạo danh.
- 2) Nêu các phương pháp chống lại tấn công phát lại thông điệp.
- 3) Nêu các mục đích của giao thức.

### 6.4 Bài tập

Xét giao thức sau:

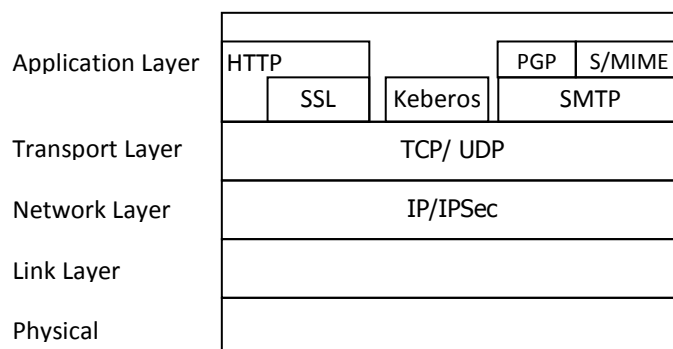


- a) B có thể chắc chắn A là người ứng với  $ID_A$  không? Nếu Trudy mạo danh A sử dụng  $ID_A$  thì B có phát hiện được không? Giải thích
- b) Giả sử A có password để định danh với B, B lưu trữ giá trị hash password của A. Hãy sửa giao thức trên để B có thể định danh được A.

## CHƯƠNG 7. MỘT SỐ ỨNG DỤNG THỰC TIỄN

### 7.1 Giới thiệu

Trong chương này, chúng ta sẽ tìm hiểu việc áp dụng các mô hình lý thuyết trong các chương trước vào một số giao thức thực tiễn. Trước hết là chuẩn chứng thực X.509, là một chuẩn thực tiễn áp dụng trong vấn đề trao đổi khóa công khai mà đã được đề cập trong phần 4.6.1. Tiếp theo sau đó chúng ta sẽ tìm hiểu về giao thức bảo mật web Secure Socker Layer (SSL), giao thức bảo mật mạng cục bộ Keberos. Có thể minh họa các giao thức trên trong mô hình mạng OSI như sau:



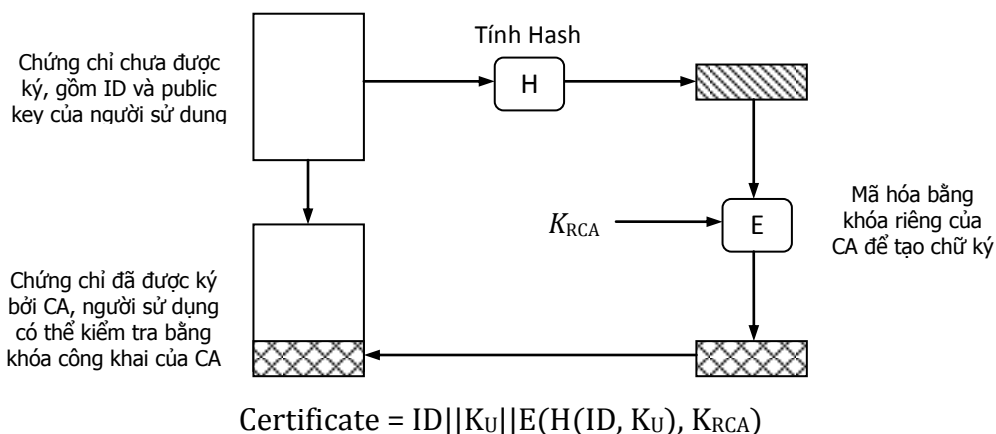
Trong mô hình trên có thể thấy việc ứng dụng bảo mật vào truyền thông trên mạng có thể được tiến hành tại các tầng khác nhau như tầng mạng hay tầng ứng dụng. Trong giao thức TCP/IP, người ta có thể thay giao thức IP thường bằng giao thức IP Security để việc bảo mật được thực hiện tại tầng mạng. Do đó các ứng dụng khác trong tầng ứng dụng sẽ không cần quan tâm đến bảo mật nữa, mọi việc bảo mật đã được IPSec thực hiện. Chi tiết về IPSec được trình bày trong [3].

Các giao thức SSL, Keberos, PGP hay S/MIME được thực hiện trong tầng ứng dụng. Vì vậy mỗi giao thức phải thực hiện cơ chế bảo mật cho riêng mình.

### 7.2 Chứng thực X.509

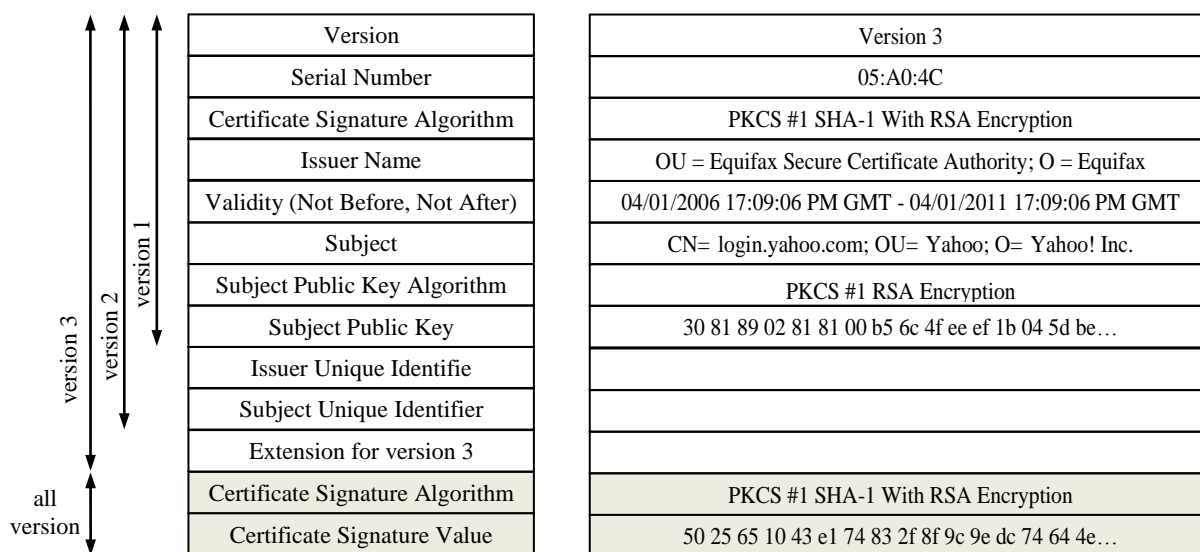
#### 7.2.1 Cấu trúc chứng thực

Chứng thực X.509 là một áp dụng dựa trên lý thuyết về chữ ký điện tử trong phần 5.4. Sơ đồ nguyên tắc để sinh ra chứng thực X.509 như sau:



Hình 7-1. Sơ đồ tạo chứng chỉ X.509

Cấu trúc một chứng chỉ X.509 gồm có các thành phần sau:



**Hình 7-2. Cấu trúc và ví dụ một chứng chỉ X.509**

Mục đích của các thành phần trên là:

- Version: phiên bản X.509 của chứng chỉ này, có 3 phiên bản là 1, 2 và 3.
- Serial Number: số serial của chứng chỉ này do trung tâm chứng thực CA ban hành.
- Certificate Signature Algorithm: thuật toán ký chứng chỉ, gồm loại hàm Hash và phương pháp mã hóa khóa công khai.
- Issuer name: Tên của trung tâm chứng thực CA (CN: common name, O: organization, OU: organization unit).
- Validity: thời gian hiệu lực của chứng chỉ.
- Subject: tên chủ sở hữu chứng chỉ, cũng gồm có CN, O, OU,...
- Subject Public Key Algorithm: thuật toán mã hóa khóa công khai mà tương ứng với khóa công khai trong chứng chỉ.
- Subject Public Key: khóa công khai trong chứng chỉ, tức khóa công khai của chủ sở hữu. Đối với RSA thì thuộc tính này lưu giữ giá trị Modulus và Exponent nối tiếp nhau (N và e).
- Issuer Unique Identifier, Subject Unique Identifier: dành cho version 2, ít được sử dụng.
- Extension: dành cho version 3.
- Certificate Signature Algorithm: thuật toán ký chứng chỉ, giống mục thứ 3.
- Certificate Signature Value: giá trị của chữ ký.

Đối với version 3 phần Extension có thể gồm các thông tin sau:

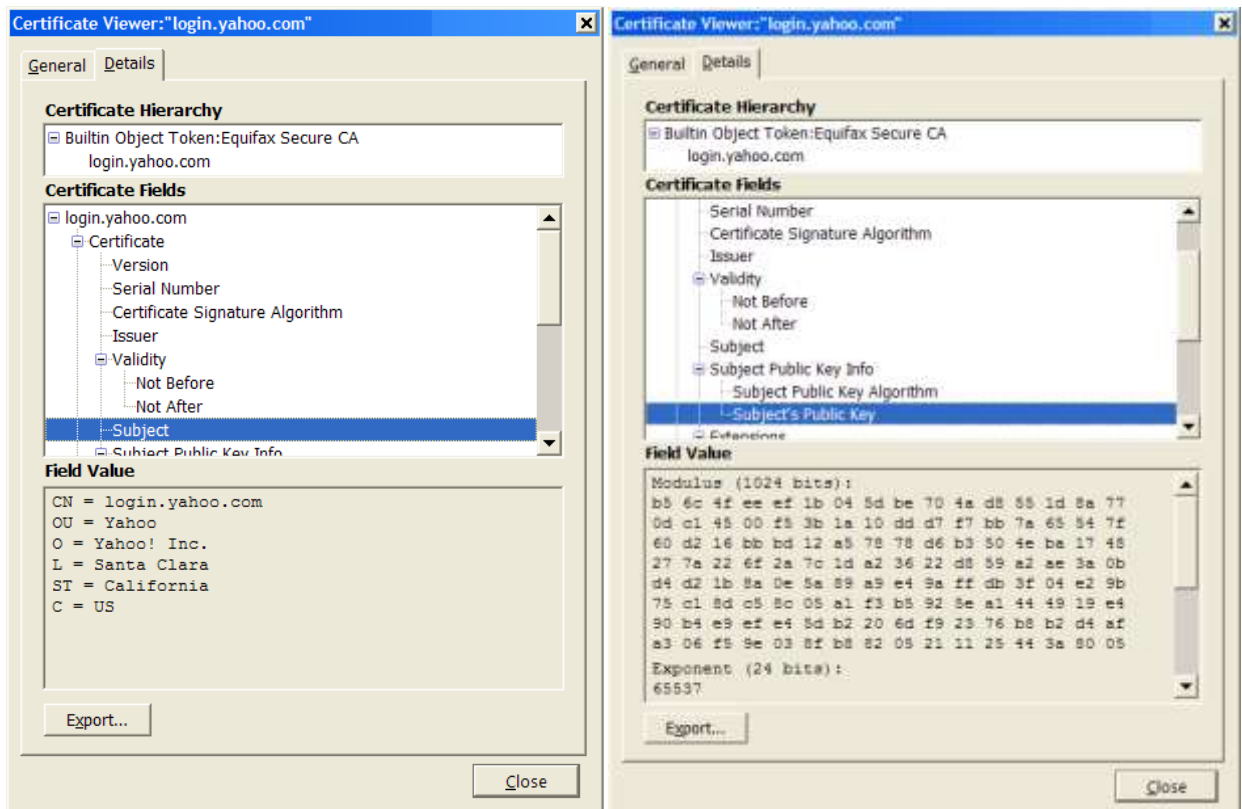
- Authority key identifier: Một con số dùng để định danh trung tâm chứng thực. Thuộc tính Issuer Name cung cấp tên trung tâm chứng thực dưới dạng text, điều này có thể gây nhầm lẫn.
- Subject key identifier: Một con số dùng để định danh người sử dụng được chứng thực. Tương tự như Issuer Name, thuộc tính Subject cũng cung cấp tên



người dưới dạng text, điều này có thể gây nhầm lẫn. Ngoài ra việc dùng một con số định danh cho phép một người sử dụng có thể có nhiều chứng chỉ khác nhau.

- Key Usage: mục đích sử dụng của chứng chỉ. Mỗi chứng chỉ có thể có một hoặc nhiều mục đích sử dụng như: mã hóa dữ liệu, mã hóa khóa, chữ ký điện tử, không thoái thác ...
- CRL Distribution Point: địa chỉ để lấy danh sách các chứng chỉ đã hết hạn hay bị thu hồi (certificate revocation list).

Một chứng chỉ thường được lưu trên một file có phần mở rộng là .cer.



**Hình 7-3. Xem nội dung một chứng thực trong Firefox 2.0 (dùng trong giao thức SSL)**

Vì chứng chỉ được ký bằng khóa riêng của CA, nên bảo đảm rằng chữ ký không thể bị làm giả và bất cứ ai tin tưởng vào khóa công khai của CA thì có thể tin tưởng vào chứng chỉ mà CA đó cấp phát. Do đó khóa công khai của CA phải được cung cấp một cách tuyệt đối an toàn đến tay người sử dụng. Trong ví dụ trên chứng thực của Yahoo được cung cấp bởi Equifax Secure. FireFox tin tưởng vào Equifax và khóa công khai của Equifax được tích hợp sẵn trong bộ cài đặt của FireFox. Vì vậy khi duyệt đến trang web của Yahoo, FireFox có được chứng chỉ của Yahoo, vì FireFox tin tưởng vào Equifax nên cũng sẽ tin tưởng vào Yahoo và cho phép người sử dụng duyệt trang web này (xem thêm phần giao thức SSL bên dưới).

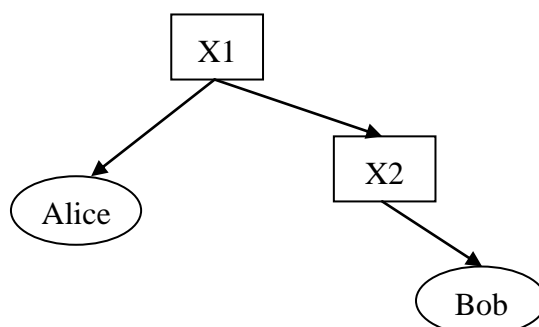
Trên thế giới hiện nay có nhiều tổ chức cung cấp chứng thực X509 như VeriSign, Equifax, Thawte, SecureNet... VeriSign hiện là tổ chức lớn nhất. Verisign cung cấp chứng chỉ X509 theo ba mức độ (class):

- Class 1: ID của một đối tượng là email của đối tượng đó. Sau khi đối tượng đăng ký email và public key qua mạng Internet, Verisign gửi email để kiểm tra địa chỉ email hợp lệ và cấp chứng thực.
- Class 2: ID là địa chỉ nơi ở của đối tượng, Verisign sẽ gửi confirm qua đường bưu điện để kiểm tra địa chỉ hợp lệ.
- Class 3: đối tượng cần có giấy tờ pháp lý để chứng minh tư cách pháp nhân.

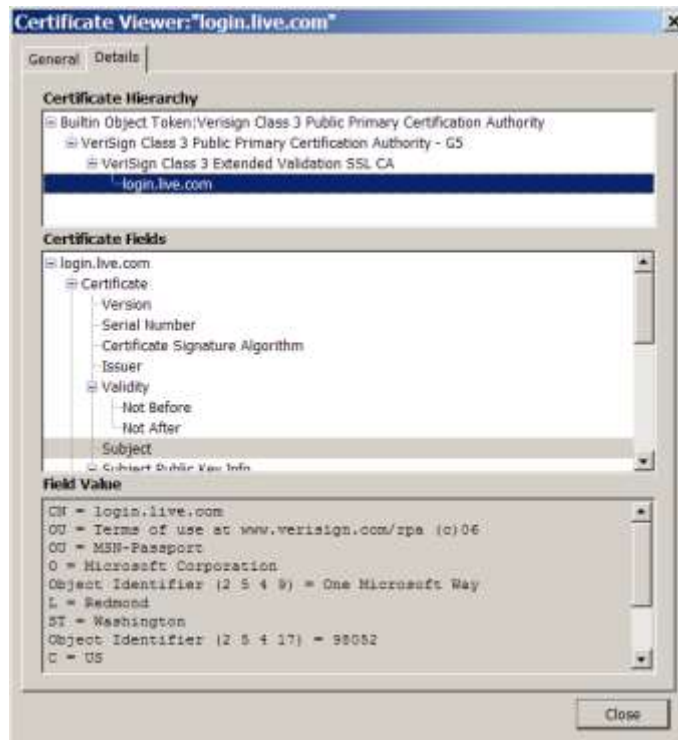
### 7.2.2 Phân cấp chứng thực

Trên thế giới không thể chỉ có một trung tâm chứng thực CA duy nhất mà có thể có nhiều trung tâm chứng thực. Những người sử dụng khác nhau có thể đăng ký chứng thực tại các CA khác nhau. Do đó để có thể trao đổi dữ liệu, một người cần phải tin tưởng vào khóa công khai của *tất cả* các trung tâm chứng thực. Để giảm bớt gánh nặng này, X.509 đề ra cơ chế phân cấp chứng thực.

Ví dụ, Alice chỉ tin tưởng vào trung tâm chứng thực  $X_1$ , còn chứng thực của Bob là do trung tâm chứng thực  $X_2$  cung cấp. Nếu Alice không có khóa công khai của  $X_2$ , thì làm sao Alice có thể kiểm tra được chứng thực của Bob? Biện pháp giải quyết là Alice có thể đọc Authority key identifier (tức ID của  $X_2$ ) trong chứng thực của Bob. Sau đó Alice kiểm tra xem  $X_1$  có cấp chứng thực nào cho  $X_2$  hay không. Nếu có, Alice có thể tìm thấy được khóa công khai của  $X_2$  và tin tưởng vào khóa này (do đã được  $X_1$  xác nhận). Từ đó Alice có thể kiểm tra tính xác thực của chứng chỉ của Bob.



Việc phân cấp chứng thực này không chỉ giới hạn trong hai trung tâm chứng thực mà có thể thông qua một dãy các trung tâm chứng thực tạo thành một mạng lưới chứng thực (Web of Trust). Hình dưới minh họa một ví dụ thực tế.



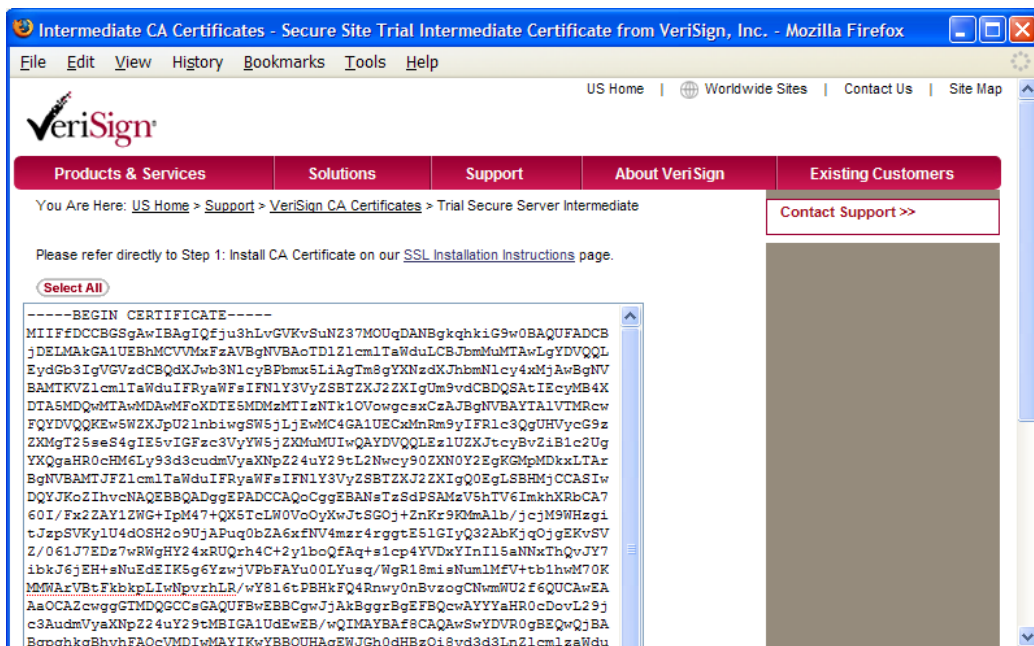
**Hình 7-4. Minh họa mô hình phân cấp chứng thực**

Trong ví dụ trên chứng thực MSN-Passport của Microsoft được chứng thực bởi “Verisign Class 3 Extended Validation SSL CA”, Firefox không có sẵn khóa công khai của trung tâm này. Tuy nhiên Firefox có khóa công khai của “Verisign Class 3 Public Primary CA”, từ đó FireFox có thể chứng thực trung tâm “Verisign Class 3 Public Primary CA – G5” và qua đó có thể chứng thực được “Verisign Class 3 Extended Validation SSL CA”.

### 7.2.3 Các định dạng file của chứng chỉ X.509

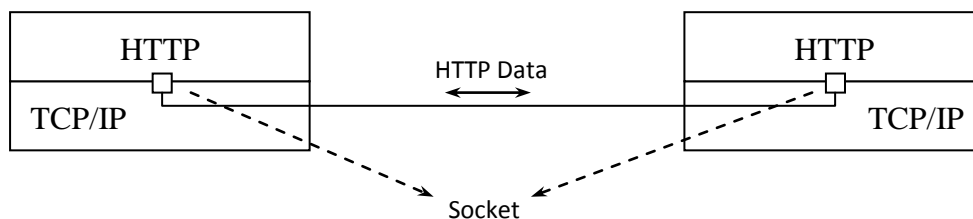
- 1) Định dạng DER (.cer): nội dung của chứng chỉ X.509 được lưu dưới format DER, một định dạng dữ liệu binary chuẩn cho các môi trường máy tính.
- 2) Định dạng PEM (.pem): là định dạng DER và được mã hóa dưới dạng text theo chuẩn Base64. Một file text PEM bắt đầu bằng dòng -----BEGIN CERTIFICATE----- và kết thúc bằng dòng -----END CERTIFICATE-----
- 3) Định dạng PKCS#7 (.p7c hay .p7b): là một định dạng dữ liệu được mã hóa hay ký. Do đó có đi kèm cả chứng chỉ.
- 4) Định dạng PKCS#10 (.p10 hay .p10): là một định dạng dùng để gửi yêu cầu cấp chứng chỉ X509 đến trung tâm chứng thực. Định dạng này có ID và public key của người yêu cầu.
- 5) Định dạng PKCS#12 (.p12): lưu trữ chứng chỉ X509 và private key tương ứng (có password bảo vệ) trong cùng 1 file.
- 6) Định dạng PFX (.pfx): cũng lưu chứng chỉ X509 và private key theo định dạng của Microsoft.

Hình bên dưới là một chứng chỉ của Verisign được cung cấp dưới dạng PEM



### 7.3 Giao thức bảo mật web Secure Socket Layer version 3 - SSLv3

Dữ liệu Web được trao đổi giữa trình duyệt và web server được thực hiện qua giao thức HTTP. Client kết nối với server qua socket của giao thức TCP/IP.



Hình sau minh họa dữ liệu của giao thức HTTP khi thực hiện tìm kiếm từ “Nhà Trang” trong website vn.search.yahoo.com.

```
GET /search?p=Nha+Trang&fcss=on&fr=yfp-t-101&toggle=1&cop=&ei=UTF-8 HTTP/1.1
Host: vn.search.yahoo.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.13) Gecko/2009073022
Firefox/3.0.13 (.NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://vn.yahoo.com/?p=us
```

và hình dưới là dữ liệu phản hồi của server yahoo. Dữ liệu này gồm hai phần, phần đầu theo quy định của giao thức HTTP, phần sau là dữ liệu HTML.

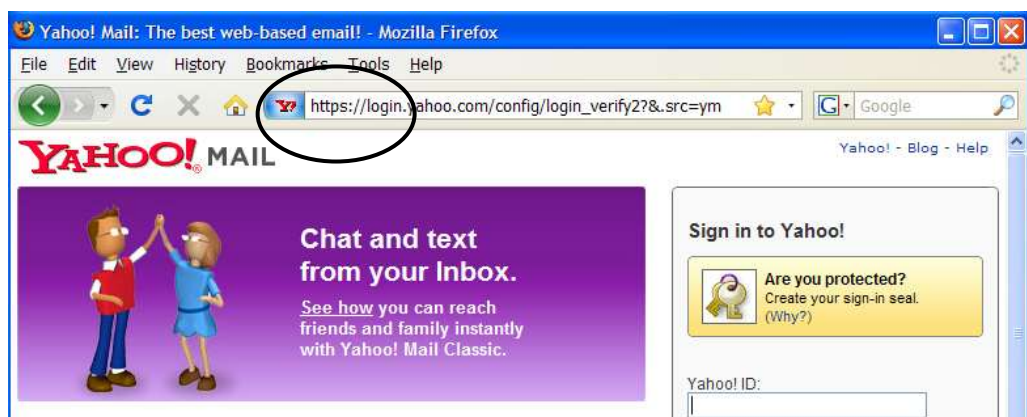
```

HTTP/1.1 200 OK
Date: Fri, 14 Aug 2009 10:25:49 GMT
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: Keep-Alive
Content-Encoding: gzip

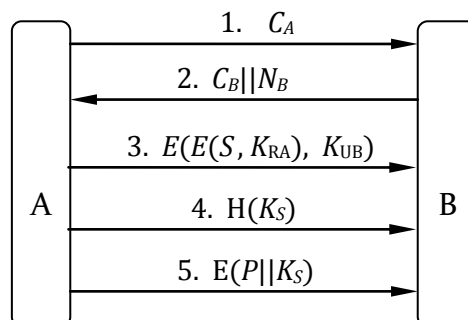
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="vi"><head> ... </head>
....
</html>

```

Giao thức SSL bảo mật dữ liệu trao đổi qua socket. Vì vậy nên có tên gọi là Secure Socket Layer (URL bắt đầu bằng <https://>). Đây là giao thức bảo mật kết hợp mã hóa khóa công khai và khóa đối xứng như đã trình bày trong phần 4.6.2 trong đó mã hóa RSA được dùng để trao đổi khóa phiên của mã hóa đối xứng.

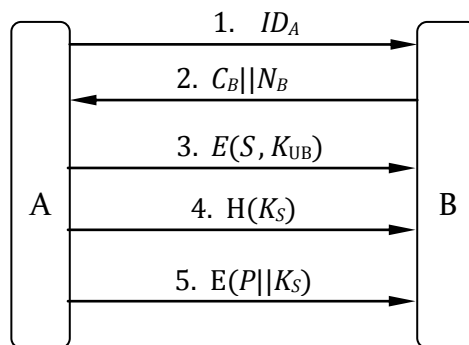


Xét lại mô hình trao đổi khóa phiên trong phần 6.2.2.



Mô hình này yêu cầu mỗi người duyệt web (A) và mỗi website (B) đều phải có cặp khóa riêng và khóa công khai. Hay nói cách khác website và người duyệt phải có chứng thực. Điều này sẽ gây khó khăn cho người duyệt web vì phải có chứng chỉ. Đây là yêu cầu cần thiết để đảm bảo tuyệt đối tính chứng thực cho cả hai phía website và người duyệt. Nghĩa là khóa  $K_S$  phải xuất phát từ một người duyệt A cụ thể nào đó mà website biết, đồng thời khóa  $K_S$  đến đúng website B chứ không phải là website khác.

Tuy nhiên trong thực tế không phải lúc nào cũng cần chứng thực từ phía người sử dụng. Ví dụ, khi bạn mua hàng tại cửa hàng sách Amazon. Amazon không cần biết bạn là ai, chỉ cần bạn có tài khoản để mua hàng (việc bảo mật tài khoản người mua là trách nhiệm của mã hóa đối xứng). Do đó Amazon không cần chứng thực người duyệt web. Vì vậy trong trường hợp này, người duyệt không cần có chứng chỉ. Lúc này mô hình trao đổi khóa là:



**Hình 7-5. Sơ đồ trao đổi khóa phiên chỉ cần chứng thực 1 phía**

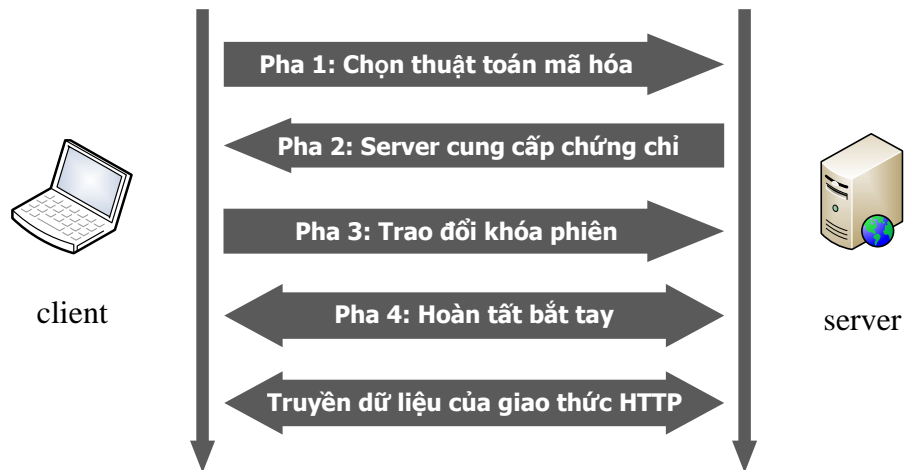
Mô hình trên đảm bảo ngoài người duyệt A chỉ có website B là biết được khóa phiên  $K_S$ , còn A là ai thì website không cần biết. Để chứng thực người sử dụng, website có thể đơn giản lưu password của người sử dụng và chứng thực qua cơ chế login. Cách thức này hiện nay đang được sử dụng phổ biến hơn là phải yêu cầu người sử dụng cung cấp chứng chỉ chứng thực.

Giao thức SSL cho phép thực hiện cả hai khả năng trao đổi khóa nói trên.

Một phương pháp khác mà SSL cũng sử dụng để trao đổi khóa là phương pháp Diffie-Hellman. SSL có ba dạng Diffie-Hellman.

- Fixed Diffie-Hellman: là phương pháp trao đổi khóa Diffie-Hellman mà trong đó các yếu tố công khai ( $g, t$ ) được chứng thực giống như chứng thực khóa công khai của RSA. Điều này giúp ngăn chặn hình thức tấn công kẻ-đứng-giữa.
- Ephemeral Diffie-Hellman: là phương pháp trao đổi khóa Diffie-Hellman được bảo vệ bằng mã hóa khóa công khai RSA. Đây là hình thức Diffie-Hellman an toàn nhất.
- Anonymous Diffie-Hellman: Diffie-Hellman thường, do đó có thể bị tấn công theo hình thức kẻ-đứng-giữa.

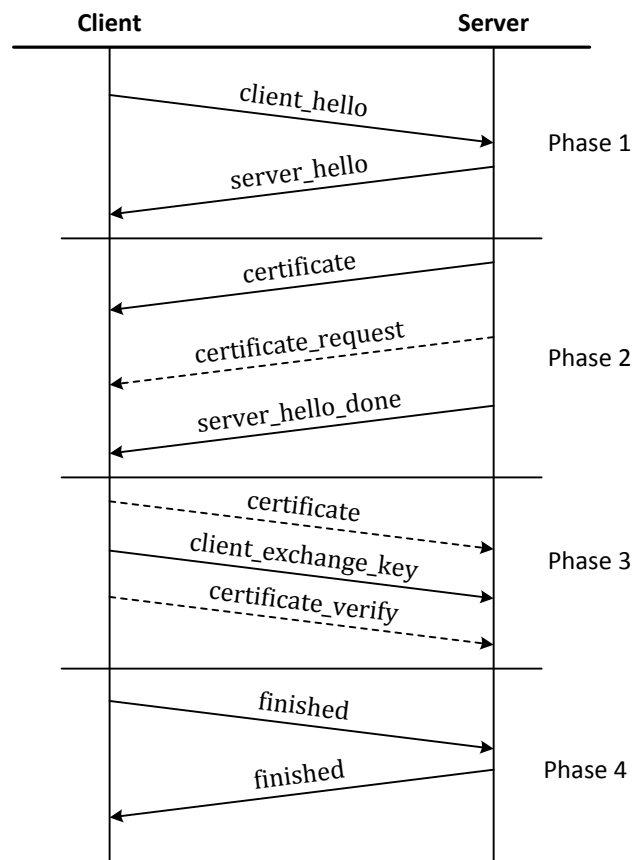
Các phương pháp mã hóa đối xứng mà SSL có thể thực hiện là RC4, RC2, DES, 3DES, IDEA, AES. Hình sau đây minh họa mô hình đơn giản của giao thức SSL.



Do có thể áp dụng nhiều phương pháp mã hóa khác nhau nên đặc tả của giao thức SSL khá phức tạp. Phần tiếp theo sẽ chủ yếu trình bày giao thức SSL version 3 trong trường hợp sử dụng RSA. SSL gồm có hai phần cơ bản là giao thức bắt tay và giao thức truyền dữ liệu.

### 7.3.1 Giao thức bắt tay - SSL Handshaking Protocol

Trước khi tiến hành truyền số liệu, SSL thực hiện giao thức bắt tay để chứng thực website và chứng thực người duyệt web, trao đổi khóa phiên và thống nhất các thuật toán mã hóa được sử dụng. Sơ đồ bắt tay được minh họa trong hình bên dưới.



(đường nét đứt là các thông điệp không bắt buộc, chỉ sử dụng khi cần chứng thực từ phía client)

### Hình 7-6. Giao thức bắt tay SSL

Sơ đồ trên gồm có 10 loại thông điệp và được chia thành 4 pha:

1) **Pha 1:** thỏa thuận về phương pháp mã hóa được sử dụng. Pha này bắt đầu bằng thông điệp *client\_hello* được gửi từ client đến website, thông điệp này gồm các tham số sau:

- Version: phiên bản SSL cao nhất mà client sử dụng
- Random: là một cấu trúc ngẫu nhiên gồm 32 byte
- SessionID: nếu bằng 0 có nghĩa là client muốn thiết lập một session mới hoàn toàn. Nếu khác 0 nghĩa là client muốn thiết lập một kết nối mới trong session này. Việc dùng session giúp cho client và server giảm các bước thỏa thuận trong quá trình bắt tay.
- CompressionMethod: phương pháp nén dữ liệu sử dụng trong quá trình truyền dữ liệu
- CipherSuite: Các phương pháp mã hóa khóa công khai dùng để trao đổi khóa phiên như RSA, Fixed Diffie-Hellman, Ephemeral Diffie-Hellman, Anonymous Diffie-Hellman. Phương pháp nào liệt kê trước thì có được ưu tiên hơn. Ứng với mỗi phương pháp trao đổi khóa là danh sách các loại mã hóa đối xứng được sử dụng. Gồm các tham số sau:
  - CipherAlgorithm: phương pháp mã hóa đối xứng sử dụng (là một trong các phương pháp mã khối RC2, DES, 3DES, IDEA, AES, Fortezza hay mã dòng RC4)
  - Hash Algorithm: MD5 hay SHA-1.
  - CipherType: mã hóa đối xứng là mã khối hay mã dòng.
  - KeyMaterial: một chuỗi byte được dùng để sinh khóa.
  - IV Size: kích thước của IV dùng trong mô hình CBC của mã khối.
- Sau khi nhận được *client\_hello* server sẽ trả lời bằng thông điệp *server\_hello* để xác các thuật toán được sử dụng.

2) **Pha 2:** chứng thực server và trao đổi khóa của mã hóa công khai. Sau khi đã xác nhận thuật toán mã hóa với client, server tiếp tục thực hiện các thông điệp sau:

- Thông điệp *certificate*: server cung cấp *certificate* của mình cho client (dưới dạng chứng chỉ X.509) .
- Thông điệp *certificate\_request*: trong trường hợp server cần chứng thực người sử dụng, server sẽ gửi thông điệp này để yêu cầu client cung cấp chứng chỉ.
- Thông điệp *server\_hello\_done*: báo hiệu server đã hoàn tất pha 2.

3) **Pha 3:** chứng thực client và trao đổi khóa của mã hóa đối xứng

- Thông điệp *certificate*: nếu server yêu cầu *certificate*, client cung cấp *certificate* của mình cho server.
- Thông điệp *client\_key\_exchange*: trong bước này client gửi các thông số cần thiết cho server để tạo khóa bí mật. Ta cũng sẽ chỉ đề cập đến trường hợp RSA. Trong trường hợp này client tạo một giá trị bất kỳ gọi là “tiền khóa chủ” (pre-master secret) có kích thước 48 byte, mã hóa bằng khóa



công khai của server. Sau khi có “pre-master secret”, client và server sẽ tính giá trị “khóa chủ” (master-secret) như sau:

```
master_secret = MD5(pre_master_secret || SHA('A' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random)) ||
    MD5(pre_master_secret || SHA('BB' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random)) ||
    MD5(pre_master_secret || SHA('CCC' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random))
```

Master\_secret cũng có chiều dài là 48 byte (384 bit). Phép toán || là phép nối

- Thông điệp certificate\_verify: là chữ ký của client trong trường hợp server cần chứng thực client. Client phải dùng khóa riêng để ký chữ ký, do đó server có thể đảm bảo được là không ai khác dùng certificate của client để giả mạo.

**4) Pha 4:** hoàn tất quá trình bắt tay. Trong pha này client và server gửi thông điệp *finished* để thông báo hoàn tất quá trình bắt tay lẫn nhau. Tham số của thông điệp này là một giá trị hash để hai bên có thể kiểm tra lẫn nhau. Giá trị hash này kết nối của 2 giá trị hash:

```
MD5(master_secret || pad2 ||
    MD5(handshake_messages || Sender || master_secret || pad1))
SHA(master_secret || pad2 ||
    SHA(handshake_messages || Sender || master_secret || pad1))
```

Trong đó handshake\_messages là tất cả các thông điệp đầu đến trước thông điệp *finished* này. Sender là mã để phân biệt thông điệp *finished* này là từ client hay từ server. Đây là cơ chế chống replay attack dùng hàm hash mà chúng ta đã tìm hiểu trong chương 6.

Dựa trên giá trị master\_secret, client và server sẽ tính các tham số cần thiết cho mã hóa đối xứng như sau:

- Hai khóa dành cho việc mã hóa dữ liệu, một khóa dành cho chiều server gửi client và 1 khóa dành cho chiều client và server.
- Hai giá trị IV, cũng dành cho server và client tương ứng
- Hai khóa dành cho việc tính giá trị MAC, cũng tương ứng cho server và client

Tùy theo phương pháp mã hóa đối xứng được sử dụng mà các tham số này có chiều dài khác nhau. Tuy nhiên, chúng được lấy từ dãy bit theo công thức sau:

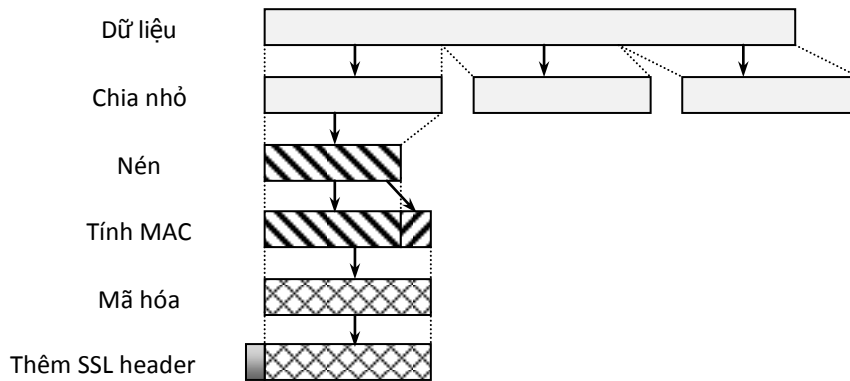
```
key_block = MD5(master_secret || SHA('A' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('BB' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('CCC' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    ...
```

Việc dùng các giá trị ClientHello.random và ServerHello.random sẽ làm phức tạp việc phá mã hơn.

Đến đây client và server đã hoàn tất quá trình bắt tay trao đổi khóa, sẵn sàng để truyền số liệu theo giao thức truyền số liệu.

### 7.3.2 Giao thức truyền số liệu - SSL Record Protocol

Hình bên dưới minh họa các bước thực hiện trong quá trình truyền số liệu:



**Hình 7-7. Truyền dữ liệu theo khối trong SSL**

Trong giao thức truyền số liệu, dữ liệu được chia thành các khối có kích thước là  $2^{14}$  byte (16384) Sau đó, dữ liệu này được nén lại. Tuy nhiên hiện nay trong SSL version 3 chưa mô tả cụ thể một phương pháp nén nào nên mặc định xem như là không nén.

Bước tiếp theo giá trị MAC của khối dữ liệu nén được tính theo công thức sau:

$\text{hash}(\text{MAC\_key} || \text{pad\_2} || \text{hash}(\text{MAC\_key} || \text{pad\_1} || \text{seq\_num} || \text{type} || \text{length} || \text{data}))$   
trong đó:

- Hàm hash là hàm MD5 hay SHA-1
- MAC\_key: khóa tính MAC đã được client và server thống nhất trong phần bắt tay
- pad\_1: byte 0x36 (00110110) được lặp lại 48 lần (384 bit) đối với hàm hash MD5 và 40 lần (320 bit) đối với hàm hash SHA-1
- pad\_2: byte 0x5C (10101100) được lặp lại 48 lần đối với MD5 và 40 lần với SHA-1
- seq\_num: số thứ tự của khối dữ liệu
- type: loại khối dữ liệu (xem phần bên dưới)
- length: kích thước khối dữ liệu
- data: khối dữ liệu

Sau khi tính MAC xong, khối dữ liệu cùng với giá trị MAC được mã hóa bằng một thuật toán mã khối đã được lựa chọn trong giao thức bắt tay.

Cuối cùng một SSL header được gắn vào đầu khối dữ liệu. SSL header gồm các field sau:

- Content Type (1 byte): Ngoài việc truyền dữ liệu của giao thức HTTP, SSL Record Protocol còn được dùng để truyền dữ liệu của giao thức Handshake cũng như hai giao thức còn lại SSL Change Cipher Spec và SSL Alert. Giá trị

của field này dùng để xác định loại giao thức đang được sử dụng. Đối với giao thức giao thức Handshake dữ liệu được truyền thẳng không cần nén, tính MAC và mã hóa.

- Major Version (1 byte): số hiệu chính của phiên bản SSL. Với SSLv3 field này có giá trị là 3.
- Minor Version (1 byte): số hiệu phụ của phiên bản SSL. Với SSLv3 field này có giá trị là 0.
- Compressed Length (2 byte): kích thước tính bằng byte của khối dữ liệu sau bước nén.

SSL Handshake Protocol	SSL Change Cipher Spec Protocol	SSL Alert Protocol	HTTP
SSL Record Protocol			
TCP			
IP			

**Hình 7-8. Mối liên hệ giữa các giao thức con của SSL**

### 7.3.3 SSL Session và SSL Connection

Để tránh việc mỗi lần kết nối với server là client phải tiến hành giao thức bắt tay lại từ đầu, SSL đưa ra khái niệm Session và Connection. Có thể hình dung, khi bạn mở trình duyệt và kết nối đến trang chủ một website, là bạn tạo một session mới, còn khi bạn click vào các link để đi đến các trang web khác trong cùng website, là bạn tạo connection mới trong session đã có này. Do đó SSL chỉ cần thực hiện giao thức bắt tay khi tạo session, còn khi tạo mới connection, SSL sẽ giữ nguyên tất cả các phương pháp mã hóa đã được chọn, giữ nguyên giá trị “pre-master secret”. Lúc này SSL chỉ cần thay đổi hai giá trị ClientHello.Random và ServerHello.Random, sau đó tính lại các giá trị “master secret” và 2 khóa MAC, 2 khóa mã hóa và 2 IV. Và việc trao đổi dữ liệu trên connection mới đã có thể bắt đầu mà không phải thực hiện giao thức bắt tay lại từ đầu.

## 7.4 Giao thức bảo mật mạng cục bộ Keberos

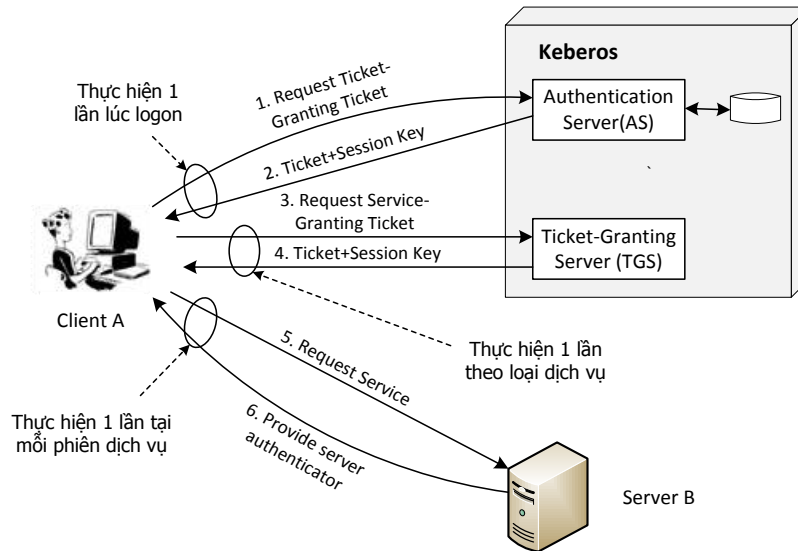
### 7.4.1 Keberos version 4.

Trong các phần trên, chúng ta đã tìm hiểu về chứng thực X.509 và giao thức SSL dùng để bảo mật dữ liệu truyền đi trên mạng Internet. Mỗi server trên internet đều có chứng chỉ X.509 và cơ chế xác thực mật khẩu người sử dụng để bảo đảm tính chứng thực của cả hai bên, đồng thời thiết lập khóa phiên để bảo mật dữ liệu.

Giao thức Keberos là một giao thức chứng thực sử dụng trong môi trường mạng quy mô nhỏ hơn như là mạng cục bộ LAN. Trong mạng LAN sử dụng trong các tổ chức và doanh nghiệp, cũng có các dịch vụ được cung cấp qua mạng như dịch vụ in ấn, dịch vụ chia sẻ file, cơ sở dữ liệu, email... Mỗi dịch vụ này đều cần chứng thực người sử dụng cũng như bảo mật. Dĩ nhiên là có thể dùng chứng thực X509. Tuy nhiên trong môi trường

mạng nhỏ như mạng LAN, giao thức Kerberos có thể được sử dụng như là một giải pháp thay thế.

Kerberos là giao thức chứng thực dựa trên khái niệm trung tâm phân phối khóa KDC (xem phần 3.9 và mô hình mở rộng chống replay attack trong chương 6), tức Kerberos chỉ dựa trên mã hóa đối xứng. Giao thức này do MIT chuẩn hóa. Mục đích của Kerberos là để trao đổi khóa phiên, thông qua đó đảm bảo tính bảo mật và tính chứng thực. Do nguyên tắc của Kerberos dựa trên KDC nên Kerberos cũng kế thừa được những ưu điểm của mô hình KDC như tính phi trạng thái. Hình dưới minh họa mô hình hoạt động của Kerberos version 4.



**Hình 7-9. Mô hình chứng thực và trao đổi khóa phiên Kerberos**

Trong mô hình trên, client A cần kết nối sử dụng dịch vụ tại server B. Authentication Server AS (chỉ có một AS) và Ticket-Granting Server TGS (có thể có nhiều TGS) đóng vai trò là các KDC. Server AS có nhiệm vụ cung cấp khóa đối xứng cho trao đổi giữa client A và server TGS. Server TGS có nhiệm vụ cung cấp khóa đối xứng cho trao đổi giữa client A và server dịch vụ B. Các người sử dụng A cần đăng ký mật khẩu  $K_A$  của mình với Server AS. Các server dịch vụ B đăng ký khóa bí mật  $K_B$  với Server TGS. Server TGS cũng đăng ký khóa bí mật  $K_{TGS}$  với Server AS. Quá trình phân phối khóa phiên  $K_{AB}$  để người sử dụng A kết nối với Server B trải qua ba giai đoạn như sau.

a) Giai đoạn đăng nhập: có hai thông điệp

1.  $A \rightarrow AS: ID_A || ID_{TGS} || TS_1$
2.  $AS \rightarrow A: E(K_{ATGS} || ID_{TGS} || TS_2 || Lifetime_2 || Ticket_{TGS}, K_A)$   
 $Ticket_{TGS} = E(K_{ATGS} || ID_A || AD_A || ID_{TGS} || TS_2 || Lifetime_2, K_{TGS})$

Trước tiên A sẽ gửi yêu cầu đến server AS, đề nghị cung cấp khóa phiên để kết nối với server TGS.  $ID_A$  và  $ID_{TGS}$  nhằm định danh client A và server TGS,  $TS_1$  là timestamp xác định thời điểm client A gửi yêu cầu. Sau đó server AS sẽ phát sinh khóa phiên  $K_{ATGS}$  này và mã hóa thành hai bản, một bản dành cho A (được mã hóa bởi  $K_A$ ) và một bản dành cho TGS (được mã hóa bởi  $K_{TGS}$ ). Tuy nhiên bản dành cho TGS được giao cho A quản lý và được gọi là Ticket-Granting Ticket (TGT). A sẽ

dùng ticket này để thiết lập kết nối với TGS.  $TS_2$  là timestamp xác định thời điểm cấp thẻ,  $Lifetime_2$  là thời hạn hiệu lực của thẻ này.  $AD_A$  là địa chỉ mạng của client A, yếu tố này dùng để chống lại phá hoại replay attack.

b) Giai đoạn đăng ký sử dụng dịch vụ:

3.  $A \rightarrow TGS: ID_B || Ticket_{TGS} || Authenticator$   
 $Authenticator = E(ID_A || AD_A || TS_3, K_{ATGS})$
4.  $TGS \rightarrow A: E(K_{AB} || ID_B || TS_4 || Ticket_B, K_{ATGS})$   
 $Ticket_B = E(K_{AB} || ID_A || AD_A || ID_V || TS_4 || Lifetime_4, K_B)$

Sau khi được cấp ticket TGT và khóa phiên  $K_{ATGS}$  để trao đổi với server TGS, client A gửi ticket này cho server TGS cùng với một authenticator để TGS chứng thực client A. Trong thông điệp này client cũng yêu cầu TGS cấp khóa phiên để kết nối với server dịch vụ B.  $ID_B$  nhằm xác định server dịch vụ này.  $TS_3$  là timestamp xác định thời điểm A sử dụng  $K_{ATGS}$  (chống replay attack).

Sau khi giải mã ticket, TGS có được khóa phiên  $K_{ATGS}$ . Từ đó TGS có thể kiểm tra tính chứng thực của client A qua Authenticator. Sau đó TGS sẽ phát sinh khóa phiên  $K_{AB}$  và mã hóa thành hai bản, một bản dành cho A (được mã hóa bởi  $K_{ATGS}$ ) và một bản dành cho B (được mã hóa bằng  $K_B$ ). Tương tự như TGT, bản dành cho B cũng được giao cho A quản lý và được gọi là service ticket. A dùng ticket này trao đổi dữ liệu với B.

$TS_4$  và  $Lifetime_4$  là thời điểm hiệu lực và thời hạn hiệu lực của ticket này.

c) Giai đoạn sử dụng dịch vụ:

5.  $A \rightarrow B: Ticket_B || Authenticator$   
 $Authenticator = E(ID_A || AD_A || TS_5, K_{AB})$
6.  $B \rightarrow A: E(TS_5 + 1, K_{AB})$

Tương tự như ở thông điệp 3, sau khi được cấp service ticket và khóa phiên  $K_{AB}$  để trao đổi với server B, client A gửi ticket này cho server B cùng với một Authenticator để B chứng thực A (tương tự như authenticator để TGS chứng thực A). B giải mã ticket này để có được khóa phiên  $K_{AB}$  và từ đó B giải mã authenticator để kiểm tra tính chứng thực của A.  $TS_5$  là timestamp xác định thời điểm A sử dụng  $K_{AB}$  (chống replay attack)

Tiếp theo B có thể gửi lại  $TS_5+1$  cho A để A chứng thực B. Sau thông điệp này A và B có thể tiến hành trao đổi dữ liệu thông qua khóa phiên  $K_{AB}$ .

A có thể sử dụng  $Ticket_B$  để kết nối với server B nhiều lần trong thời hạn  $Ticket_B$  còn hiệu lực. Khi ticket này hết hạn, A có thể gửi lại yêu cầu mới cho TGS để TGS cấp ticket khác.

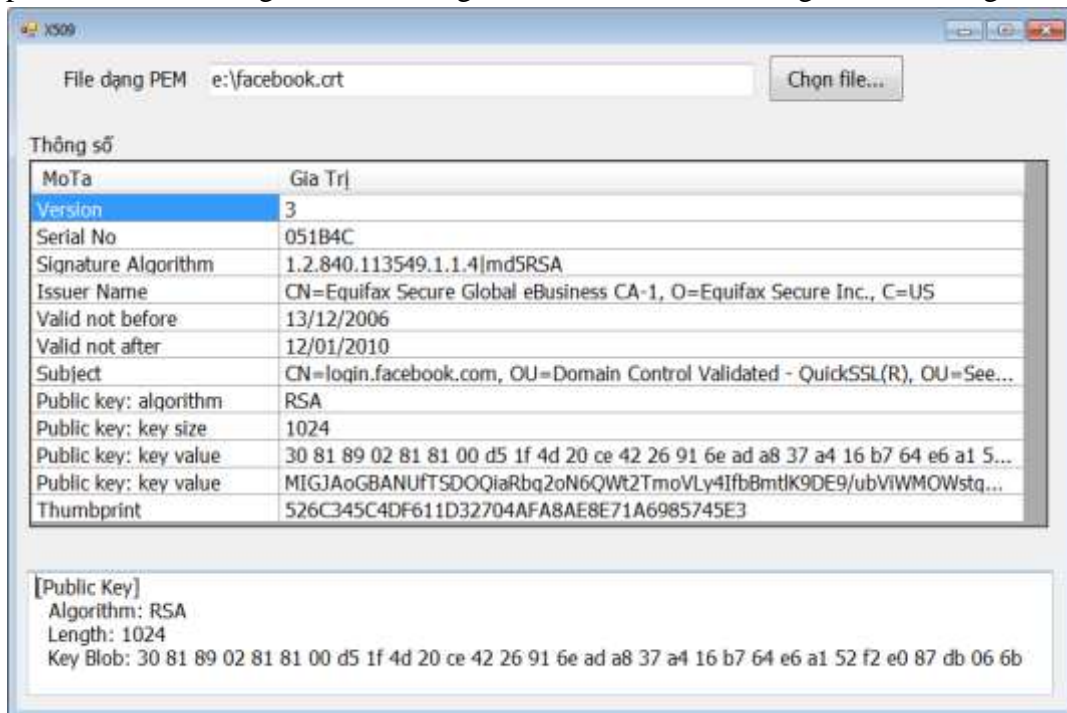
## 7.5 Câu hỏi ôn tập

1. Tại sao nếu Bob tin tưởng vào khóa công khai của trung tâm chứng thực X thì Bob có thể tin tưởng vào khóa công khai của Alice? (khóa này được nhúng trong chứng chỉ X.509 do X cấp cho Alice)

2. Trong giao thức SSL, client có cần cung cấp chứng chỉ X.509 cho server không?
3. Trong giao thức SSL, dữ liệu Web (HTML) được mã hóa dùng phương pháp mã hóa khóa công khai hay mã hóa đối xứng?
4. Giao thức SSL có thể bảo đảm dữ liệu truyền trên mạng. Vậy mục đích của giao thức Keberos là gì?

## 7.6 Bài tập thực hành

1. Tạo chứng chỉ X.509 theo các cách thức:
  - Dùng công cụ makecert của microsoft
  - Dùng công cụ openssl
  - Đăng ký tại Verisign
2. Lập trình xem nội dung của một chứng chỉ X509, trích khóa công khai từ chứng chỉ.



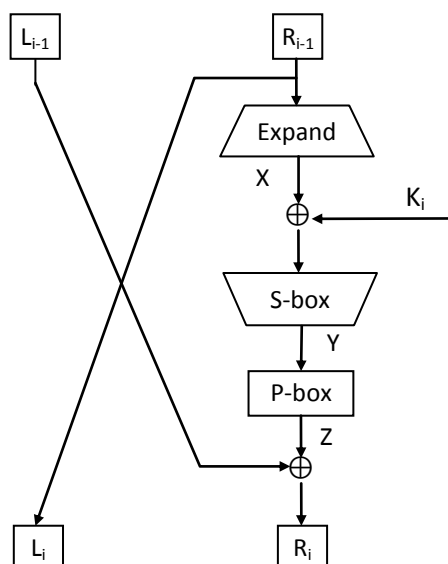
3. Cài đặt SSL cho web server Internet Information Server IIS
4. Cài đặt SSL cho web server Apache.

## CHƯƠNG 8. PHÁ MÃ VI SAI VÀ PHÁ MÃ TUYẾN TÍNH

Trong chương 3 chúng ta đã đề cập sơ lược đến ba cách thức phá mã DES. Chương này trình bày cách thức phá mã vi sai và phá mã tuyến tính. Việc tìm hiểu hai cách thức tấn công này giúp chúng ta hiểu rõ hơn về đặc điểm và cách thức xây dựng mã khối. Để đơn giản, chúng ta sẽ tìm hiểu phá mã TinyDES. Việc phá mã DES cũng thực hiện theo nguyên tắc tương tự.

### 8.1 Phá mã vi sai (Differential Cryptanalysis)

Trong chương 3, chúng ta đã tìm hiểu hiệu ứng lan truyền của mã DES, dưới tác động của các S-box và khóa K, chỉ cần thay đổi một bit trong bản rõ hay trong khóa sẽ dẫn đến sự thay đổi của nhiều bit trong các giá trị trung gian  $L_i R_i$  và trong bản mã. Do đó người phá mã khó phân tích được mối liên quan giữa bản rõ, bản mã và khóa – cho dù phá mã trong trường hợp known-plaintext hay chosen-plaintext.



Tuy nhiên, nếu xét dưới góc độ giá trị vi sai (differential) thì tác dụng lan truyền của khóa  $K$  và hàm S-box lại mất hiệu lực. Ta định nghĩa khái niệm vi sai như sau:

Giả sử hai giá trị  $X_1$  và  $X_2$  cùng số bit, thì vi sai giữa  $X_1$  và  $X_2$  là:  $X = X_1 \oplus X_2$

Tính chất của giá trị vi sai qua các phép biến đổi:

- 1) Phép XOR với giá trị khóa:

$$\text{Cho } Y_1 = X_1 \oplus K$$

$$Y_2 = X_2 \oplus K$$

$$\text{thì: } Y_1 \oplus Y_2 = X_1 \oplus X_2$$

như vậy input XOR bằng output XOR, điều đó có nghĩa là giá trị vi sai không chịu tác động của khóa. Đây là yếu tố quan trọng của phá mã vi sai.

- 2) Phép P-box:

$$\text{Cho } Y_1 = P\text{-box}(X_1)$$

$$Y_2 = P\text{-box}(X_2)$$

thì:  $Y_1 \oplus Y_2 = P\text{-box}(X_1 \oplus X_2)$

điều đó có nghĩa là nếu vi sai của đầu vào (*input XOR*) là cố định thì vi sai của đầu ra (*output XOR*) cũng cố định. Phép biến đổi *P-box* là tuyến tính, ứng với mỗi giá trị đầu vào có 1 giá trị đầu ra và ngược lại.

### 3) Phép *Expand*:

Cho  $Y_1 = \text{Expand}(X_1)$

$Y_2 = \text{Expand}(X_2)$

thì:  $Y_1 \oplus Y_2 = \text{Expand}(X_1 \oplus X_2)$

tương tự như hàm *P-box*, trong hàm *Expand*, nếu *input XOR* là cố định thì *output XOR* cũng cố định. Hàm *Expand* cũng là phép biến đổi tuyến tính.

### 4) Phép *S-box*

Xét *S-box* của mã TinyDES (cũng là hộp *S1* của mã DES) với 6 bit đầu vào và 4 bit đầu ra:

Cho  $Y_1 = S\text{-box}(X_1)$

$Y_2 = S\text{-box}(X_2)$

Trong trường hợp này *output XOR*  $Y_1 \oplus Y_2$  không cố định và *S-box* không phải là phép biến đổi tuyến tính. Ví dụ, xét bảng dưới đây trong trường hợp *input XOR* là 000001:

$X_1$	$X_2$	$X_1 \oplus X_2$	$Y_1$	$Y_2$	$Y_1 \oplus Y_2$
000000	000001	000001	1110	0000	1110
001000	001001	000001	0010	1110	1100
100000	100001	000001	0100	1111	1011
...	...	...	...	...	...

Ứng với mỗi giá trị của  $X_1$  thì có một  $X_2$  tương ứng để giá trị *XOR* là 000001. Do đó bảng trên có  $2^6 = 64$  dòng tương ứng với 64 cặp  $(X_1, X_2)$ . Tương tự như vậy đối với các *input XOR* khác. Dù rằng ứng với cùng một *input XOR* thì các giá trị *output XOR* là khác nhau, nhưng nếu xét dưới góc độ thống kê thì vẫn tồn tại mối quan hệ giữa *input XOR* và *output XOR*, điều đó được thể hiện qua bảng sau:



Input XOR (6 bit)	Output XOR (4 bit)															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	64			6		2	4	4		10	12	4	10	6	2	4
1				8		4	4	4		6	8	6	12	6	4	2
2	14	4	2	2	10	6	4	2	6	4	4		2	2	2	
3				6		10	10	6		4	6	4	2	8	6	2
4	4	8	6	2	2	4	4	2		4	4		12	2	4	6
5		4	2	4	8	2	6	2	8	4	4	2	4	2		12
6	2	4	10	4		4	8	4	2	4	8	2	2	2	4	4
7																
8				12		8	8	4		6	2	8	8	2	2	4
9	10	2	4		2	4	6		2	2	8		10		2	12
A		8	6	2	2	8	6		6	4	6		4		2	10
B	2	4		10	2	2	4		2	6	2	6	6	4	2	12
C				8		6	6			6	6	4	6		14	2
D	6	6	4	8	4	8	2	6		6	4	6		2		2
E		4	8	8	6	6	4		6	6	4			4		8
F	2		2	4	4	6	4	2	4	8	2	2	2	6	8	8
10							2	14		6	6	12	4	6	8	6
11	6	8	2	4	6	4	8	6	4		6	6		4		
12		8	4	2	6	6	4	6	6	4	2	6	6		4	
13	2	4	4	6			4	6	2		6	8	4	6	4	6
14		8	8		10		4	2	8	2	2	4		8	4	
15		4	6	4	2	2	4	10	6	2		10		4	6	4
16		8	10	8		2	2	6	10	2		2		6	2	6
17	4	4	6		10	6		2	4	4	4	6	6	6	2	
18		6	6		8	4	2	2	2	4	6	8	6	6	2	2
19	2	6	2	4		8	4	6	10	4		4	2	8	4	
1A		6	4		4	6	6	6	6	22	2		4	4	6	8
1B	4	4	2	4	10	6	6	4	6	2	2	4		2	4	2
1C		10	10	6	6			12	6	4			2	4	4	
1D	4	2	4		8			2	12		2	6	6	6	14	
1E		2	6		14	2		6	4	10		8	2	2	6	2
1F	2	4	10	6	2	2	2	8	6	8				4	6	4
20				10		12	8	2		6	4	4	4	2		12
21		4	2	4	4	8	10		4	4	10		4		2	8
22	10	4	6	2	2	8	2	2	2	2	6		4		4	10
23		4	4	8		2	6		6	6	2	10	2	4		10
24	12			2	2	2	2		14	14	2		2	6	2	4
25	6	4	4	12	4	4	4	10	2	2	2		4	2	2	2
26		4	4	10	10	10	2	4		4	6	4	4	4	2	
27	10	4	2		2	4	2		4	8		4	8	8	4	4
28	12	2	2	8	2	6	12			2	6		4		6	2
29	4	2	2	10		2	4			14	10		4	6		4
2A	4	2	4	6		2	8	2	2	14	2	2	6	6	2	2
2B	12	2	2	2	4	6	6	2		2	6	2	6		8	4
2C	4	2	2	4		2	10	4	2	2	4	8	8	4	2	6
2D	6	2	6	2	8	4	4	4	2	4	6		8	2		6
2E	6	6	2	2		2	4	6	4		6	2	12	2	6	4
2F	2	2	2	2	2	6	8	8	2	4	4	6	8	3	4	3
30		4	6		12	6	2	2	8	2	4	4	6	2	2	4
31	4	8	2	10	2	2	2	2	6			2	2	4	10	8
32	4	2	6	4	4	2	2	4	6	6	4	8	2	2	8	
33	4	4	6	2	10	8	4	2	4		2	2	4	6	2	4
34		8	16	6	2			12	6					8		6
35	2	2	4		8				14	4	6	8		2	14	
36	2	6	2	2	8		2	2	4	2	6	8	6	4	10	
37	2	2	12	4	2	4	4	10	4	4	2	6		2	2	4
38		6	2	2	2	2	2	4	6	4	4	4	4	6	10	10
39	6	2	2	4	12	6	4	8	4		2	4	2	4	4	
3A	6	4	6	4	6	8		6	2	2	6	2	2	6	4	
3B	2	6	4			2	4	6	4	6	8	6	4	4	6	2
3C		10	4		12	4	4	2	6		4	12	4	4	2	
3D		8	6	2	2	6		8	4	4		4		12	4	4
3E	4	8	2	2	2	4	4	14	4	2		2		8	4	4
3F	4	8	4	2	4		2	4	4	2	4	8	8	6	2	2

Trong bảng trên tổng của mỗi dòng là 64 (là số cặp  $X_1, X_2$  ứng với *input XOR* tương ứng), tuy nhiên số 64 này không phân bố đều trên các *output XOR*. Ta có kết luận về giá trị vi sai của *S-box* này như sau:

- Nếu *input XOR* là 00 thì *output XOR* chắc chắn là 0.
- Nếu *input XOR* là 10 thì *output XOR* là 7 với xác suất 14/64. Bảng dưới liệt kê các cặp đầu vào và đầu ra tương ứng

$X_1 \oplus X_2 = 10$		$Y_1 \oplus Y_2 = 7$	
$X_1$	$X_2$	$Y_1$	$Y_2$
08	18	2	5
09	19	E	9
0B	1B	2	5
23	33	C	B
24	34	E	9
2C	3C	2	5
2D	3D	1	6

- Nếu *input XOR* là 34 thì *output XOR* là 2 với xác suất 16/64. Bảng dưới liệt kê các cặp đầu vào và đầu ra tương ứng

$X_1 \oplus X_2 = 34$		$Y_1 \oplus Y_2 = 2$	
$X_1$	$X_2$	$Y_1$	$Y_2$
04	30	D	F
05	31	7	5
0E	3A	8	A
11	25	A	8
12	26	A	8
14	20	6	4
1A	2E	9	B
1B	2F	5	7

Từ đó ta có kết luận sau về giá trị vi sai của hàm  $F$  trong TinyDES, với *input XOR* và *output XOR* của hàm  $F$  là 4 bit:

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

- Nếu *input XOR* của  $F$  là 0:  $\rightarrow$  *output XOR* của *Expand* là 0 (6 bit)  $\rightarrow$  *input XOR* của *S-box* là 0 (khóa không ảnh hưởng đến vi sai)  $\rightarrow$  *input XOR* của *P-box* (4 bit) chắc chắn là 0  $\rightarrow$  *output XOR* của  $F$  chắc chắn là 0.
- Nếu *input XOR* của  $F$  là 3:  $\rightarrow$  *output XOR* của *Expand* là 34  $\rightarrow$  *input XOR* của *P-box* (4 bit) là 2 với xác suất 16/64  $\rightarrow$  *output XOR* của  $F$  là 8 với xác suất 16/64 = 1/4.
- Nếu *input XOR* của  $F$  là 1:  $\rightarrow$  *output XOR* của *Expand* là 10  $\rightarrow$  *input XOR* của *P-box* (4 bit) là 7 với xác suất 14/64  $\rightarrow$  *output XOR* của  $F$  là B với xác suất 14/64 = 7/32.

Như vậy, dù không biết giá trị của khóa K nhưng ta vẫn có thể tin được sự lan truyền của giá trị vi sai qua 3 vòng của TinyDES. Xét ví dụ sau:

Chọn  $P = (L_0, R_0)$  và  $\bar{P} = (\bar{L}_0, \bar{R}_0)$  sao cho vi sai  $P \oplus \bar{P} = 83$  (83: số thập lục phân). Quá trình lan truyền vi sai qua các vòng TinyDES được thể hiện trong bảng bên dưới.

$P = (L_0, R_0)$	$\bar{P} = (\bar{L}_0, \bar{R}_0)$	$(L_0, R_0) \oplus (\bar{L}_0, \bar{R}_0) = 83$	Xác suất
$L_1 = R_0$ $R_1 = L_0 \oplus F(R_0, K_1)$	$\bar{L}_1 = \bar{R}_0$ $\bar{R}_1 = \bar{L}_0 \oplus F(\bar{R}_0, K_1)$	$(L_1, R_1) \oplus (\bar{L}_1, \bar{R}_1) = 30$	1/4
$L_2 = R_1$ $R_2 = L_1 \oplus F(R_1, K_2)$	$\bar{L}_2 = \bar{R}_1$ $\bar{R}_2 = \bar{L}_1 \oplus F(\bar{R}_1, K_2)$	$(L_2, R_2) \oplus (\bar{L}_2, \bar{R}_2) = 03$	$(1/4) \times 1$
$L_3 = R_2$ $R_3 = L_2 \oplus F(R_2, K_3)$	$\bar{L}_3 = \bar{R}_2$ $\bar{R}_3 = \bar{L}_2 \oplus F(\bar{R}_2, K_3)$	$C \oplus \bar{C} =$ $(L_3, R_3) \oplus (\bar{L}_3, \bar{R}_3) = 38$	$(1/4) \times (1/4)$

Như vậy vi sai của bản mã là 38 với xác suất 1/16. Điều đó có nghĩa là trung bình trong 16 cặp bản rõ có vi sai là 83 thì sẽ tìm thấy 1 cặp có vi sai bản mã là 38.

Với 1 khóa K cụ thể nào đó, giả sử ta thực hiện chosen-plaintext cho 16 cặp (bản rõ, bản mã) và tìm thấy 1 cặp sau:

$$P = (L_0, R_0) = 2B ; C = (L_3, R_3) = E5$$

$$\bar{P} = (\bar{L}_0, \bar{R}_0) = A8 ; \bar{C} = (\bar{L}_3, \bar{R}_3) = DD$$

(có vi sai bản rõ là 83 và vi sai bản mã là 38)

Như vậy, tại vòng thứ 3, input của hàm F trong hai trường hợp tương ứng là  $R_2 = L_3 = E$  và  $\bar{R}_2 = \bar{L}_3 = D$ . Do đó output của hàm Expand trong 2 trường hợp là 2F và 1B. Vì input XOR và output XOR của S-box trong vòng thứ 3 này là 34 và 2. Tra bảng, ta có các khóa  $K_3$  có thể có là:

$X_1 \oplus X_2 = 34$		$K_3$
$X_1 = 2F \oplus K_3$	$X_2 = 1B \oplus K_3$	
04	30	2B
05	31	2A
0E	3A	21
11	25	3E
12	26	3D
14	20	3B
1A	2E	35
1B	2F	34

$X_1 \oplus X_2 = 34$		$K_3$
$X_1 = 1B \oplus K_3$	$X_2 = 2F \oplus K_3$	
04	30	1F
05	31	1E
0E	3A	15
11	25	0A
12	26	09
14	20	0F
1A	2E	01
1B	2F	00

Tương tự, chọn  $P = (L_0, R_0)$  và  $\bar{P} = (\bar{L}_0, \bar{R}_0)$  sao cho vi sai  $P \oplus \bar{P} = B1$ . Quá trình lan truyền vi sai qua các vòng TinyDES được thể hiện trong bảng bên dưới.

$P = (L_0, R_0)$	$\bar{P} = (\bar{L}_0, \bar{R}_0)$	$(L_0, R_0) \oplus (\bar{L}_0, \bar{R}_0) = B1$	Xác suất
$L_1 = R_0$ $R_1 = L_0 \oplus F(R_0, K_1)$	$\bar{L}_1 = \bar{R}_0$ $\bar{R}_1 = \bar{L}_0 \oplus F(\bar{R}_0, K_1)$	$(L_1, R_1) \oplus (\bar{L}_1, \bar{R}_1) = 10$	7/32
$L_2 = R_1$ $R_2 = L_1 \oplus F(R_1, K_2)$	$\bar{L}_2 = \bar{R}_1$ $\bar{R}_2 = \bar{L}_1 \oplus F(\bar{R}_1, K_2)$	$(L_2, R_2) \oplus (\bar{L}_2, \bar{R}_2) = 01$	$(7/32) \times 1$
$L_3 = R_2$ $R_3 = L_2 \oplus F(R_2, K_3)$	$\bar{L}_3 = \bar{R}_2$ $\bar{R}_3 = \bar{L}_2 \oplus F(\bar{R}_2, K_3)$	$C \oplus \bar{C} =$ $(L_3, R_3) \oplus (\bar{L}_3, \bar{R}_3) = 1B$	$(7/32)^2 \approx 0.048$

Như vậy vi sai của bản mã là 1B với xác suất 0.048. Điều đó có nghĩa là trung bình trong 21 cặp bản rõ có vi sai là B1 thì sẽ tìm thấy 1 cặp có vi sai bản mã là 1B.

Với 1 khóa K cụ thể nào đó, giả sử ta thực hiện chosen-plaintext cho 21 cặp (bản rõ, bản mã) và tìm thấy 1 cặp sau:

$$P = (L_0, R_0) = 3A ; C = (L_3, R_3) = 83$$

$$\bar{P} = (\bar{L}_0, \bar{R}_0) = 8B ; \bar{C} = (\bar{L}_3, \bar{R}_3) = 98$$

(có vi sai bản rõ là B1 và vi sai bản mã là 1B)

Tại vòng thứ 3, input của hàm F trong hai trường hợp tương ứng là  $R_2 = L_3 = 8$  và  $\bar{R}_2 = \bar{L}_3 = 9$ . Do đó output của hàm Expand trong 2 trường hợp là 01 và 11. Vì input XOR và output XOR của S-box trong vòng thứ 3 này là 10 và 7. Tra bảng, ta có các khóa  $K_3$  có thể có là:

$X_1 \oplus X_2 = 10$		$K_3$
$X_1 = 01 \oplus K_3$	$X_2 = 11 \oplus K_3$	
08	18	09
09	19	08
0B	1B	0A
23	33	22
24	34	25
2C	3C	2D
2D	3D	2C

$X_1 \oplus X_2 = 10$		$K_3$
$X_1 = 11 \oplus K_3$	$X_2 = 01 \oplus K_3$	
08	18	19
09	19	18
0B	1B	1A
23	33	32
24	34	35
2C	3C	3D
2D	3D	3C

Kết hợp 2 bảng, thì  $K_3$  phải là giá trị thuộc tập  $\{ 09, 0A, 35, 3D \}$ . Gọi 8 bit của khóa K là  $k_0k_1k_2k_3k_4k_5k_6k_7$ , thì 6 bit của  $K_3$  là  $k_5k_1k_3k_2k_7k_0$ . Như vậy với từng trường hợp của  $K_3$  chúng ta có thể thử các giá trị của  $k_4$  và  $k_6$ . Ví dụ giả sử  $K_3$  là 09 (nhị phân 001001), như vậy khóa K có dạng 1001x0x0, và khóa K có 4 trường hợp: 10010000, 10010010, 10011000, 10011010. Lần lượt mã hóa bản rõ 2B với 4 trường hợp trên của khóa K, thì chỉ có trường hợp K = 1001.1010 cho ra bản rõ E5.

Như vậy thay vì vét cạn 256 trường hợp của khóa K, chúng ta chỉ cần thử  $16+21=37$  cặp bản rõ-bản mã, sau đó thử thêm 16 trường hợp của khóa K thì tìm ra được giá trị chính xác của K. Điều này chứng tỏ phương pháp phá mã vi sai là có hiệu quả để phá mã TinyDES.

## 8.2 Phá mã tuyến tính (Linear Cryptanalysis)

Trong phần mã TinyDES, chúng ta đã nói S-box là một cấu trúc phi tuyến, tức với  $Y=S\text{-}box(X)$  thì giữa  $X$  và  $Y$  không có mối liên hệ toán học. Tuy nhiên nếu chỉ xét một số bit của  $X$  và  $Y$  lại bộc lộ một số quan hệ tuyến tính.

Chúng ta ký hiệu các bit của  $X$  và  $Y$  như sau:

$$y_0y_1y_2y_3 = Sbox(x_0x_1x_2x_3x_4x_5)$$

Gọi  $a$  là các giá trị từ 1 đến 64 và  $b$  là các giá trị từ 1 đến 15,  $a_0a_1a_2a_3a_4a_5$  và  $b_0b_1b_2b_3$  là biểu diễn nhị phân tương ứng của  $a$  và  $b$ . Với một  $a$  và  $b$  cụ thể, tính:

$$LX(X, a) = \bigoplus x_i \text{ nếu } a_i = 1$$

$$LY(Y, b) = \bigoplus y_i \text{ nếu } b_i = 1$$

Với 64 trường hợp của  $Y=S\text{-box}(X)$ , ta định nghĩa số  $S(a, b)$  như sau:

$$S(a, b) \text{ là số trường hợp mà } LX(X, a) = LY(Y, b)$$

Bảng bên dưới liệt kê các giá trị  $S(a, b) - 32$  với  $a$  từ 1 đến 32 và  $b$  từ 1 đến 15.

a	b														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	-2	-2	-4	-2	0	-4	6	2	0	0	6	4	-2	-6	4
3	-2	-2	-4	-2	0	-4	6	2	8	0	-2	4	6	-6	-4
4	2	-2	-4	-2	0	-4	-6	-2	4	8	2	0	-2	-6	12
5	-2	-2	0	-2	-4	-4	-2	2	-4	-4	2	4	-10	-2	-4
6	0	0	4	0	4	0	0	0	-4	4	4	0	0	-4	-8
7	-4	0	8	0	0	0	4	4	-4	-8	-4	4	0	0	0
8	4	-2	6	-6	-6	0	-4	-4	-4	2	-2	2	-2	0	0
9	0	6	-6	-2	-6	4	-4	0	-4	-2	6	2	-6	0	-4
10	-2	0	2	0	6	8	2	-2	0	-2	4	-2	0	-2	4
11	2	-8	-2	-4	-10	4	2	-6	8	2	4	-2	-4	-2	0
12	-2	0	6	0	2	0	2	2	0	6	-4	2	-4	6	0
13	6	0	6	4	-2	-4	-2	2	0	6	4	-2	8	-6	-4
14	0	-2	-2	2	2	0	0	4	4	6	-2	2	2	-4	4
15	0	-2	6	-2	-2	4	-4	-4	-4	-2	-2	-2	-2	0	0
16	2	2	0	-2	0	4	-6	0	6	2	-4	6	-4	-4	-18
17	2	-2	-4	2	-4	-4	10	-4	2	2	-4	-2	-4	0	-6
18	4	0	0	-4	4	0	4	-6	2	2	6	2	6	6	-10
19	4	-4	-4	0	0	-8	-12	-2	-2	-6	6	2	6	2	2
20	4	0	4	-8	-4	4	0	2	6	-2	2	6	2	-2	2
21	0	4	-4	-4	4	4	-4	10	2	2	2	-6	2	6	-2
22	6	2	0	2	-4	0	2	4	2	2	0	-2	0	0	2
23	2	6	-8	6	4	0	-2	-12	-2	-2	0	-6	0	0	-2
24	2	8	2	0	6	4	2	4	-2	4	6	0	-2	-4	2
25	-2	4	-6	0	-6	0	2	4	-6	8	6	0	2	0	-6
26	0	-6	2	-2	-2	4	4	-2	-2	0	0	-4	4	2	2
27	4	6	2	-10	2	-8	4	-2	-6	4	0	4	0	-2	2
28	-4	2	2	2	-6	0	-4	-2	-2	4	0	0	4	2	2
29	4	-2	-2	2	-6	-4	0	2	2	-4	0	-12	0	-6	-6
30	2	0	-2	4	-2	0	-2	0	6	-4	-2	0	-2	0	2
31	2	-4	2	-4	-2	4	2	4	-6	4	-2	-4	2	0	2
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Nếu  $S(a, b) = 32$ , thì các bit theo  $a$  của  $X$  và các bit theo  $b$  của  $Y$  không có mối quan hệ tuyến tính

Xét  $S(16, 15) = 14$ , điều này có nghĩa là:

$$y_0 \oplus y_1 \oplus y_2 \oplus y_3 = x_1 \text{ xảy ra với xác suất } 14/64$$

$$\text{hay } y_0 \oplus y_1 \oplus y_2 \oplus y_3 \neq x_1 \text{ xảy ra với xác suất } 50/64$$

ta viết lại mối quan hệ này  $Y[0,1,2,3] = X[1]$

Như vậy nếu xét  $Y=F(X, K)$  với  $F$  là hàm Feistel trong 1 vòng của mã TinyDES:

$$y_0 y_1 y_2 y_3 = F(x_0 x_1 x_2 x_3, k_0 k_1 k_2 k_3 k_4 k_5)$$

Thì ta có quan hệ tuyến tính sau:

$$Y[0,1,2,3] = X[3] \oplus K[1]$$

Bây giờ ta liên kết mối quan hệ này qua ba vòng của TinyDES

$P = (L_0, R_0)$		Xác suất
$L_1 = R_0$ $R_1 = L_0 \oplus F(R_0, K_1)$	$R_1[0,1,2,3] \oplus L_0[0,1,2,3] = R_0[3] \oplus K_1[1]$	14/64

$L_2 = R_1$ $R_2 = L_1 \oplus F(R_1, K_2)$		
$L_3 = R_2$ $R_3 = L_2 \oplus F(R_2, K_3)$	$R_3[0,1,2,3] \oplus L_2[0,1,2,3] = R_2[3] \oplus K_3[1]$ $\Leftrightarrow R_3[0,1,2,3] \oplus R_1[0,1,2,3] = L_3[3] \oplus K_3[1]$	14/64
$C = (L_3, R_3)$		

Suy ra:

$$R_3[0,1,2,3] \oplus L_0[0,1,2,3] = R_0[3] \oplus L_3[3] \oplus K_0[1] \oplus K_3[1]$$

hay

$$K_0[1] \oplus K_3[1] = R_3[0,1,2,3] \oplus L_0[0,1,2,3] \oplus R_0[3] \oplus L_3[3]$$

Phương trình trên thỏa mãn với xác suất  $(14/64)^2 + (1-14/64)^2 = 0.66$ . Giả sử ta phá mã known-plaintext với 100 cặp bản rõ-bản mã và có được kết quả sau:

$$R_3[0,1,2,3] \oplus L_0[0,1,2,3] \oplus R_0[3] \oplus L_3[3] = 1 \text{ xuất hiện 66 lần}$$

$$R_3[0,1,2,3] \oplus L_0[0,1,2,3] \oplus R_0[3] \oplus L_3[3] = 0 \text{ xuất hiện 34 lần}$$

Thì ta có thể kết luận  $K_0[1] \oplus K_3[1] = 1$  hay  $k_1 \oplus k_2 = 1$  với  $k_1, k_2$  là bit thứ 1 và thứ 2 trong khóa  $K$  ban đầu.

Như vậy ta đã biết được mối quan hệ giữa hai bit  $k_1, k_2$  của khóa  $K$  ban đầu, điều này giúp ta chỉ cần tìm trong 128 giá trị của khóa  $K$  mà thôi. Điều này chứng tỏ phương pháp phá mã tuyến tính là có hiệu quả để phá mã TinyDES hơn là phương pháp vét cạn khóa.

### 8.3 Kết luận về nguyên tắc thiết kế mã khối.

Vì chúng ta không thể chứng minh về mặt lý thuyết là mã khối có an toàn tuyệt đối hay không nên các nhà nghiên cứu tìm cách chống lại các hình thức phá mã đã biết. Để chống lại hai hình thức phá mã vi sai và tuyến tính, một số nguyên tắc sau được đặt ra.

- 1) Tăng số vòng mã hóa: vì phá mã vi sai và tuyến tính thực hiện theo xác suất nên số vòng càng nhiều thì xác suất càng giảm.
- 2) Cải thiện hàm S-box: hàm S-box phải được cải tiến sao cho dấu vết vi sai và dấu vết tuyến tính càng ít càng tốt (xác suất vi sai và xác suất tuyến tính giảm).
- 3) Cải thiện việc xáo trộn (mix) kết quả trung gian từ vòng này qua vòng khác, thể hiện ở các hàm Expand và P-box. Việc xáo trộn tốt hơn sẽ làm cho việc liên kết vi sai và liên kết tuyến tính giữa các vòng giảm đi.

Trong chương tiếp theo, chúng ta sẽ tìm hiểu về mã hóa AES. Mã hóa này thực hiện rất tốt hàm S-box và P-box nên mã AES chỉ cần thực hiện 10 vòng so với 16 vòng của mã DES.

## CHƯƠNG 9. ADVANCED ENCRYPTION STANDARD – AES

### 9.1 Nhóm, vành, trường

Nhóm, vành, trường các yếu tố cơ bản của một ngành toán học gọi là đại số trừu tượng (abstract algebra). Trong ngành toán này, chúng ta quan tâm đến một tập các phần tử, cách thức kết hợp phần tử thứ nhất và phần tử thứ hai để tạo thành một phần tử thứ ba (giống như trong số học thường ta dùng phép cộng và phép nhân áp dụng trên hai số cho ra kết quả số thứ ba)

#### 9.1.1 Nhóm (Group)

Một nhóm, ký hiệu là  $\{G, \circ\}$ , là một tập  $G$  các phần tử và một phép kết hợp 2 ngôi  $\circ$  thỏa mãn các điều kiện sau:

A1) *Tính đóng*:  $\forall a, b \in G: a \circ b \in G$

A2) *Tính kết hợp*:  $\forall a, b, c \in G: (a \circ b) \circ c = a \circ (b \circ c)$

A3) *Phần tử đơn vị*:  $\exists e \in G: a \circ e = e \circ a = a, \forall a \in G$

A4) *Phần tử nghịch đảo*:  $\forall a \in G, \exists ! a' \in G: a \circ a' = a' \circ a = e$

Ví dụ 1: Dễ thấy tập số nguyên  $\mathbb{Z}$  và phép cộng số nguyên là một nhóm. Phần tử đơn vị là 0. Với  $a \in \mathbb{Z}$  thì nghịch đảo của  $a$  là  $-a$ . Tập  $\mathbb{Z}$  có vô hạn phần tử nên nhóm này được gọi là *nhóm vô hạn*.

Ví dụ 2: xét một tập  $S$  gồm  $n$  số nguyên  $\{1, 2, \dots, n\}$ . Định nghĩa tập  $T$  có các phần tử là các hoán vị của tập  $S$ .

Ví dụ  $n = 4$ , như vậy  $\{1, 2, 3, 4\} \in T, \{3, 2, 1, 4\} \in T, \dots$ . Tập  $T$  có  $4! = 24$  phần tử.

Tiếp theo, định nghĩa phép kết hợp  $\circ$  như sau:  $c = a \circ b$  là một hoán vị của  $a$  theo thứ tự trong  $b$ . Ví dụ:  $a = \{2, 3, 4, 1\}, b = \{3, 2, 4, 1\}$ . Hoán vị của  $a$  theo  $b$  là  $\{4, 3, 1, 2\}$ .  $c$  cũng là phần tử thuộc  $T$  nên thỏa tính chất A1.

Nếu chọn  $e = \{1, 2, 3, 4\}$  thì  $a \circ e$  không làm thay đổi thứ tự của  $a$ , còn  $e \circ a$  sẽ hoán vị  $e$  trở thành  $a$ . Vì vậy  $\{1, 2, 3, 4\}$  là phần tử đơn vị theo tính chất A3.

Ta cũng có thể chứng minh tập  $T$  và phép hoán vị thỏa mãn hai tính chất còn lại A2 và A4. Nghĩa là  $T$  và phép hoán vị tạo thành một nhóm. Tập  $T$  có hữu hạn phần tử nên nhóm này được gọi là *nhóm hữu hạn*.

Một nhóm được gọi là nhóm Abel nếu có thêm tính chất sau:

A5) *Tính giao hoán*:  $\forall a, b \in G: a \circ b = b \circ a$

Dễ thấy tập  $\mathbb{Z}$  là nhóm Abel trên phép cộng. Còn tập  $T$  và phép hoán vị không phải là nhóm Abel với  $n > 2$

#### Nhóm vòng:

Cho nhóm  $\{G, \circ\}$ , ta định nghĩa phép lũy thừa như sau:

$$a^k = a \circ a \circ \dots \circ a \text{ (lặp } k \text{ lần } a \text{ với } k \text{ là số nguyên dương)}$$

$$a^{-k} = (a')^k$$

$$a^0 = e$$

Ví dụ:  $a^3 = a \circ a \circ a$ .

Ta gọi  $G$  là nhóm vòng nếu mọi phần tử của  $G$  đều biểu diễn được dưới dạng  $a^k$  với  $a$  thuộc  $G$  và  $k$  là một số nguyên. Lúc này  $a$  được gọi là *phần tử sinh* của tập  $G$ .

Ví dụ tập  $\mathbb{Z}$  là một nhóm vòng với  $a$  là 1:  $5 = 1^5, -4 = (-1)^4$

Mọi nhóm vòng đều có tính giao hoán nên đều là nhóm Abel.

### 9.1.2 Vành (Ring)

Một vành  $R$ , ký hiệu  $\{R, +, \times\}$ , là một tập các phần tử và hai phép kết hợp 2 ngôi, gọi là *phép cộng* và *phép nhân*, nếu các tính chất sau được thỏa mãn:

*A1-A5)  $R$  là một nhóm Abel theo phép cộng:*  $R$  thỏa mãn các tính chất từ A1 đến A5, ta ký hiệu phần tử đơn vị là 0 và phần tử nghịch đảo của  $a$  trong phép cộng là  $-a$ . Ta định nghĩa phép trừ là  $a - b = a + (-b)$

*M1) Tính đóng đối với phép nhân:*  $\forall a, b \in R: ab \in R$  (viết tắt thay cho dấu  $\times$ )

*M2) Tính kết hợp đối với phép nhân:*  $\forall a, b, c \in R: (ab)c = a(bc)$

*M3) Tính phân phối giữa phép cộng và phép nhân:*  $\forall a, b, c \in R$

$$(a + b)c = ac + bc$$

$$a(b + c) = ab + ac$$

Ngắn gọn, trong một vành, chúng ta có thể thực hiện các phép cộng, trừ, nhân mà không ra khỏi vành (kết quả các phép toán cộng, trừ, nhân thuộc  $R$ )

Ví dụ: cho tập các ma trận vuông cấp  $n$  với số thực, các phép cộng và nhân ma trận tạo thành một vành.

Một vành được gọi là vành giao hoán nếu có thêm tính giao hoán đối với phép nhân:

*M4) Tính giao hoán với phép nhân:*  $\forall a, b \in R: ab = ba$

Ví dụ: cho tập các số nguyên chẵn, với các phép cộng và nhân thông thường, tạo thành một vành giao hoán, tập ma trận vuông cấp  $n$  như trên không phải là vành giao hoán.

Một vành được gọi là miền nguyên (integral domain) nếu đó là vành giao hoán và có thêm hai tính chất sau:

*M5) Tồn tại phần tử đơn vị phép nhân:*  $a1 = 1a = a$

*M6) Liên quan giữa phép nhân và phần tử đơn vị phép cộng :*

$$\text{Nếu } ab = 0 \text{ thì } a = 0 \text{ hay } b = 0$$

### 9.1.3 Trường (Field)

Một trường, ký hiệu  $\{F, +, \times\}$ , là một tập các phần tử và hai phép kết hợp 2 ngôi, gọi là *phép cộng* và *phép nhân*, nếu các tính chất sau được thỏa mãn:

*A1-A5, M1-M6)  $F$  là một miền nguyên* (thỏa các tính chất A1 đến A5 và M1 đến M6)

*M7) Tồn tại phần tử nghịch đảo của phép nhân:*

$$\forall a \in F, a \neq 0, \exists a^{-1} \in F: aa^{-1} = 1$$

Ngắn gọn, trong một trường, chúng ta có thể thực hiện các phép cộng, trừ, nhân, chia mà không ra khỏi trường (kết quả các phép toán cộng, trừ, nhân, chia thuộc  $F$ ). Định nghĩa phép chia là:  $a/b = a(b^{-1})$



Ví dụ: tập các số thực với phép cộng và nhân thông thường là một trường. Tập các số nguyên không phải là trường vì không thực hiện được phép chia.


## 9.2 Số học modulo và trường hữu hạn GF(p)

Trong chương 4 chúng ta đã tìm hiểu về phép toán modulo. Dựa trên phép toán modulo, chúng ta xây dựng một tập  $Z_n$  như sau:

Cho một số nguyên  $n$ :  $Z_n = \{ 0, 1, 2, \dots, n-1 \}$

Tương tự như tập số nguyên  $Z$ , trên tập  $Z_n$  ta cũng định nghĩa các phép cộng và nhân như sau:  $\forall a, b, c \in Z_n$ :

- Phép cộng:  $c = a + b$  nếu  $c \equiv (a + b) \pmod n$ 



Phép cộng trong  $Z_n$

Phép cộng trong số học thường
- Phép nhân:  $c = a.b$  nếu  $c \equiv (a.b) \pmod n$

Dễ thấy rằng tập  $Z_n$  cùng với phép cộng trên thỏa mãn các tính chất của một nhóm Abel với phần tử đơn vị của phép cộng là 0 (các tính chất từ A1 đến A5).

Bên cạnh đó, tập  $Z_n$  cùng với phép cộng và phép nhân trên thỏa mãn các tính chất của một miền nguyên với phần tử đơn vị của phép nhân là 1 (các tính chất từ M1 đến M6).

Ví dụ, với  $n = 7$  thì phép nhân và phép cộng là như sau:

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Tuy nhiên không phải tập  $Z_n$  nào cũng thỏa tính chất M7, nghĩa là mọi phần tử khác 0 của  $Z_n$  phải có phần tử nghịch đảo của phép nhân. Chỉ có với những  $n$  là số nguyên tố thì  $Z_n$  mới thỏa tính chất M7. (xem khái niệm 6 trong phần Lý thuyết số chương 4). Ví dụ với  $n=8$  (không thỏa M7) và  $n=7$  (thỏa M7).

$a$	$-a$	$a^{-1}$
0	0	-
1	7	1
2	6	-
3	5	3
4	4	-
5	3	5
6	2	-
7	1	7

$a$	$-a$	$a^{-1}$
0	0	-
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

Ta cũng dùng thuật toán Euclid mở rộng để tìm phần tử nghịch đảo phép nhân trong tập  $Z_n$ .

Ví dụ phép chia:  $5/4 = 5(4^{-1}) = 5.2 = 3$ .

Như vậy với  $n$  là số nguyên tố, thì tập  $Z_n$  trở thành một trường hữu hạn mà ta gọi là trường Galois (tên nhà toán học đã tìm hiểu về trường hữu hạn này). Ta đổi ký hiệu  $Z_n$  thành  $Z_p$  với quy định  $p$  là số nguyên tố. Ký hiệu trường hữu hạn trên là  $GF(p)$

### 9.3 Số học đa thức và trường hữu hạn $GF(2^n)$

#### 9.3.1 Phép toán đa thức thông thường

Trong đại số, chúng ta định nghĩa một đa thức bậc  $n$  ( $n \geq 0$ ) dưới dạng

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = \sum_{i=0}^n a_i x^i$$

Trong đó các  $a_i \in R$ ,  $a_n \neq 0$  được gọi là các hệ số. Và ta cũng định nghĩa các phép cộng, trừ, nhân đa thức như sau:

$$\text{Cho } f(x) = \sum_{i=0}^n a_i x^i \quad g(x) = \sum_{i=0}^m b_i x^i$$

$$\text{Phép cộng: } f(x) + g(x) = \sum_{i=0}^{\max(m,n)} (a_i + b_i) x^i$$

$$\text{Phép nhân: } f(x) \times g(x) = \sum_{i=0}^{m+n} c_i x^i \quad \text{với } c_k = a_0 b_k + a_1 b_{k-1} + \dots + a_k b_0$$

$$\text{Phép trừ: } f(x) - g(x) = \sum_{i=0}^{\max(m,n)} (a_i - b_i) x^i$$

Trong 3 phép toán trên ta giả định  $a_i = 0$  nếu  $i > n$  và  $b_i = 0$  nếu  $i > m$ .

Phép chia đa thức  $f(x)$  cho  $g(x)$  cũng tương tự như phép chia trên số nguyên, gồm một đa thức thương  $q(x)$  và một đa thức dư  $r(x)$ .  $r(x)$  có bậc nhỏ hơn  $g(x)$

$$\frac{f(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)}$$

$$f(x) = q(x) \times g(x) + r(x)$$

$$\text{Ví dụ: } f(x) = x^3 + x^2 + 2, \quad g(x) = x^2 - x + 1$$

- $f(x) + g(x) = x^3 + 2x^2 - x + 3$
- $f(x) - g(x) = x^3 + x + 1$
- $f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$
- $f(x)/g(x)$ : đa thức phần thương  $q(x) = x + 2$  và đa thức phần dư  $r(x) = x$

Với các phép toán cộng và nhân như trên thì tập các đa thức (mỗi đa thức là một phần tử của tập) tạo thành một vành, với phần tử đơn vị của phép cộng là đa thức  $e(x) = 0$  và phần tử đơn vị của phép nhân là đa thức  $d(x) = 1$ .

Tuy nhiên tập các đa thức trên *không tạo thành một trường* vì không tồn tại phần tử nghịch đảo của phép nhân (nên phép chia 2 đa thức tồn tại phần dư).

### 9.3.2 Đa thức định nghĩa trên tập $Z_p$

Trong phần trên ta đã định nghĩa đa thức có các hệ số trong trường số thực (tập  $R$ ). Trong phần này ta sẽ xem xét tập các đa thức  $W_p$  có hệ số thuộc trường  $Z_p$ .

$$W_p = \left\{ f(x) = \sum_{i=0}^n a_i x^i \text{ với } n \geq 0, a_i \in Z_p, a_n \neq 0 \right\}$$

Trên tập  $W_p$  ta định nghĩa các phép cộng, trừ, nhân, chia như sau:

$$f(x) = \sum_{i=0}^n a_i x^i \quad g(x) = \sum_{i=0}^m b_i x^i$$

Phép cộng:  $f(x) + g(x) = \sum_{i=0}^{\max(m,n)} (a_i + b_i) x^i$

Phép nhân:  $f(x) \times g(x) = \sum_{i=0}^{m+n} c_i x^i$  với  $c_k = a_0 b_k + a_1 b_{k-1} + \dots + a_k b_0$

Phép trừ:  $f(x) - g(x) = \sum_{i=0}^{\max(m,n)} (a_i - b_i) x^i$

Phép chia:  $f(x) / g(x)$ : có đa thức thương là  $q(x)$  và đa thức dư là  $r(x)$

Trong đó các phép toán  $a_i + b_j$ ,  $a_i b_j$ ,  $a_i - b_j$ ,  $a_i / b_j$  được định nghĩa trong tập  $Z_p$

Ví dụ: xét trường  $Z_2 = \{0, 1\}$

$$f(x) = x^7 + x^5 + x^4 + x^3 + x + 1, \quad g(x) = x^3 + x + 1$$

- $f(x) + g(x) = x^7 + x^5 + x^4$
- $f(x) - g(x) = x^7 + x^5 + x^4$
- $f(x) \times g(x) = x^{10} + x^4 + x^2 + 1$
- $f(x)/g(x)$ :  $q(x) = x^4 + 1$  và  $r(x) = 0$ .

Trong ví dụ trên có thể xem  $g(x)$  và  $q(x)$  là đa thức ước số của đa thức  $f(x)$ .  $f(x) = g(x) \cdot q(x)$ . Những đa thức  $f(x)$  như vậy gọi là đa thức không tối giản. Đa thức tối giản là đa thức chỉ có ước số là đa thức 1 và chính nó (khái niệm tối giản tương tự như khái niệm số nguyên tố trong tập số tự nhiên).

Ví dụ (cũng xét trong trường  $Z_2$ ):

- $x^3 + x + 1$  là đa thức tối giản
- $x^4 + 1$  không phải là đa thức tối giản vì  $x^4 + 1 = (x + 1)(x^3 + x^2 + x + 1)$

Tương tự như khái niệm ước số chung lớn nhất của 2 số tự nhiên, chúng ta cũng có khái niệm ước số chung lớn nhất của 2 đa thức. Khái niệm lớn ở đây là bậc lớn, ví dụ  $x^3 + 1$  lớn hơn  $x^2 + x + 1$ .

Ví dụ: xét trong trường  $Z_2$ , USCLN của hai đa thức  $a(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$  và  $b(x) = x^4 + x^2 + x + 1$  là  $c(x) = x^3 + x^2 + 1$

Tương tự như để tìm USCLN của 2 số nguyên, chúng ta có thể sử dụng thuật toán Euclid để tìm USCLN của hai đa thức

```

/* Thuật toán Euclid tính gcd(a(x),b(x)) */
EUCLID (a(x),b(x))
  A(x) = a(x); B(x) = b(x);
  while B(x) <> 0 do
    R(x) = A(x) mod B(x);
    A(x) = B(x);
    B(x) = R(x);
  end while
  return A(x);

```

### 9.3.3 Phép modulo đa thức

Tương tự như phép chia modulo trong tập số nguyên, ta định nghĩa một phép chia modulo đa thức trong tập các hàm đa thức.

Giả sử ta có hai đa thức  $f(x)$  và  $m(x)$  định nghĩa trên trường  $Z_p$ . Ta định nghĩa phép chia modulo của  $f(x)$  cho  $m(x)$  như sau:

$$r(x) = f(x) \bmod m(x) \text{ là phần dư của phép chia } f(x) / m(x)$$

Ví dụ: thuật toán AES sử dụng đa thức  $m(x) = x^8 + x^4 + x^3 + x + 1$  được định nghĩa trên trường  $Z_2$ . Cũng trên  $Z_2$  xét đa thức:

$$f(x) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

$$\text{Ta có: } f(x) \bmod m(x) = x^7 + x^6 + 1$$

### 9.3.4 Trường hữu hạn $GF(2^n)$

Tương tự như việc xây dựng tập  $Z_p$  dùng phép modulo  $p$  với  $p$  là số nguyên tố, trong phần này ta sẽ xây dựng một tập  $W_{pm}$  các đa thức dùng phép modulo đa thức.

Chọn một đa thức  $m(x)$  là đa thức tối giản trên  $Z_p$  có bậc là  $n$ . Tập  $W_{pm}$  bao gồm các đa thức trên  $Z_p$  có bậc nhỏ hơn  $n$ . Như vậy các đa thức thuộc  $W_{pm}$  có dạng:

$$f(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{với } a_i \in Z_p = \{0, 1, \dots, p-1\}$$

Tập  $W_{pm}$  có  $p^n$  phần tử.

Ví dụ:

- $p=3, n=2$  tập  $W_{pm}$  có 9 phần tử:  $\{0, 1, 2, x, x+1, x+2, 2x, 2x+1, 2x+2\}$
- $p=2, n=3$  tập  $W_{pm}$  có 8 phần tử:  $\{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$ .

Ta định nghĩa lại phép cộng và phép nhân đa thức như sau:

- Phép cộng, tương tự như phép cộng trên  $W_p$
- Phép nhân, cũng tương tự như phép nhân trên  $W_p$ , và kết quả cuối cùng được modulo với  $m(x)$  để bậc của kết quả nhỏ hơn  $n$ .

Vì  $m(x)$  là đa thức tối giản nên tương tự như số học modulo, các phần tử trong  $W_{pm}$  tồn tại phần tử nghịch đảo của phép nhân:

$$\forall f(x) \in W_{pm}, \exists f^{-1}(x) \in W_{pm}: f(x)f^{-1}(x) = 1$$

Do tồn tại phần tử nghịch đảo, nên ta có thể thực hiện được phép chia trong tập  $W_{pm}$  như sau:  $f(x) / g(x) = f(x)g^{-1}(x)$

Lúc này  $W_{pm}$  thỏa mãn các tính chất của một trường hữu hạn và ta ký hiệu trường hữu hạn này là  $GF(p^n)$  (cũng theo tên của nhà bác học Galois). Trong mã hóa, chúng ta chỉ quan tâm đến  $p=2$  tức trường đa thức hữu hạn  $GF(2^n)$  trên tập  $Z_2$ .

Ví dụ xét  $GF(2^3)$ , chọn đa thức tối giản  $m(x) = x^3 + x + 1$ , bảng bên dưới thể hiện phép cộng và phép nhân.

+	0	1	x	x+1	x <sup>2</sup>	x <sup>2</sup> +1	x <sup>2</sup> +x	x <sup>2</sup> +x+1
0	0	1	x	x+1	x <sup>2</sup>	x <sup>2</sup> +1	x <sup>2</sup> +x	x <sup>2</sup> +x+1
1	1	0	x+1	x	x <sup>2</sup> +1	x <sup>2</sup>	x <sup>2</sup> +x+1	x <sup>2</sup> +x
x	x	x+1	0	1	x <sup>2</sup> +x	x <sup>2</sup> +x+1	x <sup>2</sup>	x <sup>2</sup> +1
x+1	x+1	x	1	0	x <sup>2</sup> +x+1	x <sup>2</sup> +x	x <sup>2</sup> +1	x <sup>2</sup>
x <sup>2</sup>	x <sup>2</sup>	x <sup>2</sup> +1	x <sup>2</sup> +x	x <sup>2</sup> +x+1	0	1	x	x+1
x <sup>2</sup> +1	x <sup>2</sup> +1	x <sup>2</sup>	x <sup>2</sup> +x+1	x <sup>2</sup> +x	1	0	x+1	x
x <sup>2</sup> +x	x <sup>2</sup> +x	x <sup>2</sup> +x+1	x <sup>2</sup>	x <sup>2</sup> +1	x	x+1	0	1
x <sup>2</sup> +x+1	x <sup>2</sup> +x+1	x <sup>2</sup> +x	x <sup>2</sup> +1	x <sup>2</sup>	x+1	x	1	0

a) Bảng phép cộng

x	0	1	x	x+1	x <sup>2</sup>	x <sup>2</sup> +1	x <sup>2</sup> +x	x <sup>2</sup> +x+1
0	0	0	0	0	0	0	0	0
1	0	1	x	x+1	x <sup>2</sup>	x <sup>2</sup> +1	x <sup>2</sup> +x	x <sup>2</sup> +x+1
x	0	x	x <sup>2</sup>	x <sup>2</sup> +x	x+1	1	x <sup>2</sup> +x+1	x <sup>2</sup> +1
x+1	0	x+1	x <sup>2</sup> +x	x <sup>2</sup> +1	x <sup>2</sup> +x+1	x <sup>2</sup>	1	x
x <sup>2</sup>	0	x <sup>2</sup>	x+1	x <sup>2</sup> +x+1	x <sup>2</sup> +x	x	x <sup>2</sup> +1	1
x <sup>2</sup> +1	0	x <sup>2</sup> +1	1	x <sup>2</sup>	x	x <sup>2</sup> +x+1	x+1	x <sup>2</sup> +x
x <sup>2</sup> +x	0	x <sup>2</sup> +x	x <sup>2</sup> +x+1	1	x <sup>2</sup> +1	x+1	x	x <sup>2</sup>
x <sup>2</sup> +x+1	0	x <sup>2</sup> +x+1	x <sup>2</sup> +1	x	1	x <sup>2</sup> +x	x <sup>2</sup>	x+1

b) Bảng phép nhân

Để tìm phần tử nghịch đảo của phép nhân đa thức, ta cũng sử dụng thuật toán Euclid mở rộng tương tự như tìm nghịch đảo trong tập  $Z_p$ .

```
/* Thuật toán Euclid mở rộng trả về hai giá trị: */
/* - gcd(m(x),b(x)); */
/* - nếu gcd(m(x),b(x))=1; trả về b-1(x) mod m(x) */
```

---

```
EXTENDED_EUCLID(m(x),b(x))
```

```
A1(x) = 1; A2(x) = 0; A3(x) = m(x);
```

```
B1(x) = 0; B2(x) = 1; B3(x) = b(x);
```

```
while (B3(x)<>0)AND(B3(x)<>1) do
```

```
    Q(x) = phần thương của A3(x) / B3(x);
```

```

T1(x) = A1(x) - Q(x)B1(x);
T2(x) = A2(x) - Q(x)B2(x);
T3(x) = A3(x) - Q(x)B3(x);
A1(x) = B1(x); A2(x) = B2(x); A3(x) = B3(x);
B1(x) = T1(x); B2(x) = T2(x); B3(x) = T3(x);
end while
If B3(x)=0 then return A3(x); no inverse;
If B3(x)=1 then return 1; B2(x);

```

---

### 9.3.5 Ứng dụng GF(2<sup>n</sup>) trong mã hóa

Khi thực hiện mã hóa, đối xứng hay công khai, bản rõ và bản mã là các con số, việc mã hóa và giải mã có thể quy về việc thực hiện các phép cộng, trừ, nhân, chia. Do đó bản rõ và bản mã phải thuộc một trường nào đó để việc tính toán không ra khỏi trường. Việc quy bản rõ và bản mã về trường số thực không phải là phương án hiệu quả vì tính toán trên số thực tốn kém nhiều thời gian. Máy tính chỉ hiệu quả khi tính toán trên các số nguyên dưới dạng byte hay bit. Do đó trường  $Z_p$  là một phương án được tính đến. Tuy nhiên trường  $Z_p$  đòi hỏi  $p$  phải là một số nguyên tố, trong khi đó nếu biểu diễn bản rõ bản mã theo bit thì số lượng phần tử có dạng  $2^n$  lại không phải là số nguyên tố. Ví dụ, xét tập các phần tử được biểu diễn bởi các số nguyên 8 bit, như vậy có 256 phần tử. Tuy nhiên  $Z_{256}$  lại không phải là một trường. Nếu ta chọn trường  $Z_{251}$  thì chỉ sử dụng được các số từ 0 đến 250, các số từ 251 đến 255 không tính toán được.

Trong bối cảnh đó, việc sử dụng trường GF(2<sup>n</sup>) là một phương án phù hợp vì trường GF(2<sup>n</sup>) cũng có 2<sup>n</sup> phần tử. Ta có thể ánh xạ giữa một hàm đa thức trong GF(2<sup>n</sup>) thành một số nhị phân tương ứng bằng cách lấy các hệ số của đa thức tạo thành dãy bit  $a_{n-1}a_{n-2}...a_1a_0$ . Ví dụ xét trường GF(2<sup>3</sup>) với đa thức tối giản  $m(x) = x^3 + x + 1$  tương ứng với số nguyên 3 bit như sau:

<i>Đa thức trong GF(2<sup>3</sup>)</i>	<i>Số nguyên tương ứng</i>	<i>thập lục phân</i>
0	000	0
1	001	1
x	010	2
x+1	011	3
x <sup>2</sup>	100	4
x <sup>2</sup> +1	101	5
x <sup>2</sup> +x	110	6
x <sup>2</sup> +x+1	111	7

Bảng phép cộng và bảng phép nhân tương ứng là

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Bảng nghịch đảo của phép cộng và phép nhân:

$a$		$-a$		$a^{-1}$	
dạng đa thức	dạng số	dạng đa thức	dạng số	dạng đa thức	dạng số
0	0	0	0	-	-
1	1	1	1	1	1
x	2	x	2	$x^2+1$	5
$x+1$	3	$x+1$	3	$x^2+x$	6
$x^2$	4	$x^2$	4	$x^2+x+1$	7
$x^2+1$	5	$x^2+1$	5	x	2
$x^2+x$	6	$x^2+x$	6	$x+1$	3
$x^2+x+1$	7	$x^2+x+1$	7	$x^2$	4

Ngoài ra nếu xét bảng phép nhân của  $Z_8$

x	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Thì phân bố tần suất của các số không đều. Ta có bảng so sánh sau:

Số nguyên	1	2	3	4	5	6	7
Xuất hiện trong $Z_8$	4	8	4	12	4	8	4
Xuất hiện trong $GF(2^3)$	7	7	7	7	7	7	7

Vì vậy nếu dùng  $GF(2^3)$  thì sẽ thuận lợi hơn cho mã hóa, tránh việc sử dụng tần suất để phá mã.

### 9.3.6 Tính toán trong $GF(2^n)$

Với việc biểu diễn một hàm đa thức trong  $GF(2^n)$  thành một số nguyên  $n$  bit. Ta có thể thực hiện phép cộng và phép nhân đa thức như sau:

1) Phép cộng:

cho hai đa thức  $f(x) = \sum_{i=0}^{n-1} a_i x^i$   $g(x) = \sum_{i=0}^{n-1} b_i x^i$

Phép cộng  $f(x) + g(x)$  chính là phép XOR hai dãy bit  $a_{n-1}a_{n-2}...a_1a_0$  và  $b_{n-1}b_{n-2}...b_1b_0$ .

2) Phép nhân: liên quan đến phép modulo đa thức tối giản  $m(x)$

Chúng ta sẽ xem xét phép nhân hai đa thức trong trường  $GF(2^8)$  dùng đa thức nguyên tố  $m(x) = x^8 + x^4 + x^3 + x + 1$  (dùng trong mã hóa AES). Từ đó có thể tổng quát hóa cho trường  $GF(2^n)$  bất kỳ.

Trước hết nhận xét rằng:

$$x^8 \bmod m(x) = [m(x) - x^8] = x^4 + x^3 + x + 1$$

$$(\text{vì } m(x) \cdot 1 + x^4 + x^3 + x + 1 = x^8)$$

Một đa thức trong  $GF(2^8)$  có dạng:

$$f(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

$$\text{Suy ra: } xf(x) = (b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) \bmod m(x)$$

Nếu  $b_7 = 0$  thì:

$$xf(x) = b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x$$

Nếu  $b_7 = 1$  thì:

$$xf(x) = (b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x) + (x^4 + x^3 + x + 1)$$

Nếu viết theo dãy bit thì điều đó nghĩa là:

$$b_7b_6b_5b_4b_3b_2b_1b_0 \times 00000010 = \begin{cases} b_6b_5b_4b_3b_2b_1b_00 & \text{nếu } b_7 = 0 \\ b_6b_5b_4b_3b_2b_1b_00 \oplus 00011011 & \text{nếu } b_7 = 1 \end{cases}$$

Áp dụng lặp lại như vậy để tính  $x^k f(x)$  với  $k$  bất kỳ, và từ đó tính được  $g(x) \times f(x)$

Ví dụ: tính  $(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1)$

Phép nhân trên viết theo bit là  $01010111 \times 10000011$

- $01010111 \oplus 00000010 = 10101110$
- $01010111 \oplus 00000100 = 01011100 \oplus 00011011 = 01000111$
- $01010111 \oplus 00001000 = 10001110$
- $01010111 \oplus 00010000 = 00011100 \oplus 00011011 = 00000111$
- $01010111 \oplus 00100000 = 00001110$
- $01010111 \oplus 01000000 = 00011100$
- $01010111 \oplus 10000000 = 00111000$

$$\text{Vì vậy } 01010111 \times 10000011 = 01010111 \times [00000001 \oplus 00000010 \oplus 10000000] = 01010111 \oplus 10101110 \oplus 00111000 = 11000001$$

$$\text{Vậy } (x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^7 + x^6 + 1$$

### 9.3.7 Tính toán trong $GF(2^n)$ với phần tử sinh

Ngoài cách thức tính phép cộng và phép nhân đa thức dùng phép XOR và Shift dãy bit, ta có thể tính bằng cách dùng phần tử sinh.



Khái niệm phần tử sinh: giá trị  $g$  được gọi là phần tử sinh của trường  $GF(2^n)$  với đa thức tối giản  $m(x)$  nếu các lũy thừa  $g^0, g^1, g^2, g^3, \dots, g^{2^n-2}$  (gồm  $2^{n-1}$  phần tử) sinh ra các đa thức khác không của trường. Phép lũy thừa trên thực hiện modulo cho đa thức  $m(x)$ .

Điều kiện để  $g$  là phần tử sinh là:  $g^n = m(g) - g^n$

Ví dụ, xét lại trường  $GF(2^3)$  với  $m(x) = x^3 + x + 1$

Như vậy để  $g$  là phần tử sinh thì  $g^3 = m(g) - g^3 = g + 1$ . Ta thực hiện việc sinh các phần tử của trường như sau:

$$g^4 = g \cdot g^3 = g(g + 1) = g^2 + g$$

$$g^5 = g \cdot g^4 = g(g^2 + g) = g^3 + g^2 = g^2 + g + 1$$

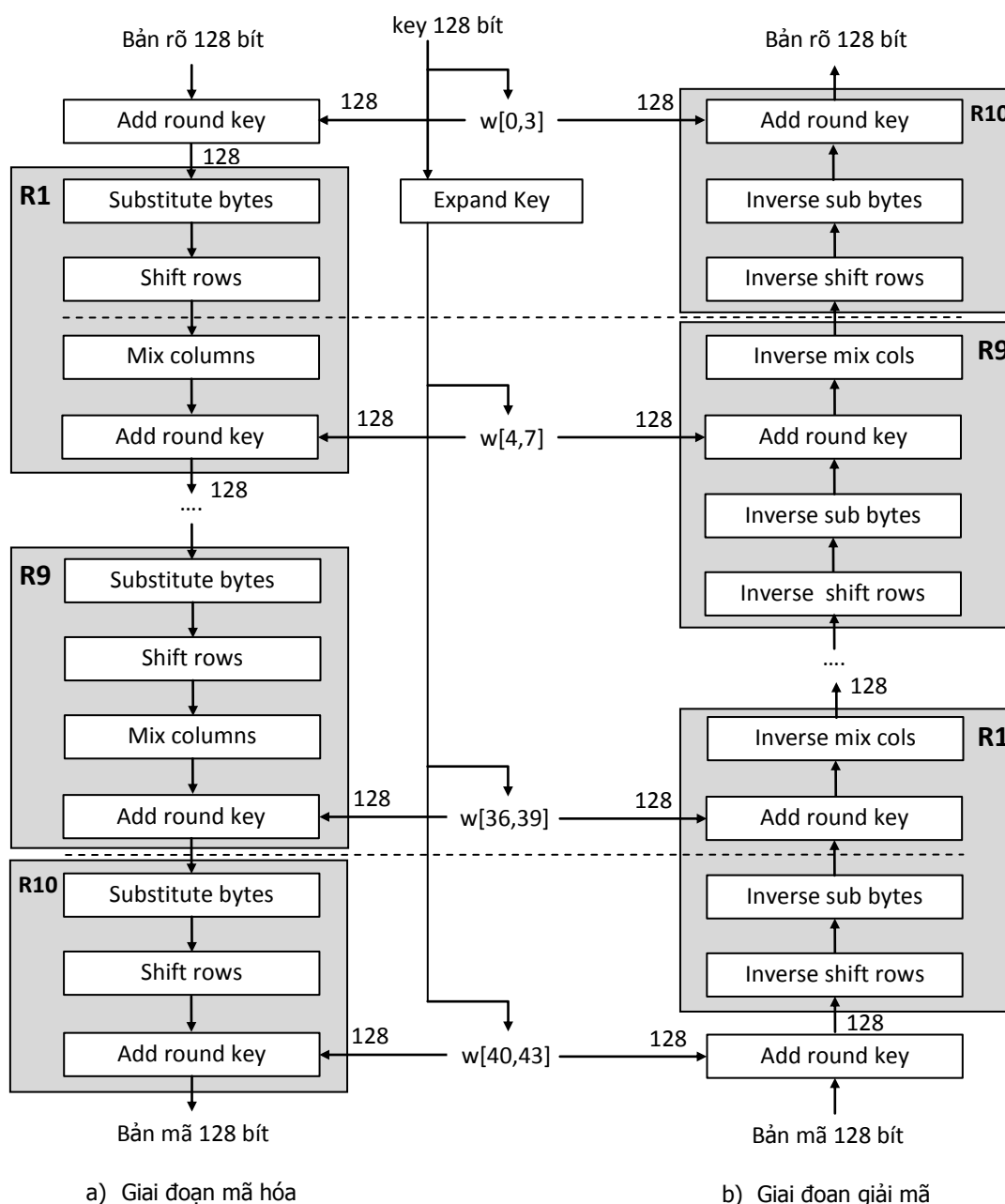
$$g^6 = g \cdot g^5 = g(g^2 + g + 1) = g^3 + g^2 + g = g^2 + 1$$

<i>Biểu diễn lũy thừa</i>	<i>Đa thức trong <math>GF(2^3)</math></i>	<i>Số nguyên tương ứng</i>	<i>thập lục phân</i>
0	0	000	0
$g^0$	1	001	1
$g^1$	$g$	010	2
$g^2$	$g^2$	100	4
$g^3$	$g+1$	011	3
$g^4$	$g^2+g$	110	6
$g^5$	$g^2+g+1$	111	7
$g^6$	$g^2+1$	101	5

Dựa vào phần tử sinh ta có thể thực hiện phép nhân đa thức bằng một phép modulo  $2^{n-1}$ . Ví dụ, để tính  $(g + 1)(g^2 + g)$ . Ta chuyển thành  $g^6 g^4 = g^{10 \bmod 7} = g^3 = g + 1$  (kết quả này giống với kết quả trong bảng phép nhân trong phần 9.3.4)

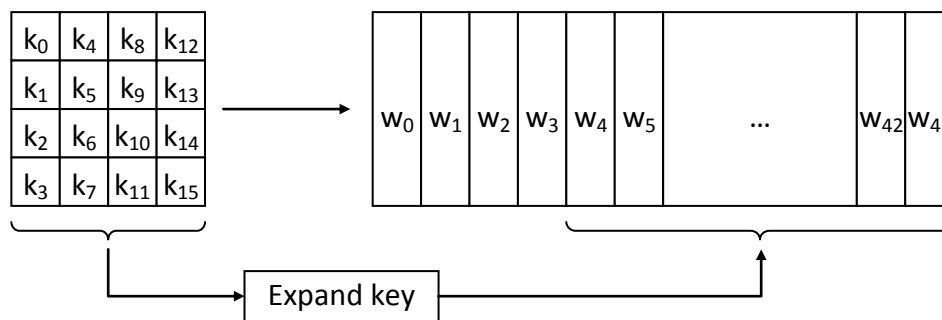
## 9.4 Mã hóa AES

Mã hóa AES là một mã hóa theo khối 128 bit không sử dụng nguyên tắc của hệ mã Feistel mà sử dụng mô hình mạng SPN. AES dùng 4 phép biến đổi chính để mã hóa một khối: Add row key, Substitute bytes, Shift rows, Mix columns. Mỗi phép biến đổi nhận tham số đầu vào có kích thước 128 bit và cho ra kết quả cũng có kích thước 128 bit. AES thực hiện 4 phép biến đổi trên nhiều lần tạo thành 10 vòng biến đổi như hình bên dưới.



Các phép biến đổi Substitute bytes, Shift rows, Mix columns có phép biến đổi ngược tương ứng là Inverse sub bytes, Inverse shift rows, Inverse mix cols. Riêng phép biến đổi Add row key đơn giản chỉ là phép XOR nên phép biến đổi ngược cũng là Add row key. Vận dụng các phép biến đổi ngược trên, thuật toán giải mã AES cũng gồm 10 vòng thực hiện theo chiều ngược lại.

Kích thước khóa ban đầu là 128 bit (gồm 16 byte). AES dùng hàm Expand key để mở rộng kích thước khóa thành 44 word 32 bit. 44 word này được chia thành 11 cụm khóa con, mỗi khóa con 4 word làm tham số cho 11 thao tác Add row key.



Mỗi khối bản rõ gồm 16 byte  $p_0 p_1 \dots p_{15}$  được tổ chức dưới dạng một ma trận 4x4 (ma trận state). Chúng ta đổi ký hiệu cho ma trận này dưới dạng  $s_{00} s_{10} s_{20} s_{30} s_{01} s_{11} \dots s_{23} s_{33}$ .

$p_0$	$p_4$	$p_8$	$p_{12}$
$p_1$	$p_5$	$p_9$	$p_{13}$
$p_2$	$p_6$	$p_{10}$	$p_{14}$
$p_3$	$p_7$	$p_{11}$	$p_{15}$

$s_{00}$	$s_{01}$	$s_{02}$	$s_{03}$
$s_{10}$	$s_{11}$	$s_{12}$	$s_{13}$
$s_{20}$	$s_{21}$	$s_{22}$	$s_{23}$
$s_{30}$	$s_{31}$	$s_{32}$	$s_{33}$

Các phép biến đổi Add row key, Substitute bytes, Shift rows, Mix columns sẽ thực hiện trên ma trận  $S$  4x4 này.

Các phép tính số học trong AES được thực hiện trong trường  $GF(2^8)$  với đa thức tối giản là  $m(x) = x^8 + x^4 + x^3 + x + 1$ . Từ đây về sau ta chỉ nói đơn giản là  $GF(2^8)$ . Phần sau trình bày chi tiết các thao tác Add row key, Substitute bytes, Shift rows, Mix columns và Expand key.

#### 9.4.1 Substitute bytes

Trong phần này, ta sử dụng một bảng tra cứu 16x16 byte (gọi là S-box). Bảng này được thiết lập như sau:

**Bước 1:** điền các con số từ 0 đến 255 vào bảng theo từng hàng. Vậy hàng 0 gồm các con số {00}, {01}, ..., {0F} (thập lục phân). Hàng 1 gồm các con số {10}, {11}, ..., {1F}. Điều này có nghĩa là tại hàng  $x$  cột  $y$  có giá trị  $\{xy\}$

**Bước 2:** thay thế mỗi byte trong bảng bằng giá trị nghịch đảo trong trường  $GF(2^8)$ . Quy ước nghịch đảo của {00} cũng là {00}

**Bước 3:** Đối với mỗi byte trong bảng, ký hiệu 8 bit là  $b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$ . Thay thế mỗi bit  $b_i$  bằng giá trị  $b'_i$  được tính sau:

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

Với  $c_i$  là bit thứ  $i$  của số {63}, tức  $c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0 = 01100011$ . Việc tính toán trên tương đương với phép nhân ma trận sau trên  $GF(2^8)$  ( $B' = XB \oplus C$ ):

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Trong đó phép cộng thực hiện như phép XOR. Hình dưới trình bày nội dung bảng S-box sau khi tính toán.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

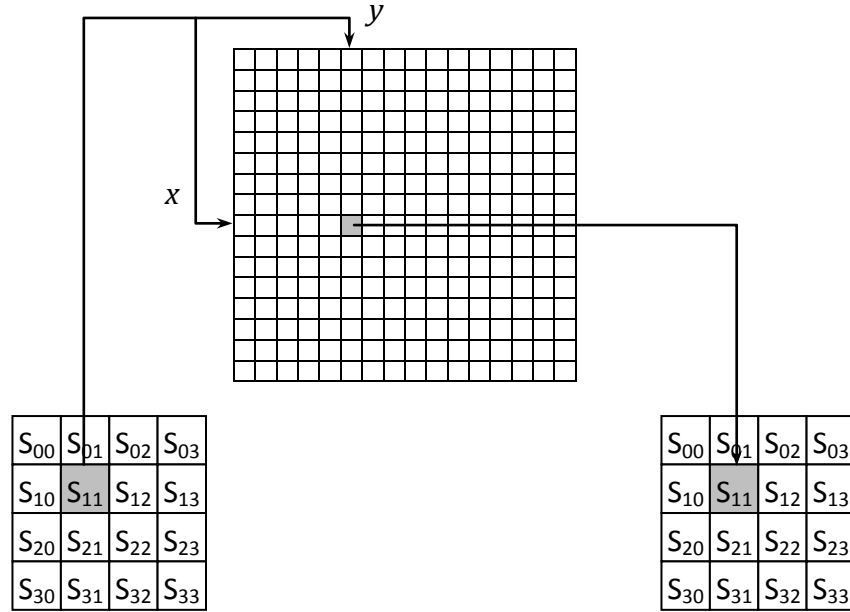
{95} → {8A} → {2A}

Ví dụ: xét giá trị {95}, tại bước 1, giá trị tại dòng 9 cột 5 là {95}, sau bước 2 tính nghịch đảo giá trị của ô này là {8A} có dạng nhị phân là 10001010. Thực hiện phép nhân ma trận:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Kết quả dưới dạng thập lục phân là {2A}

Dựa trên bảng tra cứu S-box, phép biến đổi Substitute bytes thực hiện như sau: Mỗi byte trong ma trận state S, dưới dạng thập lục phân là {xy}, được thay thế bằng giá trị trong bảng S-box tại dòng x cột y.



Sau đây là một ví dụ về phép substitute bytes

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

Phép biến đổi ngược Inverse sub bytes:

Trước tiên, ta cũng phải xây dựng một bảng Inverse S-box (IS-box). Nghĩa là nếu với đầu vào {95}, S-box cho ra kết quả {2A}, thì với đầu vào là {2A}, IS-box sẽ cho ra lại kết quả {95}. Việc xây dựng IS-box cũng giống như xây dựng S-box tại bước 1 và bước 2. Tại bước 3, IS-box thực hiện phép thay thế sau:

$$b_i = b'_{(i+2) \bmod 8} \oplus b'_{(i+5) \bmod 8} \oplus b'_{(i+7) \bmod 8} \oplus d_i$$

Với  $d_i$  là bit thứ  $i$  của số {05}, tức  $d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0 = 00000101$ . Và phép nhân ma trận tương đương là ( $B = YB' \oplus D$ ):

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Hình dưới trình bày nội dung bảng IS-box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

{2A} → {95}

Như vậy phép biến đổi Inverse sub bytes thực hiện như sau: Mỗi byte trong ma trận state S, dưới dạng thập lục phân là {xy}, được thay thế bằng giá trị trong bảng IS-box tại dòng x cột y.

Để chứng minh Inverse sub bytes là phép biến đổi ngược của Substitute bytes, ta cần chứng minh  $Y(XB \oplus C) \oplus D = B$ , nghĩa là  $YXB \oplus YC \oplus D = B$ . Ta có

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} +$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

(YXB = IB và YC = D)

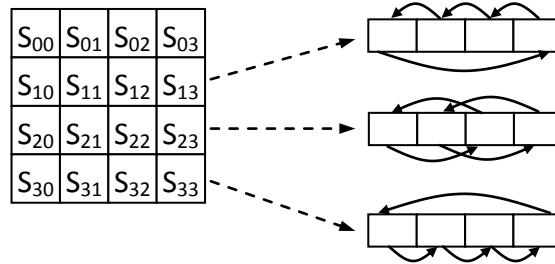
**Mục đích của Substitute bytes:**

Bảng S-box dùng để chống lại hình thức tấn công known-plaintext. Giữa input và output của phép Substitute bytes không thể mô tả bằng một công thức toán đơn giản. (xem phần mã khối an toàn lý tưởng trong chương 3)

#### 9.4.2 Shift rows

Thao tác Shift rows thực hiện hoán vị các byte trong ma trận state theo cách thức sau:

- Dòng thứ nhất giữ nguyên
- Dòng thứ 2 dịch vòng trái 1 byte
- Dòng thứ 3 dịch vòng trái 2 byte
- Dòng thứ 4 dịch vòng trái 3 byte



Thao tác Inverse shift rows thực hiện ngược lại, các dòng thứ 2, thứ 3, thứ 4 được dịch vòng phải tương ứng 1 byte, 2 byte và 3 byte.

#### Mục đích của Shift rows:

Xáo trộn các byte để tạo các cột khác nhau trước khi sử dụng cột cho thao tác Mix columns.

#### 9.4.3 Mix columns

Thao tác Mix column biến đổi độc lập từng cột trong ma trận state bằng một phép nhân đa thức.

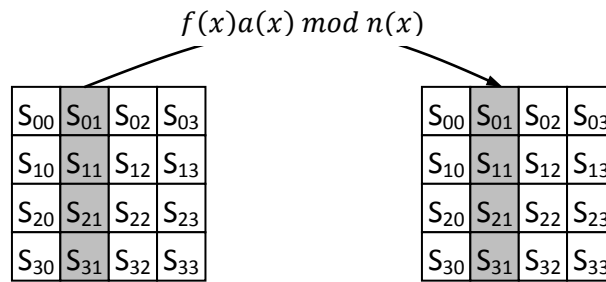
Một cột trong ma trận state có thể xem là một đa thức bậc 3, ví dụ cột 1 của ma trận viết dưới dạng đa thức là:

$$f(x) = s_{01}x^3 + s_{11}x^2 + s_{21}x + s_{31}$$

Đa thức trên được nhân với đa thức:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Trong đó phép cộng và nhân các hệ số được thực hiện trong trường  $GF(2^8)$ . Đa thức kết quả có bậc lớn hơn 3 (ví dụ hệ số của bậc 6 là  $\{03\}s_{01}$ ), tuy nhiên ta chỉ cần sử dụng 4 hệ số cho giá trị cột mới nên đa thức kết quả sẽ được modulo thêm cho đa thức  $n(x) = x^4 + 1$ . Bốn byte hệ số của  $x^3, x^2, x, x^0$  được thay thế cho bốn byte ban đầu trong cột.



Phép nhân đa thức trên có thể biểu diễn dưới dạng phép nhân ma trận như sau:

$$\begin{bmatrix} s'_{01} \\ s'_{11} \\ s'_{21} \\ s'_{31} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{01} \\ s_{11} \\ s_{21} \\ s_{31} \end{bmatrix}$$

Hình sau là một ví dụ về phép Mix columns

87	F2	4D	97	→	47	40	A3	4C
6E	4C	90	EC		37	D4	70	9F
46	E7	4A	C3		94	E4	3A	42
A6	8C	D8	95		ED	A5	A6	BC

Trong phép biến đổi ngược Inverse mix cols, mỗi cột của ma trận state được nhân với đa thức  $b(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$  và modulo cho đa thức  $x^4 + 1$ . Hay viết dưới dạng ma trận

$$\begin{bmatrix} s'_{01} \\ s'_{11} \\ s'_{21} \\ s'_{31} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{01} \\ s_{11} \\ s_{21} \\ s_{31} \end{bmatrix}$$

Có thể dễ dàng kiểm tra các tính chất sau trong  $GF(2^8)$

$$a(x)b(x) \bmod (x^4 + 1) = 1$$

và

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

và từ đó chứng minh được Inverse mix cols là phép biến đổi ngược của Mix columns

#### Mục đích của Mix columns:

Việc nhân mỗi cột với đa thức  $a(x)$  và modulo  $n(x)$  là cho mỗi byte trong cột kết quả đều phụ thuộc vào bốn byte trong cột ban đầu. Thao tác Mix columns kết hợp với Shift



rows đảm bảo rằng sau một vài vòng biến đổi, 128 bit trong kết quả đều phụ thuộc vào tất cả 128 bit ban đầu. Điều này tạo ra tính khuếch tán (diffusion) cần thiết cho mã hóa.

#### 9.4.4 Add row key

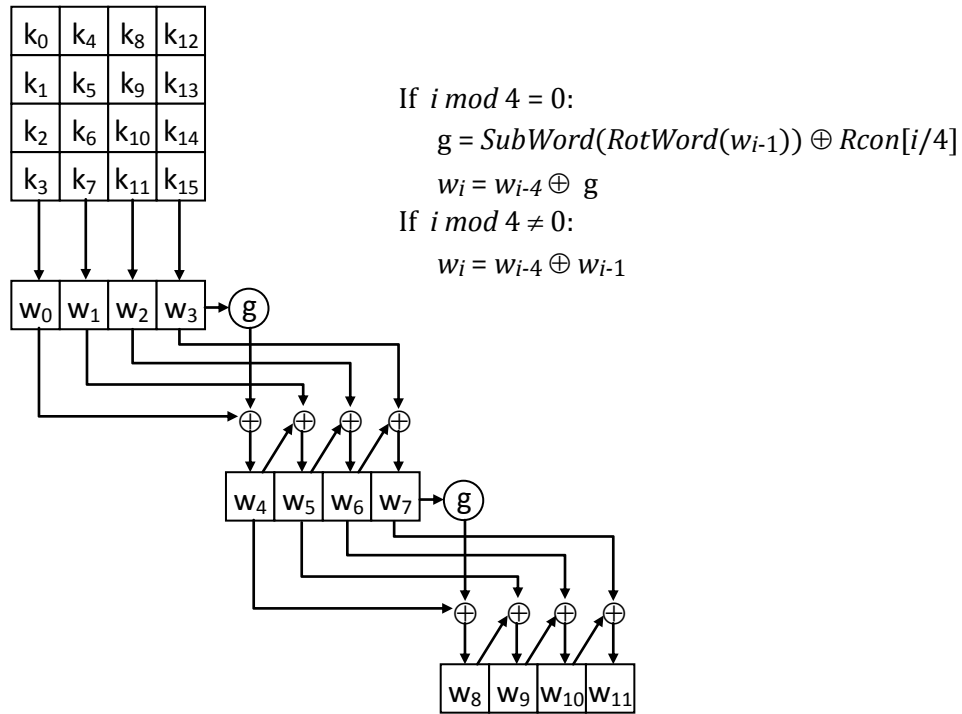
Trong thao tác Add row key, 128 bit của ma trận state sẽ được XOR với 128 bit của khóa con của từng vòng. Vì sử dụng phép XOR nên phép biến đổi ngược của Add row key cũng chính là Add row key.

Việc kết hợp với khóa bí mật tạo ra tính làm rối (confusion) của mã hóa. Sự phức tạp của thao tác Expand key giúp gia tăng tính làm rối này.

#### 9.4.5 Expand key

Thao tác Expand key có input là 16 byte (4 word) của khóa bí mật, và sinh ra một mảng 44 word (176 byte). 44 word này được sử dụng cho 11 vòng mã hóa của AES, mỗi vòng dùng 4 word.

Từ bốn word đầu vào  $w_0w_1w_2w_3$ , trong lần lặp đầu tiên thao tác Expand key sinh ra bốn word  $w_4w_5w_6w_7$ , lần lặp thứ 2 từ  $w_4w_5w_6w_7$  sinh ra  $w_8w_9w_{10}w_{11}$ , cứ như thế cho đến lần lặp thứ 10 sinh ra bốn word cuối cùng  $w_{40}w_{41}w_{42}w_{43}$  như hình vẽ bên dưới



Trong mỗi lần lặp để sinh ra 4 word, word đầu tiên sinh ra theo quy tắc  $w_i = w_{i-4} \oplus g$  với  $g = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{Rcon}[i/4]$ . Ba word tiếp theo sinh ra theo quy tắc  $w_i = w_{i-4} \oplus w_{i-1}$ . Sau đây chúng ta sẽ tìm hiểu các hàm SubWord, RotWord và mảng Rcon.

**RotWord:** dịch vòng trái một byte. Giả sử word đầu vào có 4 byte là  $[b_0, b_1, b_2, b_3]$  thì kết quả của RotWord là  $[b_1, b_2, b_3, b_0]$ .

**SubWord:** thay thế mỗi byte trong word đầu vào bằng cách tra cứu bảng S-box trong thao tác Substitute Bytes.

*Rcon*: là một mảng hằng số. Mảng này gồm 10 word ứng với 10 vòng AES. Bốn byte của một phần tử  $Rcon[j]$  là  $(RC[j], 0, 0, 0)$  với  $RC[j]$  là mảng 10 byte như sau:

j	1	2	3	4	5	6	7	8	9	10
$RC[j]$	1	2	4	8	10	20	40	80	1B	36

$(RC[j] = RC[j-1]*2$  với phép nhân thực hiện trong  $GF(2^8)$ )

**Mục đích của Expand key:** dùng để chống lại known-plaintext attack

- Biết một số bit của khóa hay khóa con cũng không thể tính các bit còn lại.
- Không thể tính ngược: biết một khóa con cũng không thể tính lại các khóa con trước đó.
- Tính khuếch tán: một bit của khóa chính tác động lên tất cả các bit của các khóa con.

#### 9.4.6 Kết luận

Phương pháp mã hóa AES đơn giản, có thể thực hiện hiệu quả trên các vi xử lý 8 bit (dùng trong smartcard) cũng như trên các vi xử lý 32 bit, chỉ dùng phép XOR và phép Shift bit. Đây chính là yếu tố cơ bản để phương pháp này được chọn làm chuẩn mã hóa của Hoa Kỳ.

Mã hóa AES còn có 1 số biến thể khác cho phép chiều dài của khóa có thể là 192 bit và 256 bit. Nếu khóa là 192 bit thì kích thước khóa mở rộng là 52 word 4 byte và do đó AES thực hiện 12 vòng, nếu khóa là 256 bit thì kích thước khóa mở rộng là 60 word và AES thực hiện 14 vòng.

## CHƯƠNG 10. MÃ HÓA ĐƯỜNG CONG ELLIPTIC

Trong chương 4 về mã hóa khóa công khai, chúng ta đã tìm hiểu phương pháp mã hóa RSA và phương pháp trao đổi khóa Diffie-Hellman. RSA dùng hàm một chiều là phép phân tích một số lớn thành tích hai thừa số nguyên tố. Diffie-Hellman dùng hàm một chiều là hàm logarit rời rạc. Trong chương này chúng ta tiếp tục tìm hiểu một loại hàm một chiều khác dựa trên số học Elliptic. Từ đó chúng ta sẽ xây dựng một phương pháp mã hóa đường cong Elliptic (ECC) và phương pháp trao đổi khóa phiên ECDiffie-Hellman. Đối với phương pháp RSA, để bảo đảm an toàn, chúng ta phải chọn số  $N$  lớn (1024 bit), điều này khiến cho RSA thực hiện chậm. Mã hóa ECC giải quyết vấn đề này khi dùng các tham số có kích thước ngắn hơn (168 bit) tuy nhiên vẫn đảm bảo độ an toàn như RSA 1024 bit.

### 10.1 Đường cong Elliptic trên số thực

Đường cong Elliptic là đường cong có dạng:

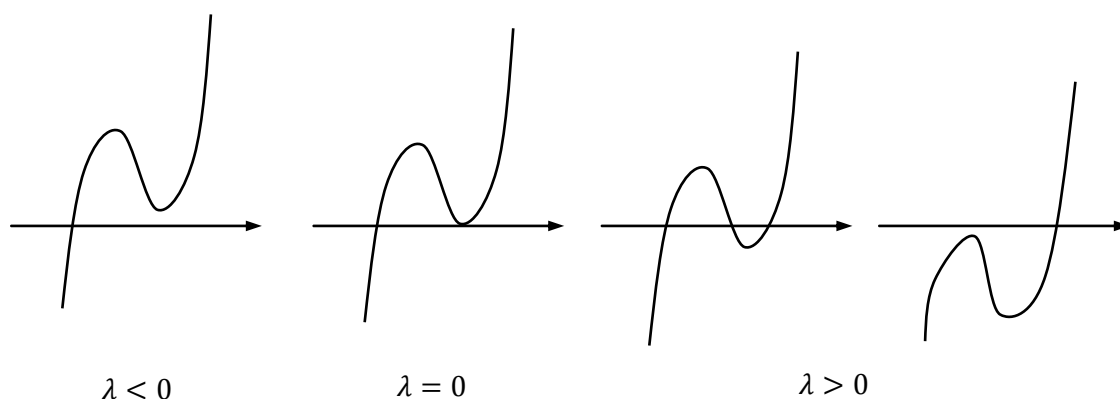
$$y^2 = x^3 + ax + b \quad a, b \in \mathbb{R}$$

Trước khi khảo sát đồ thị của đường cong Elliptic, chúng ta xem lại đường bậc 3 sau:

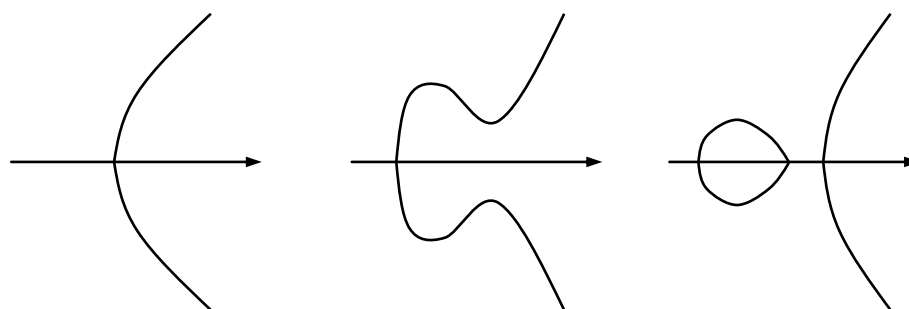
$$y = f(x) = x^3 + ax + b$$

Nếu  $a > 0$ ,  $f(x)$  đơn điệu tăng.

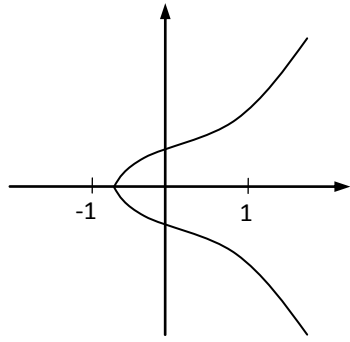
Nếu  $a \leq 0$ ,  $f(x)$  có 4 trường hợp sau: đặt  $\lambda = 4a^3 + 27b^2$



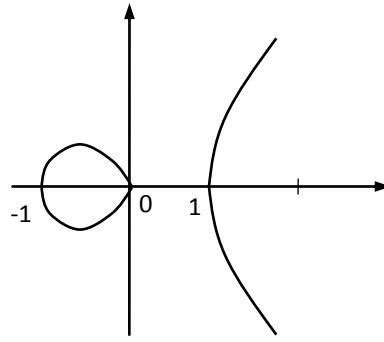
Từ đó chúng ta có các trường hợp sau đây của đường cong Elliptic (không sử dụng trường hợp  $\lambda=0$  vì lúc này đường cong bị gãy):



Hình dưới minh họa hai đường cong Elliptic  $y^2 = x^3 - x$  và  $y^2 = x^3 + x + 1$



$$y^2 = x^3 + x + 1$$

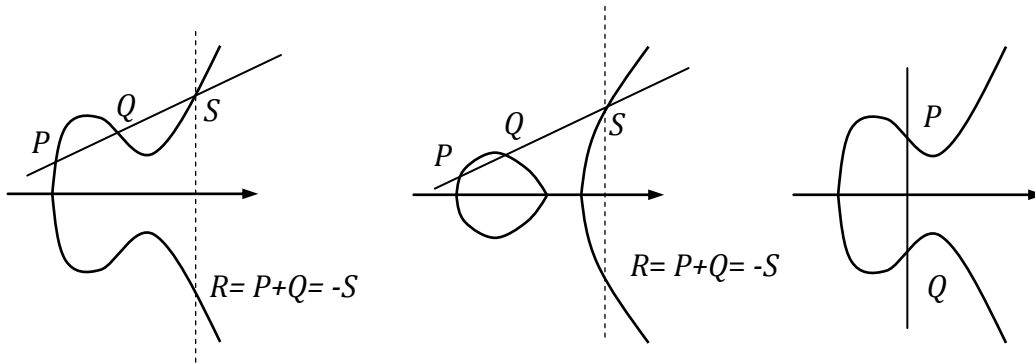


$$y^2 = x^3 - x$$

Trong đường cong Elliptic, chúng ta định nghĩa thêm một điểm  $O$  (điểm vô cực).

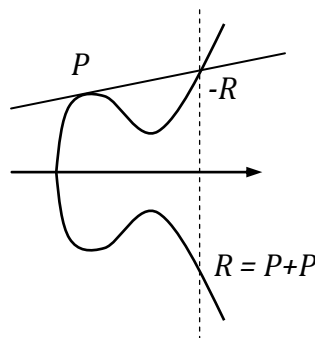
Gọi  $E(a, b)$  là tập các điểm thuộc đường cong  $y = x^3 + ax + b$  cùng với điểm  $O$ . ta định nghĩa phép cộng trên tập các điểm thuộc  $E(a, b)$  như sau:

- 1) Điểm  $O$  là phần tử đơn vị của phép cộng. Như vậy với  $P \in E(a, b), P \neq O$  thì  $P + O = O + P = P$ . Trong phần tiếp theo, ta giả định  $P \neq O$  và  $Q \neq O$ .
- 2) Phần tử nghịch đảo của điểm  $P$  trong phép cộng, ký hiệu  $-P$ , là điểm đối xứng với  $P$  qua trục hoành, như vậy  $P + (-P) = O$
- 3) Với 2 điểm  $P, Q$  bất kỳ, kẻ một đường thẳng đi qua  $P$  và  $Q$  thì sẽ cắt đường cong Elliptic tại một điểm thứ 3 là điểm  $S$ . Phép cộng  $P$  và  $Q$  sẽ là  $R = P + Q = -S$



Trong trường hợp  $P$  và  $Q$  đối xứng qua trục hoành, hay nói cách khác  $Q = -P$  thì đường thẳng nối  $P, Q$  sẽ cắt đường cong Elliptic tại vô cực, hay  $P + (-P) = O$ . Điều này phù hợp với định nghĩa 2.

- 4) Để tính  $P + P$ , ta vẽ đường thẳng tiếp tuyến với đường cong Elliptic tại  $P$ , đường thẳng này cắt đường cong tại điểm  $S$ , lúc đó  $R = P + P = -S$



Có thể thấy, tập  $E(a, b)$  cùng với phép cộng định nghĩa như trên tạo thành một *nhóm Abel*

Tính giá trị của phép cộng:

Gọi tọa độ của điểm  $P$  là  $(x_P, y_P)$ , của điểm  $Q$  là  $(x_Q, y_Q)$ . Ta tính tọa độ điểm  $R = P + Q = -S$  như sau:

Đặt hệ số góc đường thẳng là  $\Delta$ :

$$\Delta = \frac{y_Q - y_P}{x_Q - x_P}$$

Ta tính được:  $x_R = \Delta^2 - x_P - x_Q$

$$y_R = \Delta(x_P - x_R) - y_P$$

Chứng minh:

Để ngắn gọn, ký hiệu  $P(x_1, y_1)$ ,  $Q(x_2, y_2)$ ,  $S(x_3, y_3)$ . Ta có:

$$y_1^2 = x_1^3 + ax_1 + b \quad (1)$$

$$y_2^2 = x_2^3 + ax_2 + b \quad (2)$$

$$y_3^2 = x_3^3 + ax_3 + b \quad (3)$$

$$y_3 = \Delta(x_3 - x_1) + y_1 \quad (\text{điểm } S \text{ thuộc đường thẳng nối } P \text{ và } Q) \quad (4)$$

Thay (4) vào (3):

$$(\Delta(x_3 - x_1) + y_1)^2 = x_3^3 + ax_3 + b$$

$$\Leftrightarrow \Delta^2(x_3 - x_1)^2 + 2\Delta(x_3 - x_1)y_1 + y_1^2 = x_3^3 + ax_3 + b$$

Thay  $y_1^2 = x_1^3 + ax_1 + b$  vào phương trình trên, ta có:

$$\begin{aligned} \Delta^2(x_3 - x_1)^2 + 2\Delta(x_3 - x_1)y_1 &= x_3^3 + ax_3 - x_1^3 - ax_1 \\ \Leftrightarrow \Delta^2(x_3 - x_1) + 2\Delta y_1 &= x_3^2 + x_3x_1 + x_1^2 + a \end{aligned} \quad (5)$$

Lấy (2) trừ cho (1) ta có:

$$\begin{aligned} y_2^2 - y_1^2 &= x_2^3 - x_1^3 + ax_2 - ax_1 \\ \Leftrightarrow \frac{(y_2 - y_1)^2}{x_2 - x_1} + 2 \frac{(y_2 - y_1)}{x_2 - x_1} y_1 &= x_2^2 + x_2x_1 + x_1^2 + a \\ \Leftrightarrow 2\Delta y_1 &= x_2^2 + x_1x_2 + x_1^2 + a - \Delta^2(x_2 - x_1) \end{aligned} \quad (6)$$

Thay (6) vào (5) ta có:

$$\begin{aligned} \Delta^2(x_3 - x_1) - \Delta^2(x_2 - x_1) + x_2^2 + x_2x_1 + x_1^2 &= x_3^2 + x_3x_1 + x_1^2 \\ \Leftrightarrow \Delta^2(x_3 - x_2) &= x_3^2 - x_2^2 + x_3x_1 - x_2x_1 \\ \Leftrightarrow \Delta^2 &= x_3 + x_2 + x_1 \\ \Leftrightarrow x_3 &= \Delta^2 - x_2 - x_1 \end{aligned}$$

Hay nói cách khác  $x_S = \Delta^2 - x_P - x_Q$  và  $y_S = \Delta(x_S - x_P) + y_P$ , từ đó ta có đpcm.

Tương tự, thực hiện tính tọa độ của điểm  $R = P + P = -S$ , khi  $y_P \neq 0$  ta có:

$$x_R = \left( \frac{3x_P^2 + a}{2y_P} \right)^2 - 2x_P$$

$$y_R = \left( \frac{3x_P^2 + a}{2y_P} \right) (x_P - x_R) - y_P$$

Chứng minh:

Không mất tổng quát xét một nửa đường cong elliptic:

$$y = f(x) = \sqrt{x^3 + ax + b}$$

Gọi  $\Delta$  là hệ số góc của tiếp tuyến với đường cong elliptic tại điểm P, như vậy:

$$\Delta = f'(x_1) = \frac{1}{2} (3x_1^2 + a)(x_1^3 + ax_1 + b)^{-1/2}$$

$$\Delta = \frac{(3x_1^2 + a)}{2y_1}$$

Tương tự như trong cách tính  $R = P + Q = -S$  ta cũng có phương trình (5), trong đó  $(x_3, y_3)$  là tọa độ điểm S:

$$\Delta^2(x_3 - x_1) + 2\Delta y_1 = x_3^2 + x_3 x_1 + x_1^2 + a$$

$$\Leftrightarrow \Delta^2(x_3 - x_1) + 3x_1^2 + a = x_3^2 + x_3 x_1 + x_1^2 + a$$

$$\Leftrightarrow x_3^2 + (x_1 - \Delta^2)x_3 - 2x_1^2 + \Delta^2 x_1 = 0$$

$$\Leftrightarrow (x_3 - (\Delta^2 - 2x_1))(x_3 - x_1) = 0$$

Vậy ta có:  $x_3 = \Delta^2 - 2x_1$  và từ đó suy ra đpcm.

## 10.2 Đường cong Elliptic trên trường $Z_p$ .

Đường cong Elliptic trên trường  $Z_p$  là đường cong có các hệ số thuộc trường  $Z_p$ , đường cong này có dạng:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \quad a, b, x, y \in Z_p$$

Ví dụ trong trường  $Z_{23}$ , chọn  $a = 1, b = 1, x = 9, y = 7$  ta có:

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$

$$49 \bmod 23 = 739 \bmod 23 = 3$$

Khác với đường cong Elliptic trong trường số thực, chúng ta không thể biểu diễn đường cong Elliptic  $Z_p$  bằng đồ thị hàm số liên tục. Bảng bên dưới liệt kê các điểm  $(x, y)$  của đường cong trong trường  $Z_{23}$  với  $a=1, b=1$ :

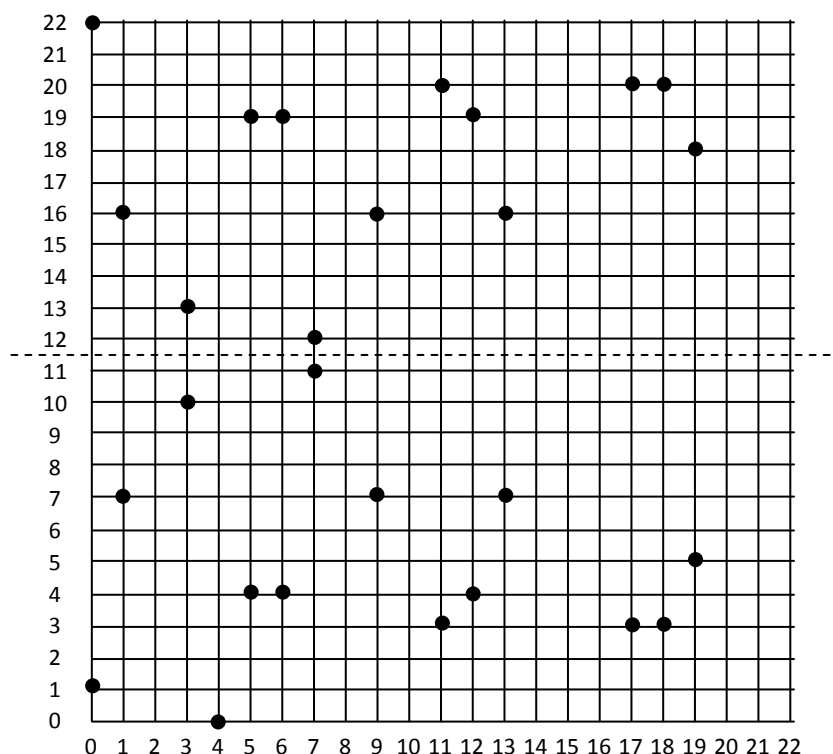
(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)

(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

Cũng tương tự như khái niệm đối xứng qua trục hoành của đường cong Elliptic số thực, đường cong Elliptic  $Z_p$  cũng đối xứng theo nghĩa đối xứng modulo. Giả sử điểm  $(x, y)$  thuộc đường cong Elliptic  $Z_p$  trên thì điểm  $(x, p - y)$  cũng thuộc đường cong trên vì:

$$(p - y)^2 = p^2 - 2py + y^2 \equiv y^2 \pmod{p}$$

Ví dụ  $(1, 7)$  đối xứng với  $(1, 16)$  vì  $7+16 = 0 \pmod{23}$ . Hình vẽ bên dưới minh họa tính đối xứng này.



Các điểm đối xứng với nhau qua đường  $y = 11.5$ . Riêng điểm  $(4, 0)$  xem như là đối xứng với chính nó.

Cũng tương tự như nhóm Abel  $E(a, b)$  định nghĩa trên đường cong Elliptic số thực, chúng ta cũng định nghĩa một nhóm Abel  $E_p(a, b)$  gồm các điểm của đường cong Elliptic  $Z_p$  cùng với điểm vô cực  $O$ .

- 1) Điểm  $O$  là phần tử đơn vị của phép cộng.  $P + O = O + P = O$ .
- 2) Phần tử nghịch đảo của điểm  $P$  trong phép cộng, ký hiệu  $-P$ , là điểm đối xứng với  $P$ , như vậy  $P + (-P) = O$
- 3) Với 2 điểm  $P, Q$  bất kỳ, phép cộng  $R = P + Q$  được xác định bằng công thức:

$$x_R = \Delta^2 - x_P - x_Q \pmod{p}$$

$$y_R = \Delta(x_P - x_R) - y_P \pmod{p}$$

Trong đó:

$$\Delta = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} \bmod p & \text{nếu } P \neq Q \\ \left( \frac{3x_P^2 + a}{2y_P} \right) \bmod p & \text{nếu } P = Q \end{cases}$$

Chú ý rằng khái niệm kẻ đường thẳng không còn áp dụng trong đường cong Elliptic  $Z_p$ . Do đó, với cách tính trên, ta cần chứng minh tính đúng, tức  $R(x_R, y_R)$  là điểm thuộc đường cong. Xét trường hợp  $P \neq Q$ :

$$y_1^2 \equiv x_1^3 + ax_1 + b \bmod p \quad (1)$$

$$y_2^2 \equiv x_2^3 + ax_2 + b \bmod p \quad (2)$$

$$x_3 = \Delta^2 - x_1 - x_2 \bmod p \quad (3)$$

$$y_3 = \Delta(x_3 - x_1) + y_1 \bmod p \quad (x_3, y_3 \text{ là tọa độ điểm } S) \quad (4)$$

Ta cần chứng minh  $(x_3, y_3)$  thuộc đường cong, nghĩa là:

$$\begin{aligned} y_3^2 &\equiv x_3^3 + ax_3 + b \bmod p \\ \Leftrightarrow \Delta^2(x_3 - x_1)^2 + 2\Delta(x_3 - x_1)y_1 + y_1^2 &\equiv x_3^3 + ax_3 + b \bmod p \\ \Leftrightarrow \Delta^2(x_3 - x_1)^2 + 2\Delta(x_3 - x_1)y_1 &\equiv x_3^3 - x_1^3 + ax_3 - ax_1 \bmod p \end{aligned} \quad (5)$$

Vì  $p$  là số nguyên tố nên tồn tại phần tử nghịch đảo của  $x_3 - x_1$  trong phép modulo  $p$ . Do đó (5) tương đương với:

$$\Delta^2(x_3 - x_1) + 2\Delta y_1 \equiv x_3^2 + x_1x_3 + x_1^2 + a \bmod p \quad (6)$$

Để chứng minh (6), lấy (2) trừ cho (1) ta cũng có:

$$\begin{aligned} y_2^2 - y_1^2 &\equiv x_2^3 - x_1^3 + ax_2 - ax_1 \bmod p \\ \Leftrightarrow \frac{(y_2 - y_1)^2}{x_2 - x_1} + 2 \frac{(y_2 - y_1)}{x_2 - x_1} y_1 &\equiv x_2^2 + x_2x_1 + x_1^2 + a \bmod p \\ \Leftrightarrow 2\Delta y_1 &\equiv x_2^2 + x_1x_2 + x_1^2 + a - \Delta^2(x_2 - x_1) \bmod p \end{aligned} \quad (7)$$

Từ (3) ta có:

$$\begin{aligned} \Delta^2 &\equiv x_3 + x_2 + x_1 \bmod p \\ \Leftrightarrow \Delta^2(x_3 - x_2) &\equiv x_3^2 - x_2^2 + x_3x_1 - x_2x_1 \bmod p \\ \Leftrightarrow \Delta^2(x_3 - x_2) + x_2^2 + x_2x_1 + x_1^2 + a &\equiv x_3^2 + x_3x_1 + x_1^2 + a \bmod p \end{aligned}$$

Thay (7) vào phương trình trên ta có:

$$\begin{aligned} \Delta^2(x_3 - x_2) + 2\Delta y_1 + \Delta^2(x_2 - x_1) &\equiv x_3^2 + x_3x_1 + x_1^2 + a \bmod p \\ \Leftrightarrow \Delta^2(x_3 - x_1) + 2\Delta y_1 &\equiv x_3^2 + x_3x_1 + x_1^2 + a \bmod p \end{aligned}$$

Vậy (6) đúng, nghĩa là điểm  $S(x_3, y_3)$  thuộc đường cong, do đó  $R(x_R, y_R)$  cũng thuộc đường cong. Chứng minh tương tự cho trường hợp  $R = P + P$ .

Ví dụ: trong  $E_{23}(1,1)$ , chọn  $P = (3,10)$ ,  $Q = (9,7)$ , vậy:

$$\Delta = \frac{7-10}{9-3} \bmod 23 = \frac{-3}{6} \bmod 23 = 20.4 \bmod 23 = 11$$

$$x_R = 11^2 - 3 - 9 \bmod 23 = 17$$



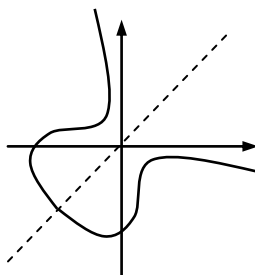
$$y_R = 11(3 - 17) - 10 \bmod 23 = 20$$

(17, 20) là điểm thuộc đường cong  $E_{23}(1,1)$

### 10.3 Đường cong Elliptic trên trường $GF(2^m)$ .

Đường cong Elliptic trên trường  $GF(2^m)$  là đường cong có các hệ số thuộc trường  $GF(2^m)$ , đường cong này có dạng hơi khác so với trên  $Z_p$ :

$$y^2 + xy = x^3 + ax + b \quad a, b, x, y \in GF(2^m)$$



**Đường cong  $y^2 + xy = x^3 + ax + b$  trên trường số thực**

Bây giờ chúng ta sẽ xét tập  $E_{2^m}(a, b)$  gồm các điểm trên đường cong Elliptic này cùng với điểm vô cực  $O$ .

Ví dụ, xét trường  $GF(2^4)$  với đa thức tối giản là  $m(x) = x^4 + x + 1$ . Phần tử sinh  $g$  của trường này có điều kiện  $g^4 = g + 1$ . Bảng các lũy thừa của  $g$  là:

<i>Biểu diễn lũy thừa</i>	<i>Đa thức trong <math>GF(2^3)</math></i>	<i>Số nhị phân</i>	<i>Biểu diễn lũy thừa</i>	<i>Đa thức trong <math>GF(2^3)</math></i>	<i>Số nhị phân</i>
0	0	0000	$g^7$	$g^3 + g + 1$	1011
$g^0$	1	0001	$g^8$	$g^2 + 1$	0101
$g^1$	$g$	0010	$g^9$	$g^3 + g$	1010
$g^2$	$g^2$	0100	$g^{10}$	$g^2 + g + 1$	0111
$g^3$	$g^3$	1000	$g^{11}$	$g^3 + g^2 + g$	1110
$g^4$	$g + 1$	0011	$g^{12}$	$g^3 + g^2 + g + 1$	1111
$g^5$	$g^2 + g$	0110	$g^{13}$	$g^3 + g^2 + 1$	1101
$g^6$	$g^3 + g^2$	1100	$g^{14}$	$g^3 + 1$	1001

Xét ví dụ về đường cong Elliptic trên  $GF(2^4)$ :

$$y^2 + xy = x^3 + g^4x + 1 \quad (a = g^4, b = 1)$$

Bảng bên dưới liệt kê các điểm thuộc đường cong này

(0, 1)	$(g^5, g^3)$	$(g^9, g^{13})$
$(1, g^6)$	$(g^5, g^{11})$	$(g^{10}, g)$
$(1, g^{13})$	$(g^6, g^8)$	$(g^{10}, g^8)$
$(g^3, g^8)$	$(g^6, g^{14})$	$(g^{12}, 0)$
$(g^3, g^{13})$	$(g^9, g^{10})$	$(g^{12}, g^{12})$

Tương tự như nhóm Abel  $E_p(a, b)$ , chúng ta cũng xây dựng một nhóm Abel  $E_{2^m}(a, b)$  gồm các điểm của đường cong Elliptic  $GF(2^m)$  cùng với điểm vô cực  $O$ .

1) Điểm  $O$  là phần tử đơn vị của phép cộng.  $P + O = O + P = O$ .

- 2) Phần tử nghịch đảo của điểm  $P$  trong phép cộng, ký hiệu  $-P$ , là điểm đối xứng với  $P$ , ký hiệu  $P = (x_P, y_P)$  thì  $-P = (x_P, -y_P)$
- 3) Với 2 điểm  $P, Q$  bất kỳ ( $P \neq Q$ ) phép cộng  $R = P + Q$  được xác định bằng công thức:

$$x_R = \Delta^2 + \Delta + x_P + x_Q + a$$

$$y_R = \Delta(x_P + x_R) + x_R + y_P$$

$$\text{Trong đó: } \Delta = \frac{y_Q + y_P}{x_Q + x_P}$$

- 4) phép cộng  $R = P + P$  được xác định bằng công thức:

$$x_R = \Delta^2 + \Delta + a$$

$$y_R = x_P^2 + (\Delta + 1)x_R$$

$$\text{Trong đó: } \Delta = x_P + \frac{y_P}{x_P}$$

## 10.4 Đường cong Elliptic trong mã hóa - ECC

Đối với mã hóa đường cong Elliptic, chúng ta xây dựng hàm một chiều như sau:

Trong nhóm Abel  $E_p(a, b)$  xây dựng từ đường cong Elliptic  $Z_p$ , xét phương trình:

$$Q = P + P + \dots + P = kP \quad (\text{điểm } Q \text{ là tổng của } k \text{ điểm } P, k < p)$$

Cho trước  $k$  và  $P$ , việc tính  $Q$  thực hiện dễ dàng. Tuy nhiên nếu cho trước  $P$  và  $Q$ , việc tìm ra  $k$  là công việc khó khăn. Đây chính là hàm logarit rời rạc của đường cong Elliptic. Ví dụ:

Xét nhóm  $E_{23}(9,17)$  với phương trình :

$$y^2 \bmod 23 = (x^3 + 9x + 7) \bmod 23 \quad a, b, x, y \in Z_{23}$$

Cho điểm  $P=(16, 5)$ ,  $Q=(4, 5)$ , chúng ta chỉ có cách là vét cạn các giá trị của  $k$  từ 2 đến  $p-1$  để tìm ra  $k$ :

$$P = (16,5); 2P = (20,20); 3P = (14,14); 4P = (19,20); 5P = (13,10)$$

$$6P = (7,3); 7P = (8,7); 8P = (12,17); 9P = (4,5)$$

Vì  $9P = Q$  nên ta xác định được  $k = 9$ . Trong thực tế chúng ta sẽ sử dụng đường cong Elliptic  $Z_p$  với giá trị  $p$  lớn, sao cho việc vét cạn là bất khả thi. Hiện nay người ta đã tìm ra phương pháp tìm  $k$  nhanh hơn vét cạn là phương pháp Pollar rho.

Dựa vào hàm một chiều trên chúng ta có 2 cách sử dụng đường cong Elliptic trong lĩnh vực mã hóa là trao đổi khóa EC Diffie-Hellman và mã hóa EC.

### 10.4.1 Trao đổi khóa EC Diffie-Hellman

Trong chương 4 chúng ta đã tìm hiểu vấn đề trao đổi khóa Diffie-Hellman dựa trên tính một chiều của hàm logarit rời rạc. Trong phần này chúng ta cũng xem xét một phương thức trao đổi khóa tương tự dùng hàm một chiều của đường cong Elliptic.

Trước tiên ta chọn một số nguyên  $q$  lớn, với  $q$  là số nguyên tố (nếu sử dụng đường cong Elliptic  $Z_p$ ) hoặc  $q$  có dạng  $2^m$  (nếu chọn đường cong  $GF(2^m)$ ), và chọn 2 tham số  $a, b$  tương ứng để tạo thành nhóm  $E_q(a, b)$ . Ta gọi  $G$  là điểm cơ sở của nhóm nếu tồn tại một số nguyên  $n$  sao cho  $nG = O$ . Số nguyên  $n$  nhỏ nhất như vậy được gọi là hạng của  $G$ .

Trong trao đổi khóa EC Diffie-Hellman, ta chọn một điểm  $G$  có hạng  $n$  lớn, và giao thức trao đổi khóa giữa Alice và Bob tiến hành như sau:

- 1) Alice chọn một số  $n_A < n$  và giữ bí mật số  $n_A$  này. Sau đó trong  $E_q(a, b)$  Alice tính  $P_A = n_A G$  và gửi  $P_A$  cho Bob.
- 2) Tương tự Bob chọn một số bí mật  $n_B$ , tính  $P_B$  và gửi  $P_B$  cho Alice.
- 3) Alice tạo khóa phiên bí mật là  $K = n_A P_B = n_A n_B G$
- 4) Bob tạo khóa phiên bí mật là  $K = n_B P_A = n_B n_A G = n_A n_B G$  (nhóm Abel có tính giao hoán) giống với khóa của Alice.

Trudy có thể chặn được  $P_A$  và  $P_B$ , tuy nhiên chỉ có thể tính được:

$$P_A + P_B = n_A G + n_B G = (n_A + n_B)G$$

Để tính được  $K = n_A n_B G$ , Trudy phải tìm được  $n_A, n_B$  từ  $P_A, P_B$  và  $G$ . Tuy nhiên điều này là bất khả thi như ta đã thấy ở phần trên.

Chú ý: khóa phiên  $K$  là một điểm trong đường cong Elliptic, để sử dụng khóa này cho mã hóa đối xứng như DES hay AES, ta cần chuyển  $K$  về dạng số thường.

#### 10.4.2 Mã hóa và giải mã EC

Tương tự như vấn đề trao đổi khóa, trong vấn đề mã hóa/giải mã, ta cũng chọn các tham số để tạo một nhóm Abel  $E_q(a, b)$  và chọn một điểm cơ sở  $G$  có hạng  $n$  lớn.

Các thành phần khóa riêng và công khai trong mã hóa EC được định nghĩa như sau:

$$K_R = (d, G, q, a, b)$$

$$K_U = (E, G, q, a, b)$$

Trong đó  $d < n$  và  $E = dG$  với  $d$  là một số bí mật do người sinh khóa chọn. Do tính chất của hàm một chiều từ  $E$  và  $G$  không thể suy ra được  $d$ .

Từ đó chúng ta có hai cách thức thực hiện mã hóa/ giải mã như sau:

##### 1) Phương pháp Elgamal:

Giả sử Alice muốn gửi một thông điệp  $M$  cho Bob, trước tiên Alice chuyển  $M$  từ dạng dãy bit sang dạng điểm  $P_M = (x, y)$ . Bản mã  $C_M$  (dùng khóa công khai của Bob) được tính là một cặp điểm như sau:

$$C_M = \{kG, P_M + kE\} \quad \text{với } k \text{ là một số ngẫu nhiên do Alice chọn}$$

Để giải mã dùng khóa riêng, Bob sẽ nhân điểm thứ nhất trong  $C_M$  với  $d$ , sau đó lấy điểm thứ hai trừ cho kết quả:

$$P_M + kE - dkG = P_M + kdG - kdG = P_M$$

Trong phương thức mã hóa, Alice đã che giấu  $P_M$  bằng cách cộng  $P_M$  với  $kE$ . Để giải mã, Bob cần trừ ra lại  $kE$ . Thay vì gửi trực tiếp  $k$  cho Bob để Bob tính  $kE$  (Trudy có thể chặn được), Alice gửi một dấu hiệu là  $kG$ . Dựa vào  $kG$  và  $d$ , Bob có thể tính  $kE$ . Còn Trudy, dù biết  $G$  và  $kG$ , tuy nhiên vẫn không thể tính được  $k$  do tính chất của hàm một chiều.

Ví dụ: chọn  $p = 751$ ,  $a = 1$ ,  $b = 188$  ta có đường cong Elliptic trên  $Z_{751}$  như sau

$$y^2 \bmod 751 = (x^3 + x + 188) \bmod 751 \quad a, b, x, y \in Z_{751}$$

Chọn điểm cơ sở là  $G = (0, 376)$ .

Giả sử Alice cần mã hóa bản rõ là điểm  $P_M = (562, 201)$  dùng khóa công khai  $E = (201, 5)$ . Alice chọn  $k = 386$ . Ta có:

$$386(0, 376) = (676, 558)$$

$$(562, 201) + 386(201, 5) = (385, 328)$$

Vậy bản mã là cặp điểm  $\{ (676, 558), (385, 328) \}$

## 2) Phương pháp Menezes - Vanstone:

Thông điệp  $M$  của Alice được tách thành hai phần  $M = (m_1, m_2)$  sao cho  $m_1, m_2 \in \mathbb{Z}_p$ . Alice chọn một số ngẫu nhiên  $k$ , kết hợp với khóa công khai của Bob, Alice tính điểm  $P$  như sau:

$$P(x_P, y_P) = kE$$

Bản mã  $C_M$  gồm ba thành phần:

$$C_M = \{c_0, c_1, c_2\} = \{kG, x_P m_1 \bmod p, y_P m_2 \bmod p\}$$

Để giải mã dùng khóa riêng, từ dấu hiệu  $kG$ , Bob tính:

$$P(x_P, y_P) = dkG$$

và từ đó tính nghịch đảo của  $x_P^{-1}$  và  $y_P^{-1}$  trong phép modulo  $p$ . Cuối cùng, bản giải mã là:

$$M = \{m_1, m_2\} = \{x_P^{-1} c_1 \bmod p, y_P^{-1} c_2 \bmod p\}$$

Tương tự như phương pháp Elgamal, dù biết  $G$  và  $kG$ , Trudy cũng không thể tính được  $k$  để tính  $P$ .

### 10.4.3 Độ an toàn của ECC so với RSA

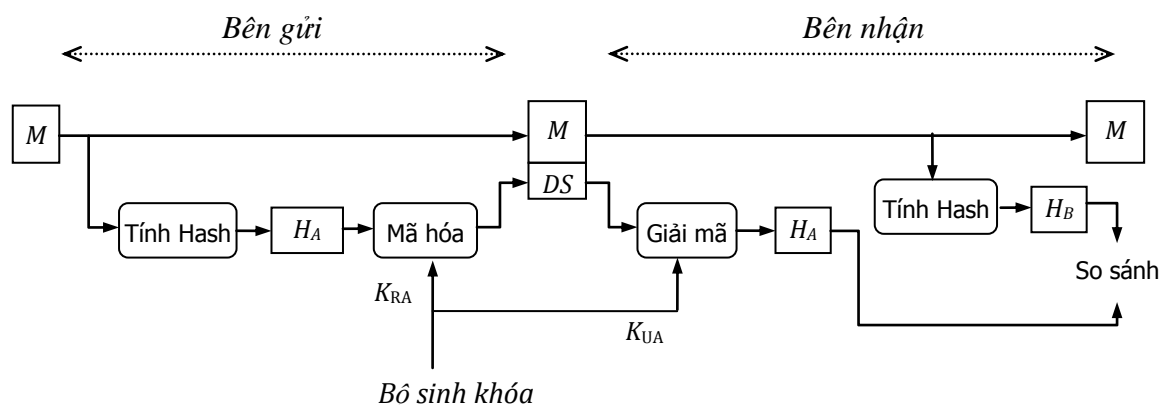
Hiện nay, phương pháp nhanh nhất để tính logarit đường cong Elliptic (tính  $k$  biết  $G$  và  $kG$ ) là phương pháp Pollard rho. Bảng sau đây liệt kê kích thước khóa của phương pháp ECC và phương pháp RSA dựa trên sự tương đương về chi phí phá mã.

Mã hóa đối xứng (số bit của khóa)	Mã hóa ECC (số bit của $n$ )	Mã hóa RSA (số bit của $N$ )
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Như vậy với cùng một độ an toàn thì mã hóa ECC chỉ dùng các phép tính có số bit nhỏ hơn nhiều lần so với mã hóa RSA.

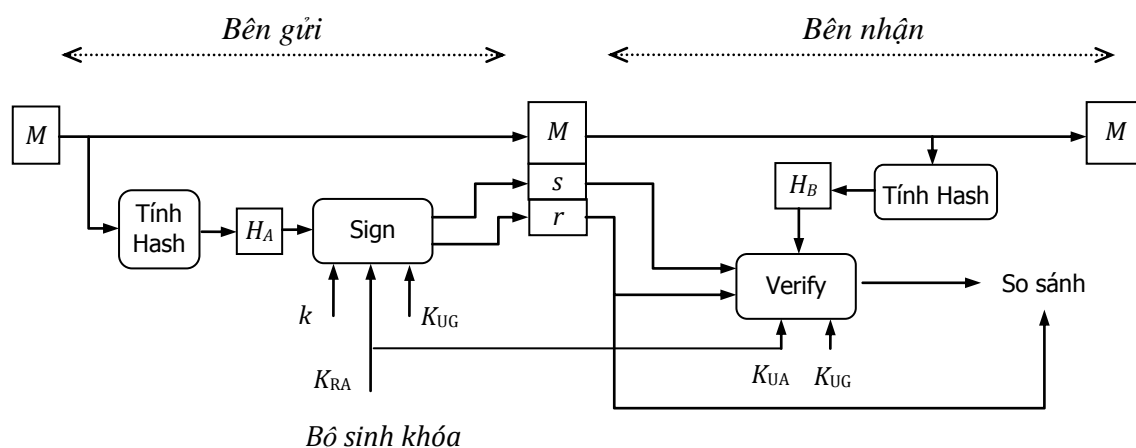
### 10.5 Chuẩn chữ ký điện tử (Digital Signature Standard – DSS)

Trong chương 5, chúng ta đã tìm hiểu về cách sử dụng hàm hash cùng với mã hóa RSA để tạo chữ ký điện tử. Hình bên dưới trình bày lại mô hình này:



*DS: Data signature – chữ ký điện tử*

DSS là một cách thực hiện chữ ký điện tử khác được đề xuất vào năm 1991. Khác với RSA, DSS không thể dùng để mã hóa hay trao đổi khóa. Tuy nhiên về cơ bản DSS cũng thuộc lĩnh vực khóa công khai. Mô hình thực hiện của DSS được minh họa trong hình bên dưới.



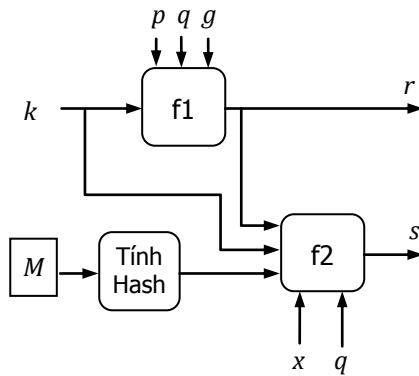
Phương pháp DSS cũng dùng hàm Hash.  $k$  là một số ngẫu nhiên do bên gửi tự chọn.  $K_{UG}$  là một tập các tham số công khai sử dụng chung cho tất cả các bên. Thủ tục Sign sử dụng khóa bí mật để cho ra chữ ký gồm hai tham số  $s$  và  $r$ . Thủ tục Verify sử dụng khóa công khai để cho ra một thông số. Thông số này được so sánh với  $r$  để kiểm tra chữ ký. Chi tiết của thủ tục Sign và Verify được gọi là thuật toán ký (Digital Signature Algorithm – DSA), được trình bày trong phần tiếp theo.

DSA cũng sử dụng hàm một chiều là phép logarithm rời rạc như được dùng trong trao đổi khóa Diffie-Hellman. Tuy nhiên cách thức thực hiện thì theo sơ đồ Elgamal-Schnor. Sơ đồ này gồm các thông số như sau:

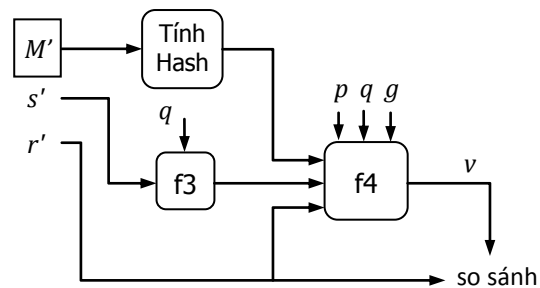
- Thông số công khai toàn cục ( $K_{UG}$ ):
  - $p$  là số nguyên tố,  $2^L < p < 2^{L-1}$  với  $512 \leq L \leq 1024$  và  $L$  chia hết cho 64. (nghĩa là số bit của  $p$  từ 512 đến 1024 và chia hết cho 64)
  - $q$  là một thừa số nguyên tố của  $p-1$ ,  $q$  có chiều dài 160 bit
  - $g = h^{(p-1)/q} \bmod p$  với  $h$  là số nguyên bất kỳ trong khoảng  $(1, p-1)$  và  $h^{(p-1)/q} \bmod p > 1$ .
- Khóa riêng của người gửi ( $K_{RA}$ ): là số ngẫu nhiên  $x$ ,  $1 < x < q$

- Khóa công khai của người gửi ( $K_{UA}$ ):  $y = g^x \bmod p$ . Do tính một chiều của logarithm rời rạc từ  $y, g$  và  $p$  không thể tính lại khóa riêng  $x$
- Số ngẫu nhiên  $k, 1 < k < q$
- Hàm Sign:
  - $r = f1(k, g, p, q) = (g^k \bmod p) \bmod q$ .
  - $s = f2(M, k, r, x, q) = k^{-1}(H(M) + xr) \bmod q$   $M$  là thông điệp cần gửi,  $H$  là hàm SHA-1
  - Output của hàm Sign là cặp  $(r, s)$  đóng vai trò là chữ ký điện tử
- Hàm Verify ( $s', r', M'$  là các giá trị người nhận nhận được):
  - $w = f3(s', q) = (s')^{-1} \bmod q$ .
  - $u1 = [H(M')w] \bmod q$
  - $u2 = (r')w \bmod q$
  - Output của hàm verify:  $v = f4(M', g, w, r', p, q) = [g^{u1}v^{u2} \bmod p] \bmod q$
- $v$  được so sánh với  $r'$ , nếu khớp thì  $M' = M$  chính là thông điệp của người gửi.

Chúng ta có thể biểu diễn lại hàm Sign và Verify trên bằng hình bên dưới:



a) Quá trình ký



b) Quá trình kiểm tra

## CHƯƠNG 11. MỘT SỐ VẤN ĐỀ AN TOÀN BẢO MẬT

### 11.1 Giấu tin trong ảnh số

Tương tự như mã hóa - *cryptography*, giấu tin - *steganography* cũng là một cách thức nhằm che giấu thông tin để cho người ngoài không nhận biết. Cách thực hiện của mã hóa là biến đổi dữ liệu thành một dạng không nhận ra, còn cách thực hiện của giấu tin là sử dụng một vật mang, sau đó đưa thông tin vào vật mang này nhằm che giấu thông tin, người khác không nhận biết được là có thông tin trong vật mang đó. Phần này trình bày một kỹ thuật giấu tin đơn giản, trong đó vật mang là một ảnh Bitmap. Tên gọi của kỹ thuật này là LSB (Least Significant Bit).

Trên máy tính, một tấm ảnh được biểu diễn dưới dạng một ma trận 2 chiều các điểm ảnh. Mỗi điểm ảnh mang một giá trị màu sắc xác định (xanh, đỏ, vàng,...). Tập hợp các giá trị màu sắc của các điểm ảnh này tạo nên cảm nhận của con người về nội dung tấm ảnh.

Ở đây chúng ta chỉ xem xét một định dạng ảnh Bitmap phổ biến là định dạng RGB 24 bit, tức mỗi điểm ảnh là một giá trị 24 bit của 3 màu đỏ (R), xanh lá (G), và xanh lam (B), mỗi màu 8 bit. Sự kết hợp 3 màu này tạo thành màu sắc mong muốn. Như vậy mỗi màu có 255 giá trị biểu diễn mức độ đóng góp của màu đó vào màu sắc cuối cùng. Ví dụ:

- R=255: màu sắc có hàm lượng đỏ cao (đỏ, đỏ tươi, cam, hồng...)
- R=0: màu sắc không có hàm lượng đỏ (xanh, xanh da trời, xanh lá, xanh lơ...)
- G=255: màu sắc có hàm lượng xanh lá cây cao.

Bảng dưới là ví dụ một số màu sắc kết hợp từ 3 màu R,G,B:

(R, G, B)	Màu
255, 0, 0	Đỏ
0, 255, 0	Xanh lá
0, 0, 255	Xanh lam
0, 0, 0	Đen
255, 255, 255	Trắng
255, 255, 0	Vàng
128, 0, 0	Đỏ sậm
255, 128, 0	Cam

Mỗi màu R, G, B được biểu diễn bởi 8 bit, do đó nếu ta thay đổi bit cuối cùng (bit thứ 8, least significant bit) thì giá trị màu chỉ thay đổi một đơn vị, ví dụ: 255→254, 198→199, 25→24, 72→73,... Việc thay đổi này có tác động rất ít đến màu sắc cuối cùng mà ***mắt người không phân biệt được***. Đây là đặc điểm chính để tiến hành giấu tin vào ảnh bitmap, 1 bit dữ liệu được giấu vào 8 bit màu. Ví dụ:

Giá trị màu R	Bit giấu	Màu kết quả
00110001 (25)	0	00110000 (24)
	1	00110001 (25)
01001000 (72)	0	01001000 (72)
	1	01001001 (73)

Một điểm ảnh có thể giấu được 3 bit dữ liệu. Do đó, một tấm ảnh RGB 24 bit kích thước  $m \times n$  có thể giấu được  $(m \times n \times 3/8)$  byte dữ liệu.

Dĩ nhiên cách giấu tin như trên là rất đơn giản, nếu người ngoài biết được quy tắc giấu thì có thể do ra nội dung được giấu. Trong thực tế, người ta dùng một khóa bí mật và dựa trên khóa này để lựa chọn ra một số điểm ảnh dùng cho việc giấu tin mà thôi.

Ngoài ảnh số, âm thanh cũng có thể dùng để giấu tin vì con người cũng không thể phát hiện ra những sự thay đổi nhỏ trong tín hiệu âm thanh.

## 11.2 Lỗi phần mềm

Các phần mềm luôn luôn có lỗi. Những lỗi này làm cho phần mềm hoạt động không như ý muốn người dùng. Tàu hạ cánh Mars Lander của NASA đã đâm vào sao Hỏa do lỗi phần mềm trong việc chuyển đổi từ đơn vị đo Anh sang đơn vị metric. Lỗi trong phần mềm quản lý hành lý khiến sân bay Denver khai trương muộn 11 tháng với thiệt hại 1 triệu USD/ngày. Trong phần này chúng ta quan tâm đến một số loại lỗi phần mềm mà hacker có thể lợi dụng để xâm nhập hệ thống thực hiện các hành vi phá hoại.

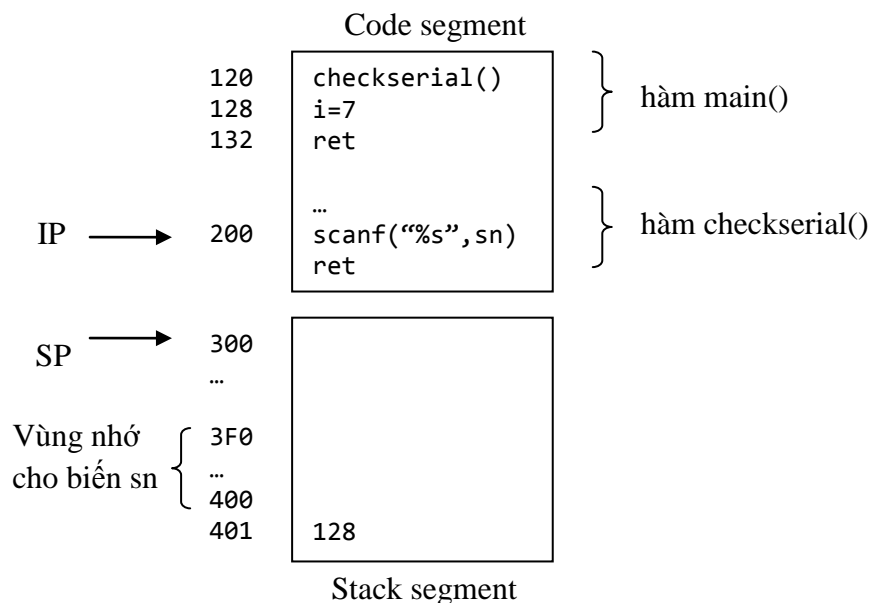
### 11.2.1 Tràn bộ đệm (Buffer Overflow)

Lỗi tràn bộ đệm thường xảy ra đối với loại dữ liệu mảng, khi dữ liệu nhập vào vượt quá kích thước mảng. Ví dụ chương trình sau:

```
void checkserial() {
    char sn[16];
    scanf("%s", sn);
}
int main() {
    checkserial();
    int i= 7;
    return 0;
}
```

Khi hàm `main()` gọi hàm `checkserial()`, trước tiên địa chỉ của lệnh `i= 7` sẽ được push vào stack để sau khi hàm `checkserial` thực hiện xong thì máy tính có thể thi hành tiếp lệnh `i= 7`. Sau đó, máy tính dành tiếp 16 byte trong stack cho mảng `sn`. Hình sau minh họa tình trạng bộ nhớ.





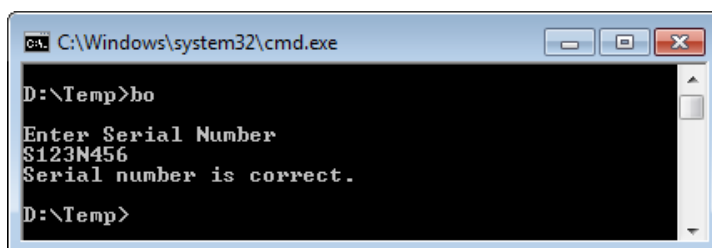
Sau khi hàm `checkserial` thực hiện xong, lệnh `RET` sẽ nạp lại giá trị 128 tại địa chỉ 401 trong stack vào con trỏ lệnh IP để quay về lại lệnh `i= 7`.

Nếu trong hàm `checkserial`, người sử dụng nhập vào chuỗi ít hơn 16 ký tự thì chương trình hoạt động bình thường, tuy nhiên nếu người sử dụng nhập vào chuỗi 16 ký tự trở lên thì lúc này ô nhớ 401 sẽ bị đè bởi ký tự thứ 16, tình trạng tràn bộ đệm xảy ra. Lúc này khi lệnh `RET` của hàm `checkserial` thực hiện, con trỏ lệnh IP sẽ có 1 giá trị khác chứ không phải là 128, do đó lệnh `i= 7` sẽ không được thực hiện. Hacker có thể lợi dụng điều này để tiến hành các hoạt động phá hoại. Xét chương trình cụ thể sau:

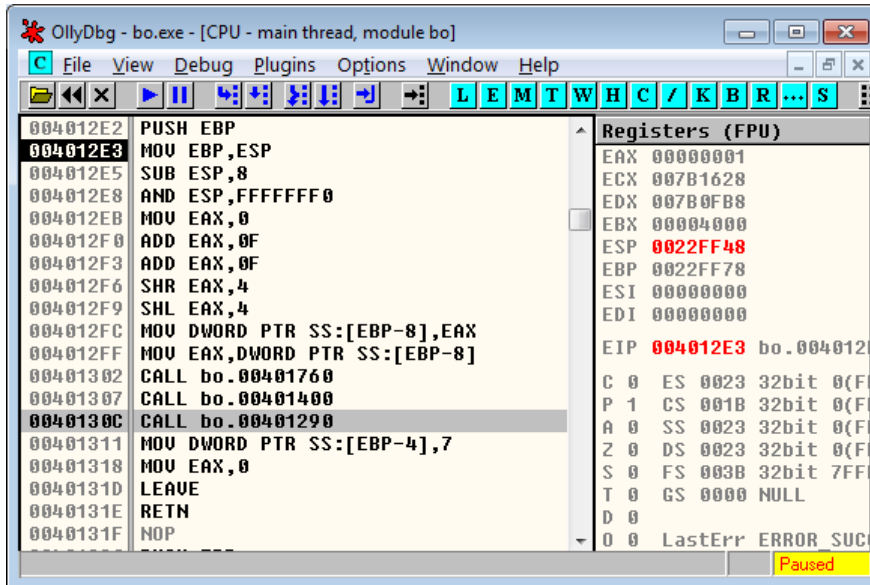
```
void checkserial() {
    char sn[16];
    printf("\nEnter a serial number\n");
    scanf("%s", sn);
    if (!strncmp(sn, "S123N456", 8)) {
        printf("Serial number is correct");
    }
}

int main() {
    checkserial();
    int i=7;
    return 0;
}
```

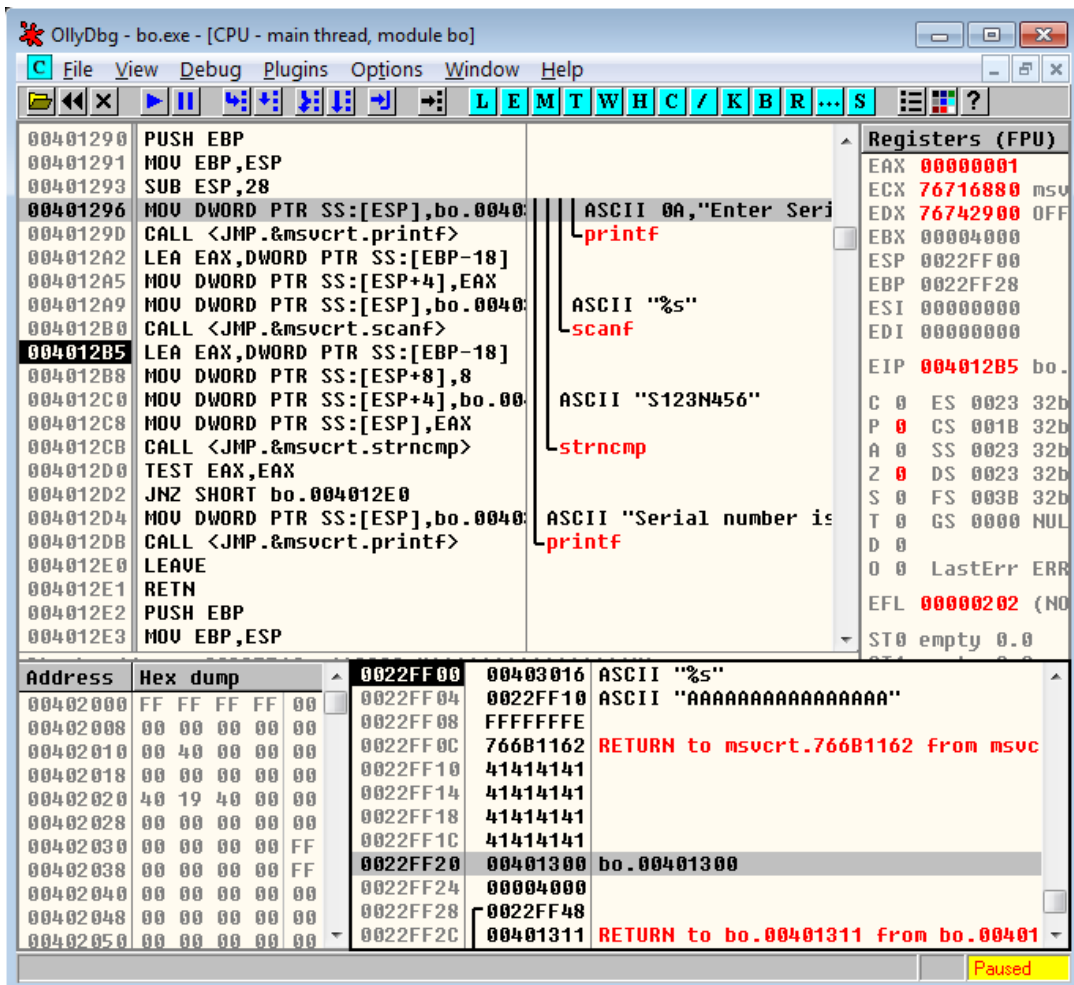
Mục đích của chương trình trên là khi người dùng nhập vào chuỗi "S123N456" thì chương trình sẽ in ra câu "Serial number is correct", nếu không thì không in gì cả.



Lợi dụng lỗi tràn bộ đệm hacker sẽ tìm cách nhập vào một chuỗi gì đó (khác “S123N456”) mà chương trình vẫn in ra câu “Serial number is correct”. Chúng ta sẽ minh họa cách thức thực hiện bằng chương trình OllyDebugger.



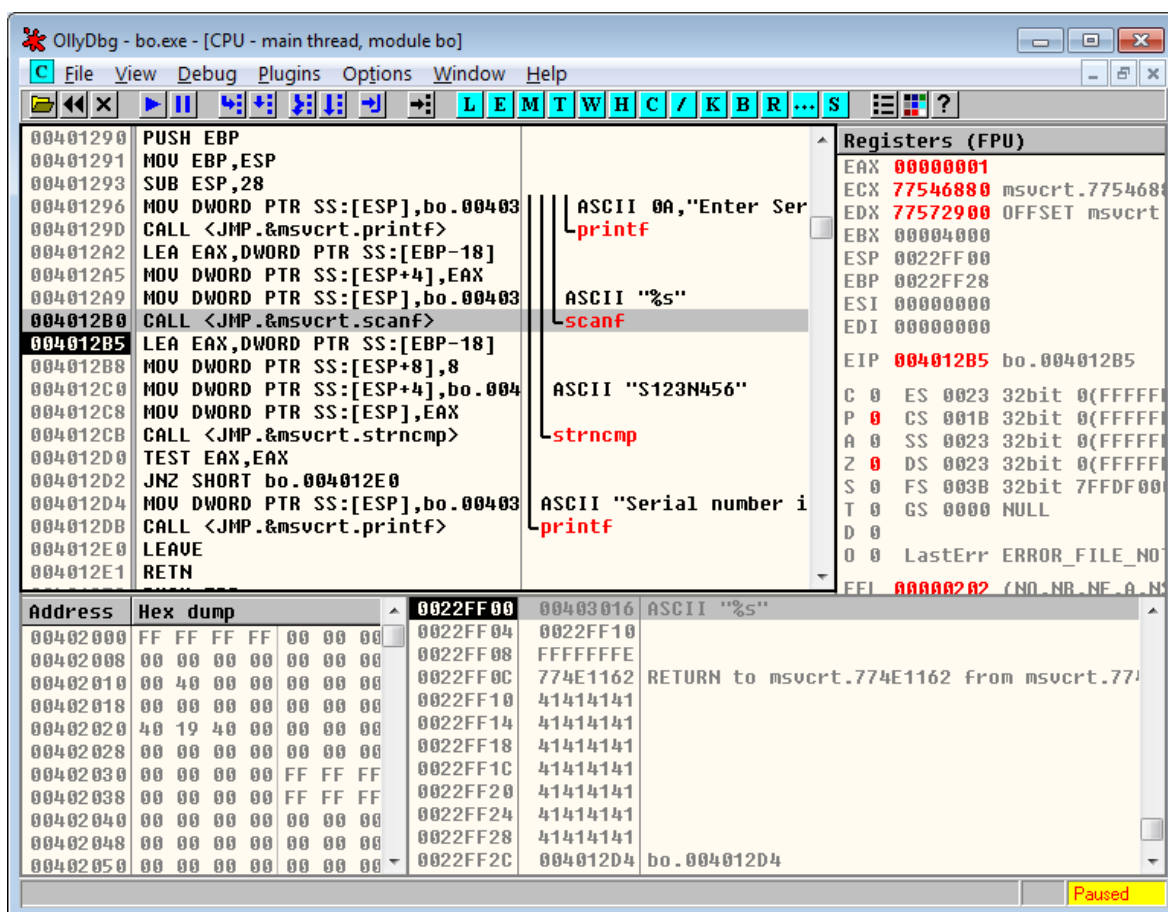
Hình trên minh họa hàm main được nạp vào bộ nhớ. Tại địa chỉ 0040130C là lệnh gọi hàm checkserial, tại địa chỉ 00401311 là lệnh để thực hiện lệnh `i = 7`. Khi thực hiện lệnh gọi hàm checkserial thì tình trạng của Stack segment như sau:



Địa chỉ quay về hàm `main` (tại lệnh `i=7`) `00401311` được đưa vào stack tại địa chỉ `0022FF2C`. Mảng sn được cấp 16 byte bắt đầu tại địa chỉ `0022FF10` đến địa chỉ `0022FF1F` (từ ô `0022FF20` đến `0022FF2B`, gồm 12 byte, bỏ trống). Do khi thực hiện hàm `scanf`, nếu người dùng nhập vào 32 ký tự, thì các ký tự thứ 29, 30, 31, 32 sẽ đè lên địa chỉ quay về `00401311` tạo thành một địa chỉ quay về mới. Hacker có thể lựa chọn giá trị nhập vào sao cho địa chỉ quay về là theo ý của hacker. Giả sử hacker muốn in ra câu “Serial number is correct”, hacker có thể chọn giá trị nhập vào sao cho địa chỉ quay về là `004012D4` (nếu biểu diễn bằng ký tự ASCII, 40: @; D4: ¤; 12: Ctrl+R). Do đó hacker sẽ nhập vào chuỗi sau:

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA^R@

Lúc này tình trạng bộ nhớ stack sẽ là:



(41 là mã ASCII của ký tự A)

Ô nhớ `0022FF2C` trong stack bây giờ có giá trị `004012D4`. Do đó, sau khi hàm `checkserial` thực hiện xong thì lệnh `RETN` (tại ô nhớ `004012E1`) sẽ không nhảy đến lệnh `i=7` của hàm `main` nữa mà nhảy đến lệnh in câu “Serial number is correct” tại ô nhớ `004012D4`. Lệnh này in ra màn hình như bên dưới.

```

C:\Windows\system32\cmd.exe
D:\Temp>bo

Enter Serial Number
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA L^RQ
Serial number is correct.

D:\Temp>

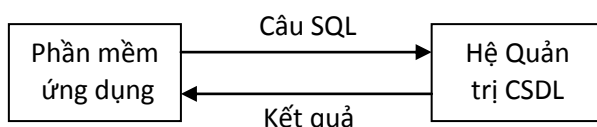
```

Lưu ý: Có thể thực hiện kết quả như trên mà không cần biết source code, chỉ cần dùng chương trình debug. Source code phần trên mang tính chất minh họa.

Nếu sn là một mảng dài vài trăm ký tự thì hacker có thể chèn vào một đoạn lệnh phá hoại (shell code - trong ví dụ trên ta đơn giản chỉ nhập các ký tự A) và sau đó thi hành đoạn lệnh phá hoại này. Đây chính là cách thức mà sâu Code Red vào năm 2001 đã sử dụng để lây nhiễm vào hơn 750.000 máy tính trên khắp thế giới, dựa vào một lỗi buffer overflow trong phần mềm Microsoft IIS.

### 11.2.2 Chèn câu lệnh SQL (SQL Injection)

Trong các phần mềm ứng dụng sử dụng cơ sở dữ liệu quan hệ như Oracle, SQL Server, MySQL, các phần mềm thường sử dụng câu truy vấn SQL (Structure Query Language) để gửi yêu cầu thao tác dữ liệu đến hệ quản trị CSDL. Hệ quản trị CSDL xử lý câu SQL và gửi trả lại dữ liệu kết quả cho phần mềm.



Khác với ngôn ngữ lập trình, câu SQL không được biên dịch sẵn. Chỉ khi nào phần mềm ứng dụng tạo câu SQL và gửi cho Hệ quản trị CSDL thì lúc đó Hệ quản trị CSDL mới biên dịch và thực hiện câu SQL. Trong quá trình tạo câu SQL, phần mềm ứng dụng thường sử dụng tham số do người dùng nhập vào. Đây chính là đặc điểm mà hacker có thể lợi dụng, tiến hành thay đổi câu SQL theo ý riêng của hacker.

Để minh họa, chúng ta xét chức năng đăng nhập mà hầu hết các phần mềm đều có. Để quản lý người dùng, người lập trình tạo một table Users trong cơ sở dữ liệu như sau (ví dụ dùng hệ quản trị SQL Server).

username	password	email
admin	tu8a9xk	admin@xyz.com
nam	34bux8kt	nam@xyz.com
son	krt87ew	son@xyz.com

Để cho phép người dùng đăng nhập, người lập trình thiết kế một form như sau (ví dụ dùng C# và ADO.NET).

Và xử lý sự kiện nhấn nút Login như sau:

```
private void btnLogin_Click(object sender, EventArgs e)
{
    string sql = " SELECT * FROM Users " +
        " WHERE Username = '" + txtUser.Text + "' AND " +
        " Password = '" + txtPass.Text + "'";
    SqlCommand cmd = new SqlCommand(sql, strConnect);
    SqlDataReader dr = cmd.ExecuteReader();
    if (dr.Read()){
        // login ok
    }
    else{
        // login failed
    }
    dr.Close();
}
```

Nếu người dùng nhập vào user là **admin** và password là **abc** thì câu SQL là:

```
SELECT * FROM Users WHERE Username='admin' AND Password='abc'
```

Câu SQL này hoạt động bình thường theo đúng ý người lập trình. Tuy nhiên nếu hacker nhập vào user là **admin' --** và bỏ trống password thì câu SQL trở thành:

```
SELECT * FROM Users WHERE Username='admin' --' AND Password=''
```

Trong SQL Server dấu – nghĩa là chú thích. Như vậy câu SQL trên cho ra kết quả là một record có username là admin. Điều đó nghĩa là hacker đăng nhập với quyền admin mà không cần biết password.

Ở đây dấu ' và dấu -- mà hacker nhập vào đã làm thay đổi cấu trúc câu SQL mà người lập trình không ngờ tới.

Nếu hacker nhập user là **'; DELETE FROM Users --** thì câu SQL trở thành:

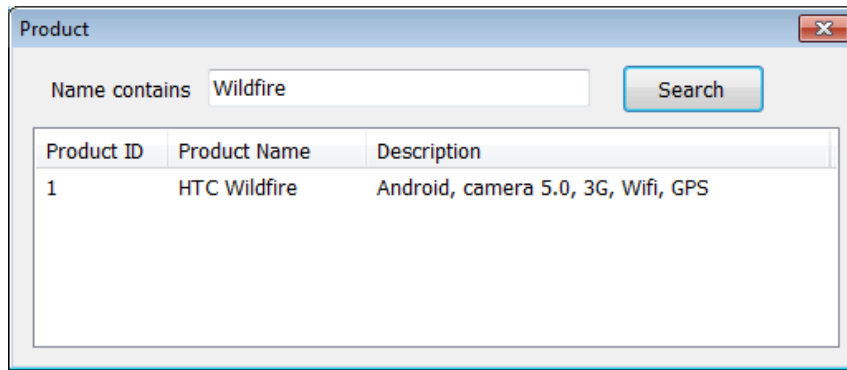
```
SELECT * FROM Users WHERE Username=''; DELETE FROM Users --' AND Password=''
```

Bây giờ có hai câu SQL, câu thứ nhất truy xuất bảng Users và câu thứ 2 xóa toàn bộ bản Users, chương trình bị khóa không đăng nhập được.

Xét ví dụ thứ 2, giả sử chương trình trên là chương trình bán hàng, trong cơ sở dữ liệu có table Products như sau:

ProductID	PName	PDescription
1	HTC Wildfire	Android, camera 5.0, 3G, Wifi, GPS
2	Samsung Omnia	Windows Mobile, Wifi, GPS, FM Radio
3	Motorola Milestone	Android 2.2, camera 8.0, HDMI

Trong phần mềm, chúng ta có một màn hình để tìm kiếm sản phẩm theo tên như sau:



Giả sử câu SQL để tìm kiếm được xây dựng như sau:

```
string sql = " SELECT ProductID, PName, PDescription FROM Products " +
            " WHERE PName like '%" + txtName.Text + "%'";
```

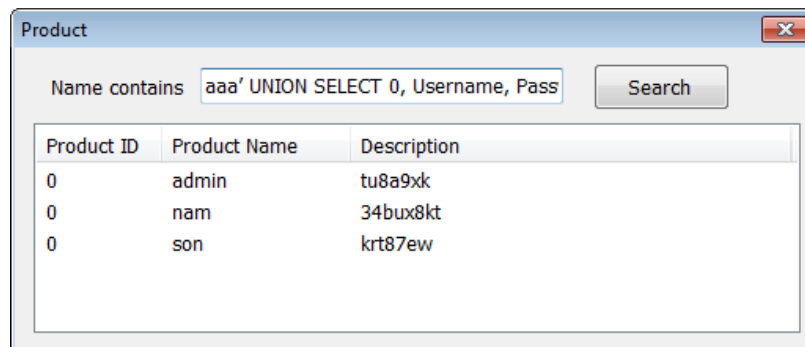
Nếu hacker nhập vào textbox từ tìm kiếm là:

```
aaa' UNION SELECT 0, Username, Password FROM Users --
```

thì câu SQL tạo thành như sau:

```
SELECT ProductID, PName, PDescription FROM Products
WHERE Pname like 'aaa'
UNION SELECT 0, Username, Password FROM Users --'
```

Điều đó có nghĩa là danh sách các người dùng cùng với password sẽ được liệt kê vào danh sách như hình bên dưới:



Để chống lại tấn công SQL Injection, chúng ta sử dụng 2 cách:

- Xử lý các ký tự đặt biệt như dấu ' trong dữ liệu nhập trước khi tạo câu SQL
- Sử dụng parameter để truyền tham số cho câu SQL.

### 11.2.3 Chèn câu lệnh script (Cross-site Scripting XSS)

Ba yếu tố cơ bản để tạo nên một trang web là HTML, CSS và JavaScript. Trong đó JavaScript là một dạng ngôn ngữ lập trình. JavaScript làm cho trang web linh động hơn, giúp nhà phát triển trang web có thể tiến hành một số xử lý ngay tại trình duyệt (client-side) hơn là phải xử lý tại webserver (server-side).

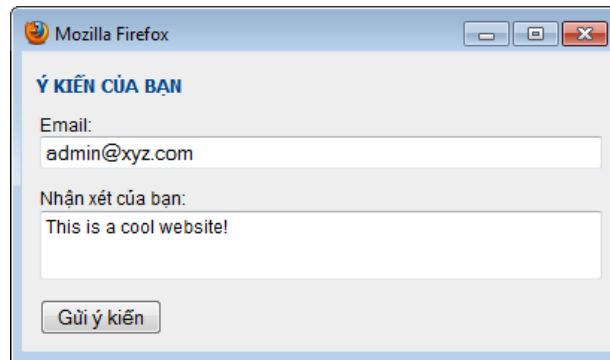
JavaScript là một ngôn ngữ lập trình dạng script, nghĩa là chúng không được biên dịch trước. Khi trình duyệt download trang web, lúc này Javascript mới được biên dịch và thực hiện (giống như câu lệnh SQL, cũng chỉ được biên dịch lúc thi hành). Điều này tạo cơ

hội cho hacker có thể chèn các câu lệnh javascript độc hại vào trang web cũng tương tự như chèn các truy vấn độc hại vào câu SQL. Chúng ta xem xét một ví dụ đơn giản để minh họa cách thức thực hiện của phương pháp tấn công này.

Giả sử có một website cho phép người duyệt post bình luận (comment), cơ sở dữ liệu để lưu trữ bình luận là bảng Comments sau:

CommentID	DatePost	Email	Content
1	12/05/10	admin@xyz.com	This is a cool website!
2	15/06/10	nam@xyz.com	Excellent!!!
3	21/07/10	son@xyz.com	5-stars website!

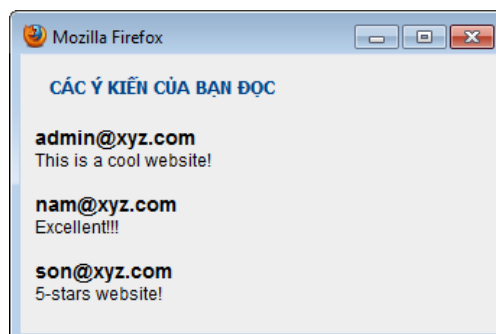
Người lập trình web thiết kế một trang web để post bình luận như sau:



Dùng ngôn ngữ lập trình web, người lập trình tạo một trang HTML để hiển thị các bình luận như sau:

```
<html>
<head> <title> Bình luận </title> </head>
<body>
<h2>CÁC Ý KIẾN CỦA BẠN ĐỌC </h2>
<div>
<h3>admin@xyz.com</h3>
<p> This is a cool website! </p>
</div>
<div>
<h3>nam@xyz.com</h3>
<p> Excellent!!! </p>
</div>
<div>
<h3>son@xyz.com </h3>
<p> 5-stars website! </p>
</div>
</body>
</html>
```

Trang HTML trên hiển thị như hình bên dưới theo đúng ý muốn người lập trình



Tuy nhiên nếu nam@xyz.com là một hacker, và nam@xyz.com nhập một comment như sau:

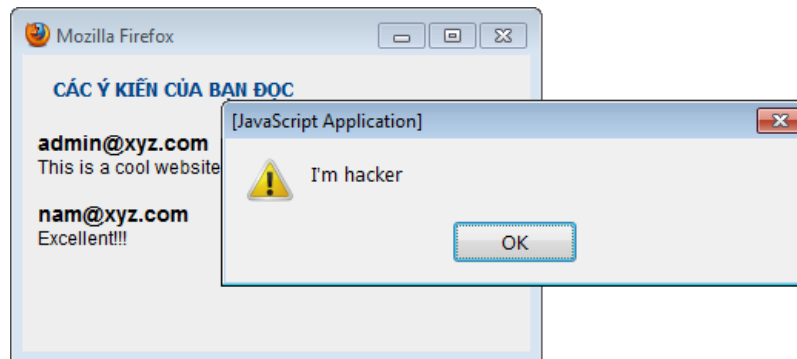
Excellent!!! <script type="text/javascript"> alert("I'm hacker"); </script>

Thì trang HTML trở thành:

```
<html>
<head>   <title> Bình luận </title>   </head>
<body>
<h2>CÁC Ý KIẾN CỦA BẠN ĐỌC </h2>
<div>
    <h3>admin@xyz.com</h3>
    <p> This is a cool website! </p>
</div>
<div>
    <h3>nam@xyz.com</h3>
    <p> Excellent!!! <script type="text/javascript">
                                alert("I'm hacker"); </script>

    </p>
</div>
<div>
    <h3>son@xyz.com </h3>
    <p> 5-stars website! </p>
</div>
</body>
</html>
```

Lúc này đoạn <script type="text/javascript"> alert("I'm hacker"); </script> không còn là nội dung bình luận nữa mà biến thành một đoạn JavaScript có thể thực hiện các lệnh mà người lập trình không mong muốn.



Hay hacker có thể gõ vào một comment như sau:

Excellent!!! <IMG src="http://hackerurl.com/hack.php" />

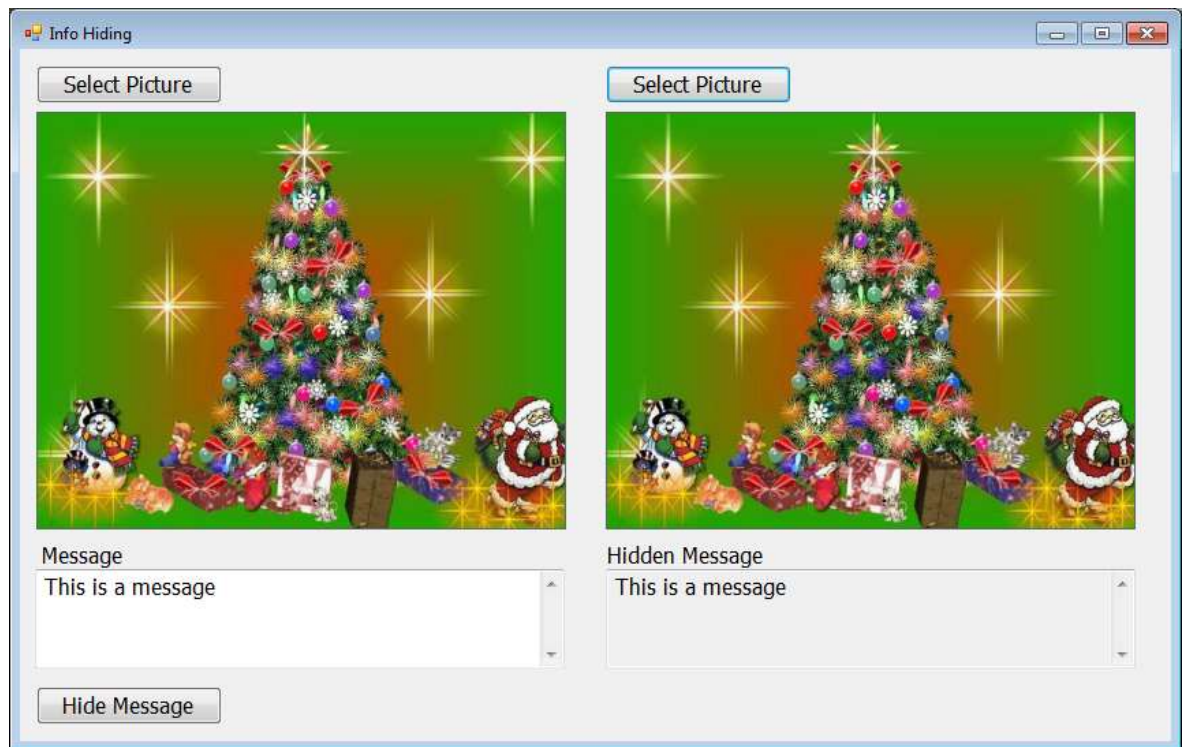
Khi hiển thị trang trình duyệt thấy có thẻ IMG nên sẽ truy xuất địa chỉ <http://hackerurl.com/hack.php>. Đây là một website chứa mã độc của hacker.

Để chống lại lỗi chèn câu lệnh script, chúng ta cần kiểm tra kỹ dữ liệu nhập vào, nếu gặp những ký tự như < và >, cần chuyển chúng sang dạng &lt; và &gt;;

### 11.3 Bài tập thực hành

1. Viết chương trình giấu tin trong ảnh bitmap theo giao diện bên dưới:





2. Viết chương trình và thực hiện tấn công buffer overflow như trong phần 2.1

# PHỤ LỤC 1

## Chi Tiết các S-box của mã hóa DES

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
	1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
	2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
	3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D

DES S-box 1

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
	1	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
	2	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
	3	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9

DES S-box 2

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
	1	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
	2	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
	3	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C

DES S-box 3

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
	1	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
	2	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
	3	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E

DES S-box 4

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
	1	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
	2	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
	3	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3

DES S-box 5

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
	1	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
	2	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
	3	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D

DES S-box 6

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
	1	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
	2	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
	3	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C

DES S-box 7

		$b_1b_2b_3b_4$															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$b_0b_5$	0	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
	1	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
	2	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
	3	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

DES S-box 8

## PHỤ LỤC 2

### Thuật toán Euclid

#### 1) Thuật toán Euclid

Thuật toán Euclid dùng để tìm ước số chung lớn nhất của hai số nguyên  $a$  và  $b$ . Ta ký hiệu ước số chung lớn nhất này là  $\gcd(a, b)$ . Thuật toán này dựa trên định lý sau:

Định lý: với mọi số nguyên  $a \geq 0$  và  $b > 0$  thì:

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

Chứng minh:

Gọi  $d$  là ước số chung lớn nhất của  $a$  và  $b$ . Gọi  $r$  là phần dư của phép chia  $a \bmod b$ :

$$a = bq + r \quad (1)$$

Ta sẽ chứng minh hai điều sau:

- $b$  và  $r$  chia hết cho  $d$ :  
Vì  $a$  và  $b$  đều chia hết cho  $d$  nên từ đẳng thức (1) ta có  $r$  phải chia hết cho  $d$ .
- Không tồn tại  $e > d$  mà  $b$  và  $r$  chia hết cho  $e$ :  
Giả sử tồn tại số  $e > d$  mà  $b$  và  $r$  chia hết cho  $e$ . Như vậy từ đẳng thức (1) ta có  $a$  cũng chia hết cho  $e$ . Vậy  $a$  và  $b$  đều chia hết cho  $e$  là trái với giả thiết  $d$  là ước số chung lớn nhất của  $a$  và  $b$ .

Vậy ước số chung lớn nhất của  $b$  và  $r$  cũng là  $d$  (đpcm).

Vì  $\gcd(b, 0) = b$  nên áp dụng liên tiếp định lý trên cho đến khi  $r = 0$  ta sẽ tìm được  $\gcd(a, b)$ . Cụ thể ta có thuật toán Euclid sau áp dụng cho trường hợp  $a \geq b > 0$ :

/\* Thuật toán Euclid tính  $\gcd(a, b)$  \*/

```
EUCLID (a, b)
  A = a; B = b;
  while B > 0 do
    R = A mod B;
    A = B;
    B = R;
  end while
  return A;
```

Thuật toán được minh họa qua hình sau:

$$\begin{aligned} A_1 &= B_1 q + R_1 \\ A_2 &= B_2 q + R_2 \\ A_3 &= B_3 q + R_3 \\ &\dots \\ A_n &= B_n q + 0 \\ \gcd(a, b) &\leftarrow A_{n+1} \end{aligned}$$

Ví dụ:  $a = 57, b = 42$

$$\begin{aligned} 57 &= 42 \times 1 + 15 \\ 42 &= 15 \times 2 + 12 \\ 15 &= 12 \times 1 + 3 \\ 12 &= 3 \times 4 + 0 \end{aligned}$$

## 2) Thuật toán Euclid mở rộng

Thuật toán này mở rộng thuật toán Euclid ở điểm trong trường hợp  $a$  và  $b$  nguyên tố cùng nhau,  $\gcd(a, b) = 1$  với  $a \geq b > 0$ , thì thuật toán cho biết thêm giá trị nghịch đảo  $b^{-1}$  của  $b$  trong phép chia modulo  $a$  (tức  $bb^{-1} \equiv 1 \pmod{a}$ )

```

/* Thuật toán Euclid mở rộng trả về hai giá trị: */
/* - gcd(a,b); */
/* - nếu gcd(a,b)=1; trả về  $b^{-1} \pmod{a}$  */


---


EXTENDED_EUCLID(a,b)
  A1 = 1; A2 = 0; A3 = a;
  B1 = 0; B2 = 1; B3 = b;
  while (B3<>0)AND(B3<>1) do
    Q = A3 div B3;
    R1 = A1 - QB1;
    R2 = A2 - QB2;
    R3 = A3 - QB3; /*  $\Leftrightarrow A3 \pmod{B3}$  */
    A1 = B1; A2 = B2; A3 = B3;
    B1 = R1; B2 = R2; B3 = R3;
  end while
  If B3=0 then return A3; no inverse;
  If B3=1 then return 1; B2;


---



```

Trước khi vào vòng lặp ta có tính chất sau:

$$aA_1 + bA_2 = A_3 \quad (1)$$

$$aB_1 + bB_2 = B_3 \quad (2)$$

do đó ở lần lặp thứ nhất:

$$\begin{aligned} aR_1 + bR_2 &= aA_1 - aQB_1 + bA_2 - bQB_2 \\ &= A_3 - QB_3 \end{aligned}$$

$$aR_1 + bR_2 = R_3 \quad (3)$$

Vậy trong suốt quá trình lặp của thuật toán các đẳng thức (1), (2), (3) luôn được thỏa mãn.

Trong trường hợp  $\gcd(a, b) \neq 1$ , thuật toán trên hoạt động tương tự như thuật toán Euclid chuẩn ( $A_3$  và  $B_3$  tương tự như  $A$  và  $B$  trong thuật toán chuẩn). Khi kết thúc vòng lặp  $B_3 = 0$ ,  $A_3$  là ước số chung lớn nhất).

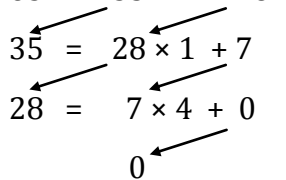
Trong trường hợp  $\gcd(a, b) = 1$ . Theo thuật toán Euclid chuẩn thì  $A_3 = 1$ ,  $B_3 = 0$ . Suy ra trong lần lặp ngay trước đó  $B_3 = 1$ . Trong thuật toán mở rộng vòng lặp sẽ kết thúc khi  $B_3 = 1$ . Ta có:

$$\begin{aligned} aB_1 + bB_2 &= B_3 \\ \Rightarrow aB_1 + bB_2 &= 1 \\ \Rightarrow bB_2 &\equiv 1 \pmod{a} \end{aligned}$$

Vậy  $B_2$  là nghịch đảo của  $b$  trong phép modulo  $m$ .

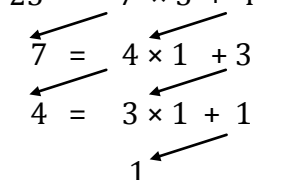
Ví dụ:  $a = 63, b = 35$

A3	B3	Q	R3	A2	B2	Q	R2
63	=	35	$\times 1 + 28$	0	=	1	$\times 1 - 1$
35	=	28	$\times 1 + 7$	1	=	-1	$\times 1 + 2$
28	=	7	$\times 4 + 0$	-1	=	2	$\times 4 - 9$
		0					


 Không có nghịch đảo

Ví dụ:  $a = 25, b = 7$

A3	B3	Q	R3	A2	B2	Q	R2
25	=	7	$\times 3 + 4$	0	=	1	$\times 3 - 3$
7	=	4	$\times 1 + 3$	1	=	-3	$\times 1 + 4$
4	=	3	$\times 1 + 1$	-3	=	4	$\times 1 - 7$
		1				-7	


 Nghịch đảo là:  $-7 + 25 = 18 \quad (7 \cdot 18 = 126 \equiv 1 \pmod{25})$

## Phương pháp kiểm tra số nguyên tố lớn Miller-Rabin

Để kiểm tra xem một số  $p$  có phải là số nguyên tố hay không, một thuật toán cổ điển là kiểm tra xem  $p$  có chia hết cho 2 và  $p$  chia hết cho các số lẻ từ 3 đến  $\lfloor \sqrt{p} \rfloor$  hay không. Nếu  $p$  không chia hết cho các số trên thì  $p$  là số nguyên tố, ngược lại chỉ cần  $p$  chia hết cho một trong các số trên, thì  $p$  không phải là số nguyên tố. Tuy nhiên nếu  $p$  là số nguyên tố lớn thì việc kiểm tra các số như vậy không hiệu quả về mặt thời gian.

Đối với số nguyên tố, ta có hai bổ đề sau:

**Bổ đề 1:** với  $p$  là số nguyên tố,  $x$  là số nguyên,  $x^2 \equiv 1 \pmod{p}$  khi và chỉ khi  $x \equiv 1 \pmod{p}$  hoặc  $x \equiv (p-1) \pmod{p}$ .

**Chứng minh:**  $x^2 \equiv 1 \pmod{p} \Leftrightarrow x^2 - 1 \equiv 0 \pmod{p} \Leftrightarrow (x-1)(x+1) \equiv 0 \pmod{p} (*)$

Vì  $p$  là số nguyên tố nên  $(*)$  tương đương với  $x-1 \equiv 0 \pmod{p}$  hay  $x+1 \equiv 0 \pmod{p}$ . Hay nói cách khác  $x \equiv 1 \pmod{p}$  hay  $x \equiv (p-1) \pmod{p}$ . (đpcm)

**Bổ đề 2:** với  $p$  là số nguyên tố, viết lại  $p$  dưới dạng  $p = 2^k q + 1$  trong đó  $q$  là số lẻ. Với  $a$  là số nguyên dương nhỏ hơn  $p$ , ta có kết luận sau:

\*) Hoặc  $a^q \equiv 1 \pmod{p}$

\*\*) Hoặc trong dãy số  $a^q, a^{2q}, a^{4q}, a^{8q}, \dots, a^{2^{k-1}q}$  tồn tại một số mà đồng dư với  $p-1 \pmod{p}$

**Chứng minh:**

Đặt  $x = a^q$  ta viết lại dãy số trên thành  $x, x^2, x^4, x^8, \dots, x^{2^{k-1}}$

Theo định lý Fermat, ta có  $a^{p-1} \equiv 1 \pmod p$  suy ra  $a^{2^k q} \equiv 1 \pmod p$

hay  $x^{2^k} \equiv 1 \pmod p$

Như vậy trong dãy số  $x, x^2, x^4, x^8, \dots, x^{2^{k-1}}, x^{2^k}$  có số cuối cùng đồng dư với 1. Vận dụng bổ đề 1, ta có kết luận sau:

- Hoặc là  $x \equiv 1 \pmod p$  và do đó các phần tử còn lại trong dãy đều đồng dư với 1. Trong trường hợp này ta có kết luận \*).
- Hoặc là có một số  $x^{2^z} \not\equiv 1 \pmod p$  ( $z < k$ ) tuy nhiên  $x^{2^{z+1}} \equiv 1 \pmod p$ . Do đó theo bổ đề 1 thì  $x^{2^z} \equiv (p-1) \pmod p$ . Trong trường hợp này ta có kết luận \*\*). (đpcm)

Như vậy nếu  $p$  là số nguyên tố thì  $p$  phải thỏa mãn hai bổ đề trên. Tuy nhiên mệnh đề ngược lại thì chưa chắc đúng, có nghĩa là một số hợp số cũng có thể thỏa mãn hai bổ đề này.

Từ nhận xét trên, người ta xây dựng thuật toán kiểm tra số nguyên tố Miller-Rabin như sau:

---

```

/* Thuật toán Miller-Rabin kiểm tra tính nguyên tố của số nguyên p */
TEST(p)
    Tìm k, q với k > 0, q lẻ thỏa mãn p = 2kq + 1
    Chọn số ngẫu nhiên a trong khoảng [2, p - 1]
    If aq mod p = 1 Then return "p có thể là số nguyên tố";
    For j = 0 to k-1 do
        If a2jq mod p = p - 1 Then return "p có thể là số nguyên tố";
    return "p không phải là số nguyên tố";

```

---

Ví dụ 1 : kiểm tra số  $p = 29$

$$29 = 2^2 \times 7 + 1 \text{ do đó } k = 2, q = 7.$$

Nếu chọn  $a = 10$ :  $10^7 \pmod{29} = 17$  do đó ta sẽ tiếp tục tính  $(10^7)^2 \pmod{29} = 28$  thủ tục kiểm tra sẽ trả về “có thể là số nguyên tố”.

Nếu chọn  $a = 2$ :  $2^7 \pmod{29} = 12$  do đó ta sẽ tiếp tục tính  $(2^7)^2 \pmod{29} = 28$  thủ tục cũng sẽ trả về “có thể là số nguyên tố”.

Vì vậy, nếu chỉ thử một vài giá trị  $a$ , ta chưa thể kết luận gì về tính nguyên tố của  $p$ . Tuy nhiên nếu thử hết các giá trị  $a$  từ 2 đến 28 ta đều nhận được kết quả “có thể là số nguyên tố”. Vì vậy có thể chắc chắn rằng 29 là số nguyên tố.

Ví dụ 2 : kiểm tra số  $p = 221$

$$221 = 2^2 \times 55 + 1 \text{ do đó } k = 2, q = 55.$$

Nếu chọn  $a = 5$ :  $5^{55} \pmod{221} = 112$  do đó ta sẽ tiếp tục tính  $(5^{55})^2 \pmod{221} = 168$ , do đó thủ tục kiểm tra sẽ trả về “không phải là số nguyên tố”. Điều này đúng vì  $221 = 13 \times 17$ .

Tuy nhiên nếu chọn  $a = 21$ :  $21^{55} \bmod 221 = 200$  do đó ta sẽ tiếp tục tính  $(21^{55})^2 \bmod 29 = 220$ , lúc này thủ tục sẽ trả về “có thể là số nguyên tố”. Nghĩa là trong một số trường hợp của  $a$ , thuật Miller-Rabin không xác định được tính nguyên tố của 221.

Người ta đã tính được xác suất để trong trường hợp  $p$  là hợp số, thuật toán Miller-Rabin đưa ra khẳng định “không phải là số nguyên tố” là 75%. Trong 25% còn lại, Miller-Rabin không xác định được  $p$  nguyên tố hay hợp số. Do đó nếu chúng ta áp dụng thuật toán  $t$  lần (mỗi lần với các giá trị  $a$  khác nhau) thì xác suất không xác định (trong cả  $t$  lần) là  $(0.25)^t$ . Với  $t$  bằng 10, xác suất trên là rất bé, nhỏ hơn 0.000001.

Tóm lại nguyên tắc kiểm tra tính nguyên tố của số nguyên  $p$  thực hiện như sau:

- Thực hiện thuật toán Miller-Rabin 10 lần với 10 số  $a$  ngẫu nhiên khác nhau.
- Nếu cả 10 lần thuật toán cho ra kết quả “có thể là số nguyên tố”, thì ta khẳng định  $p$  là số nguyên tố.
- Chỉ cần một lần thuật toán cho ra kết quả “không phải là số nguyên tố”, thì ta khẳng định  $p$  là hợp số.

Ví dụ 3:  $p = 41$ ,  $41 = 2^3 \times 5 + 1$  do đó  $k = 3, q = 5, p-1 = 40$ .

$a$	$a^q \bmod p$	$a^{2q} \bmod p$	$a^{4q} \bmod p$
7	38	9	<b>40</b>
8	9	<b>40</b>	
9	9	<b>40</b>	
12	3	9	<b>40</b>
13	38	9	<b>40</b>
16	<b>1</b>		
24	14	32	<b>40</b>
25	<b>40</b>		
31	<b>40</b>		
37	<b>1</b>		

$\Rightarrow 41$  là số nguyên tố

Ví dụ 4:  $p = 133$ ,  $133 = 2^2 \times 33 + 1$  do đó  $k = 2, q = 33, p-1 = 132$ .

$a$	$a^q \bmod p$	$a^{2q} \bmod p$
11	1	
<b>17</b>	<b>83</b>	<b>106</b>
27	132	
30	1	
<b>38</b>	<b>76</b>	<b>57</b>
58	1	
75	132	
94	132	
102	1	
121	1	

$\Rightarrow 133$  không phải là số nguyên tố ( $133 = 7 * 19$ )

Tuy tính toán hơi phức tạp nhưng thuật toán Miller-Rabin là thuật toán kiểm tra số nguyên tố hiệu quả nhất, thực hiện nhanh nhất trong các thuật toán kiểm tra số nguyên tố đã biết.



## Định lý số dư Trung Hoa

Định lý số dư Trung Hoa cho phép thay vì phải thực hiện các phép toán  $\text{mod } T$  trong trường hợp  $T$  lớn, ta có thể chuyển về tính toán trên các phép  $\text{mod } t_i$ , với các  $t_i$  nhỏ hơn  $T$ . Do đó định lý số dư Trung Hoa giúp tăng tốc độ tính toán của thuật toán RSA.

Giả sử:  $T = t_1 \cdot t_2 \dots t_k = \prod_{i=1}^k t_i$ . Trong đó các số  $t_1, t_2, \dots, t_k$  nguyên tố cùng nhau từng đôi một. Xét tập  $Z_T$  và tập  $X$  là tích Decarte của các tập  $Z_{t_i}$  ( $Z_T$  là tập các số nguyên từ 0 đến  $T-1$ ):

$$X = Z_{t_1} \times Z_{t_2} \times \dots \times Z_{t_k}$$

Ta có hai định lý số dư Trung Hoa sau:

**Định lý 1:** Tồn tại một song ánh giữa tập  $Z_T$  và tập  $X$ . Nghĩa là:

- $\forall A \in Z_T, \exists! (a_1, a_2, \dots, a_k) \in X$  sao cho  $A = f(a_1, a_2, \dots, a_k)$
- và  $\forall (a_1, a_2, \dots, a_k) \in X, \exists! A \in Z_T$  sao cho  $(a_1, a_2, \dots, a_k) = f^{-1}(A)$

Chứng minh:

1) *Ánh xạ thuận:* Để chuyển  $A$  thành  $(a_1, a_2, \dots, a_k)$ , ta có thể tính  $a_i = A \text{ mod } t_i$ .

2) *Ánh xạ nghịch:* Để chuyển  $(a_1, a_2, \dots, a_k)$  thành  $A$ , ta thực hiện như sau:

Phương án 1 (do nhà toán học người Trung Quốc Chiu-Shao đề xuất vào năm 1247):

Đặt  $T_i = T/t_i = t_1 \cdot t_2 \dots t_{i-1} \cdot t_{i+1} \dots t_k$ , như vậy  $T_i \equiv 0 \text{ mod } t_j, \forall i \neq j$ . Ngoài ra còn có  $T_i$  nguyên tố cùng nhau với  $t_i$  (theo giả thiết các  $t_i$  đều nguyên tố cùng nhau). Suy ra tồn tại phần tử nghịch đảo  $T_i^{-1}$  sao cho:  $T_i T_i^{-1} \equiv 1 \text{ mod } t_i$ .

Ta tính  $A$  bằng công thức:

$$A = (a_1 T_1 T_1^{-1} + a_2 T_2 T_2^{-1} + \dots + a_n T_n T_n^{-1}) \text{ mod } T$$

Để bảo đảm ánh xạ nghịch là đúng, ta cần chứng minh  $a_i = A \text{ mod } t_i$ . Ta có:

$$\begin{aligned} A \text{ mod } t_i &= ((a_1 T_1 T_1^{-1} + a_2 T_2 T_2^{-1} + \dots + a_n T_n T_n^{-1}) \text{ mod } T) \text{ mod } t_i \\ &= (a_1 T_1 T_1^{-1} + a_2 T_2 T_2^{-1} + \dots + a_n T_n T_n^{-1}) \text{ mod } t_i \quad (\text{vì } T \text{ chia hết cho } t_i) \\ &= a_1 T_1 T_1^{-1} \text{ mod } t_i + \dots + a_i T_i T_i^{-1} \text{ mod } t_i + \dots + a_n T_n T_n^{-1} \text{ mod } t_i \\ &= a_i T_i T_i^{-1} \text{ mod } t_i \quad (\text{vì } T_j \equiv 0 \text{ mod } t_i, \forall j \neq i) \\ &= a_i 1 \text{ mod } t_i \quad (\text{vì } T_i T_i^{-1} \equiv 1 \text{ mod } t_i) \\ &= a_i \quad (\text{đpcm}) \end{aligned}$$

Phương án 2 (do nhà toán học H.L.Garner đề xuất vào năm 1959):

Trong phương án này dùng thuật toán Euclid mở rộng, chúng ta lập  $C_n^2$  hằng số  $c_{ji}$  với  $1 \leq j < i \leq k$  như sau:

$$c_{ji} t_j \equiv 1 \text{ mod } t_i$$

$j$	$i$	$t_1$	$t_2$	$t_3$	$t_4$
$t_1$			$c_{12}$	$c_{13}$	$c_{14}$
$t_2$				$c_{23}$	$c_{24}$
$t_3$					$c_{34}$
$t_4$					

Và tính  $k$  giá trị trung gian  $b_i$  như sau:

$$b_1 = a_1 \bmod t_1$$

$$b_2 = (a_2 - b_1)c_{12} \bmod t_2$$

$$b_3 = ((a_3 - b_1)c_{13} - b_2)c_{23} \bmod t_3$$

....

$$b_k = (...((a_k - b_1)c_{1k} - b_2)c_{2k} - \dots - b_{k-1})c_{(k-1)k} \bmod t_k$$

Và  $A$  được tính theo công thức:

$$A = b_k t_{k-1} \dots t_2 t_1 + \dots + b_3 t_2 t_1 + b_2 t_1 + b_1$$

Để bảo đảm ánh xạ nghịch là đúng, ta cần chứng minh  $a_i = A \bmod t_i$ . Ta có:

$$\begin{aligned} A \bmod t_i &= (b_k t_{k-1} \dots t_2 t_1 + \dots + b_3 t_2 t_1 + b_2 t_1 + b_1) \bmod t_i \\ &= (b_i t_{i-1} \dots t_2 t_1 + \dots + b_3 t_2 t_1 + b_2 t_1 + b_1) \bmod t_i \end{aligned}$$

Ta có:

$$\begin{aligned} b_i &= (...((a_i - b_1)c_{1i} - b_2)c_{2i} - \dots - b_{i-1})c_{(i-1)i} \\ &= a_i c_{1i} c_{2i} \dots c_{(i-1)i} - b_1 c_{1i} c_{2i} \dots c_{(i-1)i} - b_2 c_{2i} c_{3i} \dots c_{(i-1)i} - \dots - b_{i-1} c_{(i-1)i} \end{aligned}$$

Đặt  $X = b_i t_{i-1} \dots t_2 t_1 \bmod t_i$ , ta có:

$$\begin{aligned} X &= (a_i c_{1i} t_1 c_{2i} t_2 \dots c_{(i-1)i} t_{i-1} \bmod t_i \\ &\quad - b_1 c_{1i} t_1 c_{2i} t_2 \dots c_{(i-1)i} t_{i-1} \bmod t_i \\ &\quad - b_2 t_1 c_{2i} t_2 \dots c_{(i-1)i} t_{i-1} \bmod t_i \\ &\quad - b_3 t_1 t_2 c_{3i} t_3 \dots c_{(i-1)i} t_{i-1} \bmod t_i \\ &\quad \dots \\ &\quad - b_{i-1} t_1 t_2 \dots t_{i-2} c_{(i-1)i} t_{i-1} \bmod t_i) \bmod t_i \\ &= (a_i - b_1 - b_2 t_1 - b_3 t_1 t_2 - \dots - b_{i-1} t_1 t_2 \dots t_{i-2}) \bmod t_i \end{aligned}$$

Vậy ta có kết luận:

$$A \bmod t_i = a_i$$

**Định lý 2:** Các phép toán số học modulo thực hiện trên  $Z_M$  có thể được thực hiện bằng  $k$  phép toán tương tự lần lượt trên:  $Z_{t_1}, Z_{t_2}, \dots, Z_{t_k}$ . Cụ thể, nếu  $A \leftrightarrow (a_1, a_2, \dots, a_k)$ ,  $B \leftrightarrow (b_1, b_2, \dots, b_k)$  thì:

$$(A + B) \bmod T \leftrightarrow ((a_1 + b_1) \bmod t_1, (a_2 + b_2) \bmod t_2, \dots, (a_k + b_k) \bmod t_k)$$

$$(A - B) \bmod T \leftrightarrow ((a_1 - b_1) \bmod t_1, (a_2 - b_2) \bmod t_2, \dots, (a_k - b_k) \bmod t_k)$$

$$(A \times B) \bmod T \leftrightarrow ((a_1 \times b_1) \bmod t_1, (a_2 \times b_2) \bmod t_2, \dots, (a_k \times b_k) \bmod t_k)$$

Dựa vào định lý này nếu  $A, B, M$  là các số rất lớn thuộc không gian  $Z_T$  khó tính toán, ta có thể chuyển đổi  $A, B, M$  về dạng  $a_i, b_i, t_i$ . Sau đó thực hiện tính toán trên không gian các tập  $Z_{t_i}$ , cuối cùng chuyển ngược kết quả lại về không gian  $Z_T$ . Do đó nếu số phép tính nhiều, thì việc thực hiện trên không gian các tập  $Z_{t_i}$  sẽ mang lại hiệu quả cao so với chi phí chuyển đổi.

**Ví dụ định lý số dư Trung Hoa:**

Cho  $T = 1813 = 37 \times 49$ . Tính  $X+Y = 678+973 \mod 1813$ .

Ta có  $t_1 = 37, t_2 = 49$ .

Vậy  $X$  được biểu diễn thành:  $(678 \mod 37, 678 \mod 49) = (12, 41)$ .  $Y$  được biểu diễn thành  $(973 \mod 37, 973 \mod 49) = (11, 42)$ . Do đó:

$$(678+973) \mod 1813 = ((12+11) \mod 37, (41+42) \mod 49) = (23, 34)$$

Và cuối cùng kết quả của phép cộng là:

Theo phương án 1:

$$T_1 = 49, T_2 = 37$$

$$T_1^{-1} = 34, T_2^{-1} = 4$$

$$X + Y = 23 \times 49 \times 34 + 34 \times 37 \times 4 \mod 1813$$

$$= 38318 + 5032 \mod 1813$$

$$= 43350 \mod 1813$$

$$= 1651 \text{ (chính là } 678 + 973)$$

Theo phương án 2:

$$c_{12} = 4$$

$$b_1 = 23, b_2 = (34 - 23)4 \mod 49 = 44$$

$$X + Y = b_1 + b_2 t_1 = 23 + 44.37 = 1651.$$

**Cài đặt giao thức SSL cho Web server IIS**

(Xem nội dung tại MSOpenLab <http://msopenlab.com/index.php?article=68>)

## TÀI LIỆU THAM KHẢO

- [1]. Bảo mật thông tin, mô hình và ứng dụng – Nguyễn Xuân Dũng – Nhà xuất bản Thống Kê – 2007.
- [2]. Cryptography and Network Security Principles and Practices, 4<sup>th</sup> Edition – William Stallings – Prentice Hall – 2005.
- [3]. Information Security Principles and Practices – Mark Stamp – John Wiley&Son, Inc – 2006.
- [4]. Applied Cryptography, 2<sup>nd</sup> Edition – Bruce Schneier – John Wiley&Son, Inc – 1996.
- [5]. AES Proposal: Rijndael Block Cipher– Joan Daemen, Vincent Rijmen.
- [6]. Differential Cryptanalysis of DES-like cryptosystem – Adi Shamir, Eli Biham. - 1990
- [7]. Linear Cryptanalysis Method for DES cipher – Matsui – Springer-Verlag – 1998
- [8]. Guide to elliptic curve cryptography – Hankerson, Menezes, Vanstone – Springer, 2004
- [9]. How Secure Is Your Wireless Network – Lee Barken – Prentice Hall – 2003