BỘ GIAO THÔNG VẬN TẢI
TRƯỜNG ĐẠI HỌC HÀNG HẢI
BỘ MÔN: KHOA HỌC MÁY TÍNH
KHOA: CÔNG NGHỆ THÔNG TIN

Giáo trình

AN TOÀN VÀ BẢO MẬT THÔNG TIN

TÊN HỌC PHẦN : **An toàn và bảo mật Thông tin**
MÃ HỌC PHẦN : 17212
TRÌNH ĐỘ ĐÀO TẠO : **ĐẠI HỌC CHÍNH QUY**
DÙNG CHO SV NGÀNH : **CÔNG NGHỆ THÔNG TIN**

HẢI PHÒNG - 2014

Đề cương học phần

Tên học phần: An toàn và bảo mật thông tin **Mã HP: 17212**

a. Số tín chỉ: *03 TC* **BTL** ☐ **ĐAMH** ☐

b. Đơn vị giảng dạy: Bộ môn Khoa học máy tính.

c. Phân bổ thời gian:

- Tổng số (TS): 51 tiết.
- Lý thuyết (LT): 36 tiết.
- Thực hành (TH): 12 tiết.
- Bài tập (BT): 0 tiết.
- Hướng dẫn BTL/ĐAMH (HD): 0 tiết.
- Kiểm tra (KT): 3 tiết.

d. Điều kiện đăng ký học phần:

Học phần này được bố trí sau các học phần: Lập trình hướng đối tượng; Cấu trúc dữ liệu và giải thuật.

e. Mục đích của học phần:

Kiến thức:

- Tổng quan học phần và ứng dụng thực tiễn trong CNTT
- Mật mã đối xứng, công khai
- Hàm băm, chữ ký số, giao thức mật mã.

Kỹ năng:

- Nắm được các kiến thức cơ bản của học phần
- Có khả năng áp dụng lý thuyết để cài đặt các chương trình ứng dụng.

Thái độ nghề nghiệp:

- Hình thành nhận thức các kiến thức cơ bản của học phần và ứng dụng thực tiễn trong CNTT.

f. Tóm tắt nội dung học phần:

Học phần này trình bày các kiến thức cơ sở trong an toàn bảo mật thông tin như các giải thuật mã hóa đối xứng, mã hóa công khai, mã khối kèm theo độ phức tạp, độ an toàn của các giải thuật đó; Các hệ mã khóa bí mật và mã khóa công khai; Các thuật toán tạo hàm băm và chữ ký điện tử; Các giao thức trao chuyển khóa; Cách quản lý khóa và các giao thức mật mã.

g. Người biên soạn: **Phạm Tuấn Đạt – BM Khoa học máy tính, Khoa CNTT**

h. Nội dung chi tiết học phần:

TÊN CHƯƠNG MỤC	PHÂN PHỐI SỐ TIẾT					
	TS	LT	BT	TH	HD	KT
Chương 1. Giới thiệu nhiệm vụ của an toàn và bảo mật thông tin	**3**	**3**	**0**	**0**	**0**	**0**
1.1. Các khái niệm mở đầu		*0,5*				
1.2. Mục tiêu và nguyên tắc chung của ATBMTT.		*0,5*				
1.3. Giới thiệu chung về các mô hình mật mã		*2*				
Chương 2. Cơ sở toán học	**7**	**4**	**0**	**2**	**0**	**1**
2.1. Lý thuyết Thông tin		*0,5*				
2.2. Lý thuyết độ phức tạp		*0,5*				

2.2.1 Độ an toàn tính toán		*0,25*				
2.2.2 Độ an toàn không điều kiện		*0,25*				
2.3. Lý thuyết số học		*2*		*1,5*		
2.3.1 Phép đồng dư, số nguyên tố, ước số chung lớn nhất		0,5				
2.3.2 Giải thuật lũy thừa nhanh		0,5		0,5		
2.3.3 Giải thuật euclit mở rộng		0,5		0,5		
2.3.4 Định lý Trung hoa		0,5		0,5		
Bài kiểm tra số 1						1
2.4. Các giải thuật kiểm tra số nguyên tố		*1*		*0,5*		
2.4.1 Giải thuật Soloway - Strassen		0,25				
2.4.2 Giải thuật Rabin - Miller		0,25				
2.4.3 Giải thuật Lehman		0,5		0,5		
Tự học: *Cài đặt giải thuật euclit mở rộng, Lehman*						
Chương 3. Các hệ mã khóa bí mật	**13**	**8**	**0**	**4**	**0**	**1**
3.1. Các hệ mã cổ điển		*3*		*2*		
3.1.1 Mật mã hóa Cesar		0,5		0,5		
3.1.2 Mật mã Affine		0,5		0,5		
3.1.3 Mật mã Vigenere		1,0		0,5		
3.1.4 Mật mã Hill		1,0		0,5		
3.2. Các hệ mã khối		*5*		*2*		
3.2.1 Mật mã khối		0,5				
3.2.2 Chuẩn mã hóa dữ liệu DES		2		0,5		
3.2.3 TripleDES		0,5		0,5		
3.2.4 Chuẩn mã hóa cao cấp AES		1		0,5		
3.2.5 Các chế độ sử dụng mã khối		1		0,5		
Bài kiểm tra số 2						1
Tự học: *Cài đặt giải thuật mã đối xứng, Des, Aes*						
Chương 4. Các hệ mã mật khóa công khai	**10**	**8**	**0**	**2**	**0**	**0**
4.1. Khái niệm khóa công khai		*0,5*				
4.2. Nguyên tắc cấu tạo hệ khóa công khai		*0,5*				
4.3. Một số giải thuật mã khóa công khai		*7*				
4.3.1 Hệ mã Trapdoor Knapsack		1,5		*1*		
4.3.2 Hệ mã RSA		1,5		0,5		
4.3.3 Hệ mã ElGamal		1,5		0,5		
4.3.4 Hệ mã dựa trên đường cong Eliptic		2,5				

Tự học: *Cài đặt giải thuật RSA*						
Chương 5. Chữ ký điện tử và hàm băm	**11**	**6**	**0**	**4**	**0**	**1**
5.1. Chữ ký điện tử		*2*		*2*		
5.1.1 Định nghĩa		0,5				
5.1.2 Hệ chữ ký điện tử RSA		0,5		1		
5.1.3 Hệ chữ ký điện tử ElGamal		0,5				
5.1.4 Hệ chữ ký điện tử DSS		0,5		1		
5.2. Hàm băm		*1,5*				
5.2.1 Định nghĩa		0,25				
5.2.2 Các tính chất		0,25				
5.2.3 Xung đột hàm băm		1				
Bài kiểm tra số 3						1
5.2.4 Những hàm băm nổi tiếng		2,5		*2*		
Tự học: *Cài đặt giải thuật DSA, hàm băm MD-5*						
Chương 6. Quản lý khóa	**4**	**4**	**0**	**0**	**0**	**0**
6.1. Quản lý khóa trong mạng truyền tin		*0,5*				
6.2. Các hệ phân phối khóa		*2,0*				
6.2.1 Giao thức trao đổi khóa Blom		1,0				
6.2.2 Giao thức trao đổi khóa Diffie-Hellman		0,5				
6.2.3 Giao thức Kerberos		0,5				
6.3. Các hệ thỏa thuận khóa		*1,5*				
6.3.1. Giao thức trao đổi khóa Diffie-Hellman có chứng chỉ		1,0				
6.3.2. Giao thức trao đổi khóa Girault		0,5				
Tự học: *Cài đặt giải thuật trao đổi khóa Diffie-Hellman trên đường cong Eliptic*						
Chương 7. Giao thức mật mã	**3**	**3**	**0**	**0**	**0**	**0**
7.1. Khái niệm giao thức mật mã		*1,5*				
7.1.1 Định nghĩa giao thức mật mã		0,5				
7.1.2 Mục đích giao thức mật mã		0,5				
7.1.3 Các bên tham gia vào giao thức mật mã		0,5				
7.2. Tìm hiểu thiết kế các giao thức mật mã điển hình		*1,5*				
7.2.1.Một số dạng tấn công đối với giao thức mật mã		0,75				
7.2.2.Giới thiệu một số giao thức mật mã		0,75				
Tổng số tiết:	**51**	**36**	**0**	**12**	**0**	**3**

i. Mô tả cách đánh giá học phần:

Sinh viên phải tham dự tối thiểu 75% số giờ lên lớp và phải đạt các điểm thành phần X_2, X_3 từ 4,0 trở lên (X_1 là điểm chuyên cần ≥ 6.0, X_2 là điểm trung bình của ít nhất 02 bài kiểm tra trên lớp).

Điểm học phần (Z) được tính theo công thức: $Z = 0.3X + 0.7Y$

Trong đó:

- X: điểm quá trình, bằng trung bình cộng của X_1, X_2, X_3.
- Y: điểm bài kiểm tra kết thúc học phần
- Hình thức thi: trắc nghiệm trên máy tính; thời gian: 60 phút

Thang điểm đánh giá: A^+, A, B^+, B, C^+, C, D^+, D và F

k. Giáo trình:

Nguyễn Hữu Tuân, *Giáo trình An toàn và bảo mật Thông tin,* NXB Giao thông vận tải, 2008.

l. Tài liệu tham khảo:

1. Phan Đình Diệu. *Lý thuyết mật mã và An toàn thông tin.* Đại học Quốc Gia Hà Nội.
2. Bruce Schneier, *Applied Cryptography*, John Wiley & Sons, 1996.

m. Ngày phê duyệt: 30/06/2014

n. Cấp phê duyệt: Khoa CNTT

Trưởng Khoa	**P. Trưởng Bộ môn**	**Người biên soạn**
TS Lê Quốc Định	***TS. Nguyễn Duy Trường Giang***	***TS. Nguyễn Hữu Tuân***

o. Tiến trình cập nhật Đề cương:

Cập nhật lần 1: *ngày 26/06/2014* **Nội dung**: Rà soát theo kế hoạch Nhà trường gồm: - Chỉnh sửa, làm rõ các Mục **c, e, i** theo các mục tiêu đổi mới căn bản. - Mục **h**: bổ sung Nội dung tự học cuối mỗi chương mục, chuyển một số nội dung giảng dạy sang phần tự học. - Bổ sung các mục **m, n, o**	Người cập nhật *Phạm Tuấn Đạt, Đặng Hoàng Anh* P. Trưởng Bộ môn *Nguyễn Văn Thủy*
Cập nhật lần 2: *ngày / /* **Nội dung**	Người cập nhật Trưởng Bộ môn

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

LỜI NÓI ĐẦU

Từ trước công nguyên con người đã phải quan tâm tới việc làm thế nào để đảm bảo an toàn bí mật cho các tài liệu, văn bản quan trọng, đặc biệt là trong lĩnh vực quân sự, ngoại giao. Ngày nay với sự xuất hiện của máy tính, các tài liệu văn bản giấy tờ và các thông tin quan trọng đều được số hóa và xử lý trên máy tính, được truyền đi trong một môi trường mà mặc định là không an toàn. Do đó yêu cầu về việc có một cơ chế, giải pháp để bảo vệ sự an toàn và bí mật của các thông tin nhạy cảm, quan trọng ngày càng trở nên cấp thiết. Mật mã học chính là ngành khoa học đảm bảo cho mục đích này. Khó có thể thấy một ứng dụng Tin học có ích nào lại không sử dụng các thuật toán mã hóa thông tin. Tài liệu này dựa trên những kinh nghiệm và nghiên cứu mà tác giả đã đúc rút, thu thập trong quá trình giảng dạy môn học An toàn và Bảo mật Thông tin tại khoa Công nghệ Thông tin, Đại học Hàng hải Việt nam. Với bảy chương được chia thành các chủ đề khác nhau từ cơ sở toán học của mật mã học cho tới các hệ mã, các giao thức mật mã, hy vọng sẽ cung cấp cho các em sinh viên, các bạn độc giả một tài liệu bổ ích. Mặc dù đã rất cố gắng song vẫn không tránh khỏi một số thiếu sót, hy vọng sẽ được các bạn bè đồng nghiệp, các em sinh viên, các bạn độc giả góp ý chân thành để tôi có thể hoàn thiện hơn nữa cuốn sách này.

Xin gửi lời cảm ơn chân thành tới các bạn bè đồng nghiệp, những người thân đã luôn động viên, góp ý cho tôi trong quá trình biên soạn. Xin gửi lời cảm ơn tới Thạc sỹ Nguyễn Đình Dương, người đã đọc và cho những nhận xét, góp ý quí báu cho phần viết về hệ mã khóa công khai dựa trên các đường cong Elliptic. Xin gửi lời cảm ơn sâu sắc tới Thạc sỹ Phạm Tuấn Đạt, người đã hiệu đính một cách kỹ càng và cho rất nhiều nhận xét có giá trị cho bản thảo của cuốn sách này. Cuối cùng xin gửi lời cảm ơn tới Ban chủ nhiệm khoa Công nghệ Thông tin, đặc biệt là Tiến sỹ Lê Quốc Định – chủ nhiệm khoa, đã luôn tạo điều kiện tốt nhất, giúp đỡ để cuốn sách này có thể hoàn thành.

Hải phòng, tháng 12 năm 2007

Tác giả

Nguyễn Hữu Tuân

CHƯƠNG I: GIỚI THIỆU

1. An toàn bảo mật thông tin và mật mã học

Trải qua nhiều thế kỷ hàng loạt các giao thức (protocol) và các cơ chế (mechanism) đã được tạo ra để đáp ứng nhu cầu an toàn bảo mật thông tin khi mà nó được truyền tải trên các phương tiện vật lý (giấy, sách, báo …). Thường thì các mục tiêu của an toàn bảo mật thông tin không thể đạt được nếu chỉ đơn thuần dựa vào các thuật toán toán học và các giao thức, mà để đạt được điều này đòi hỏi cần có các kỹ thuật mang tính thủ tục và sự tôn trọng các điều luật. Chẳng hạn sự bí mật của các bức thư tay là do sự phân phát các lá thư đã có đóng dấu bởi một dịch vụ thư tín đã được chấp nhận. Tính an toàn về mặt vật lý của các lá thư là hạn chế (nó có thể bị xem trộm) nên để đảm bảo sự bí mật của bức thư pháp luật đã đưa ra qui định: việc xem thư mà không được sự đồng ý của chủ nhân hoặc những người có thẩm quyền là phạm pháp và sẽ bị trừng phạt. Đôi khi mục đích của an toàn bảo mật thông tin lại đạt được nhờ chính phương tiện vật lý mang chúng, chẳng hạn như tiền giấy đòi hỏi phải được in bằng loại mực và giấy tốt để không bị làm giả.

Về mặt ý tưởng việc lưu giữ thông tin là không có nhiều thay đổi đáng kể qua thời gian. Ngày xưa thông tin thường được lưu và vận chuyển trên giấy tờ, trong khi giờ đây chúng được lưu dưới dạng số hóa và được vận chuyển bằng các hệ thống viễn thông hoặc các hệ thống không dây. Tuy nhiên sự thay đổi đáng kể đến ở đây chính là khả năng sao chép và thay đổi thông tin. Người ta có thể tạo ra hàng ngàn mẩu tin giống nhau và không thể phân biệt được nó với bản gốc. Với các tài liệu lưu trữ và vận chuyển trên giấy điều này khó khăn hơn nhiều. Và điều cần thiết đối với một xã hội mà thông tin hầu hết được lưu trữ và vận chuyển trên các phương tiện điện tử chính là các phương tiện đảm bảo an toàn bảo mật thông tin độc lập với các phương tiện lưu trữ và vận chuyển vật lý của nó. Phương tiện đó chính là mật mã học, một ngành khoa học có lịch sử lâu đời dựa trên nền tảng các thuật toán toán học, số học, xác suất và các môn khoa học khác.

2. Khái niệm hệ thống và tài sản của hệ thống

Khái niệm hệ thống: Hệ thống là một tập hợp các máy tính gồm các thành phần phấn cứng, phần mềm và dữ liệu làm việc được tích luỹ qua thời gian.

Tài sản của hệ thống bao gồm:

- Phần cứng
- Phần mềm
- Dữ liệu
- Các truyền thông giữa các máy tính của hệ thống
- Môi trường làm việc
- Con người

3. Các mối đe doạ đối với một hệ thống và các biện pháp ngăn chặn

Có 3 hình thức chủ yếu đe dọa đối với hệ thống:

- Phá hoại: kẻ thù phá hỏng thiết bị phần cứng hoặc phần mềm hoạt động trên hệ thống.
- Sửa đổi: Tài sản của hệ thống bị sửa đổi trái phép. Điều này thường làm cho hệ thống không làm đúng chức năng của nó. Chẳng hạn như thay đổi mật khẩu, quyền người dùng trong hệ thống làm họ không thể truy cập vào hệ thống để làm việc.
- Can thiệp: Tài sản bị truy cập bởi những người không có thẩm quyền. Các truyền thông thực hiện trên hệ thống bị ngăn chặn, sửa đổi.

Các đe dọa đối với một hệ thống thông tin có thể đến từ nhiều nguồn và được thực hiện bởi các đối tượng khác nhau. Chúng ta có thể chia thành 3 loại đối tượng như sau: các đối tượng từ ngay bên trong hệ thống (insider), đây là những người có quyền truy cập hợp pháp đối với hệ thống, những đối tượng bên ngoài hệ thống (hacker, cracker), thường các đối tượng này tấn công qua những đường kết nối với hệ thống như Internet chẳng hạn, và thứ ba là các phần mềm (chẳng hạn như spyware, adware …) chạy trên hệ thống.

Các biện pháp ngăn chặn:

Thường có 3 biện pháp ngăn chặn:

- Điều khiển thông qua phần mềm: dựa vào các cơ chế an toàn bảo mật của hệ thống nền (hệ điều hành), các thuật toán mật mã học
- Điều khiển thông qua phần cứng: các cơ chế bảo mật, các thuật toán mật mã học được cứng hóa để sử dụng
- Điều khiển thông qua các chính sách của tổ chức: ban hành các qui định của tổ chức nhằm đảm bảo tính an toàn bảo mật của hệ thống.

Trong môn học này chúng ta tập trung xem xét các thuật toán mật mã học như là một phương tiện cơ bản, chủ yếu để đảm bảo an toàn cho hệ thống.

4. Mục tiêu và nguyên tắc chung của an toàn bảo mật thông tin

Ba mục tiêu của an toàn bảo mật thông tin:

– Tính bí mật: Tài sản của hệ thống chỉ được truy cập bởi những người có thẩm quyền. Các loại truy cập gồm có: đọc (reading), xem (viewing), in ấn (printing), sử dụng chương trình, hoặc hiểu biết về sự tồn tại của một đối tượng trong tổ chức.Tính bí mật có thể được bảo vệ nhờ việc kiểm soát truy cập (theo nhiều kiểu khác nhau) hoặc nhờ các thuật toán mã hóa dữ liệu. Kiểm soát truy cập chỉ có thể được thực hiện với các hệ thống phần cứng vật lý. Còn đối với các dữ liệu công cộng thì thường phương pháp hiệu quả là các phương pháp của mật mã học.

– Tính toàn vẹn dữ liệu: tài sản của hệ thống chỉ được thay đổi bởi những người có thẩm quyền.

– Tính sẵn dùng: tài sản luôn sẵn sàng được sử dụng bởi những người có thẩm quyền.

Hai nguyên tắc của an toàn bảo mật thông tin:

– Việc thẩm định về bảo mật phải là khó và cần tính tới tất cả các tình huống, khả năng tấn công có thể được thực hiện.

– Tài sản được bảo vệ cho tới khi hết gía trị sử dụng hoặc hết ý nghĩa bí mật.

5. Mật mã học (cryptology)

Mật mã học bao gồm hai lĩnh vực: mã hóa (cryptography) và thám mã (cryptanalysis-codebreaking) trong đó:

– Mã hóa: nghiên cứu các thuật toán và phương thức để đảm bảo tính bí mật và xác thực của thông tin (thường là dưới dạng các văn bản lưu trữ trên máy tính). Các sản phẩm của lĩnh vực này là các hệ mã mật, các hàm băm, các hệ chữ ký điện tử, các cơ chế phân phối, quản lý khóa và các giao thức mật mã.

– Thám mã: Nghiên cứu các phương pháp phá mã hoặc tạo mã giả. Sản phẩm của lĩnh vực này là các phương pháp thám mã, các phương pháp giả mạo chữ ký, các phương pháp tấn công các hàm băm và các giao thức mật mã.

Trong giới hạn của môn học này chúng ta chủ yếu tập trung vào tìm hiểu các vấn đề mã hóa với các hệ mã mật, các hàm băm, các hệ chữ ký điện tử, các giao thức mật mã.

Mã hóa (cryptography) là một ngành khoa học của các phương pháp truyền tin bảo mật. Trong tiếng Hy Lạp, "Crypto" (krypte) có nghĩa là che dấu hay đảo lộn, còn "Graphy" (grafik) có nghĩa là từ. [3]

Người ta quan niệm rằng: những từ, những ký tự của bản văn bản gốc có thể hiểu được sẽ cấu thành nên bản rõ (P-Plaintext), thường thì đây là các đoạn văn bản trong một ngôn ngữ nào đó; còn những từ, những ký tự ở dạng bí mật không thể hiểu được thì được gọi là bản mã (C-Ciphertext).

Có 2 phương thức mã hoá cơ bản: thay thế và hoán vị:

– Phương thức mã hoá thay thế là phương thức mã hoá mà từng ký tự gốc hay một nhóm ký tự gốc của bản rõ được thay thế bởi các từ, các ký hiệu khác hay kết hợp với nhau cho phù hợp với một phương thức nhất định và khoá.

– Phương thức mã hoá hoán vị là phương thức mã hoá mà các từ mã của bản rõ được sắp xếp lại theo một phương thức nhất định.

Các hệ mã mật thường sử dụng kết hợp cả hai kỹ thuật này.

6. Khái niệm hệ mã mật (CryptoSystem)

Một hệ mã mật là bộ 5 ($\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D}$) thoả mãn các điều kiện sau:

1) $\mathcal{P}$ là không gian bản rõ: là tập hữu hạn các bản rõ có thể có.

2) $\mathcal{C}$ là không gian bản mã: là tập hữu hạn các bản mã có thể có.

3) $\mathcal{K}$ là kkhông gian khoá: là tập hữu hạn các khoá có thể có.

4) Đối với mỗi $k \in \mathcal{K}$, có một quy tắc mã hoá $e_k \in \mathcal{E}$ và một quy tắc giải mã tương ứng $d_k \in \mathcal{D}$. Với mỗi e_k: $\mathcal{P} \to \mathcal{C}$ và d_k: $\mathcal{C} \to \mathcal{P}$ là những hàm mà $d_k(e_k(x)) = x$ cho mọi bản rõ $x \in \mathcal{P}$. Hàm giải mã d_k chính là ánh xạ ngược của hàm mã hóa e_k [5]

Thường thì không gian các bản rõ và không gian các bản mã là các văn bản được tạo thành từ một bộ chữ cái $\mathcal{A}$ nào đó. Đó có thể là bộ chữ cái tiếng Anh, bộ mã ASCII, bộ mã Unicode hoặc đơn giản nhất là các bit 0 và 1.

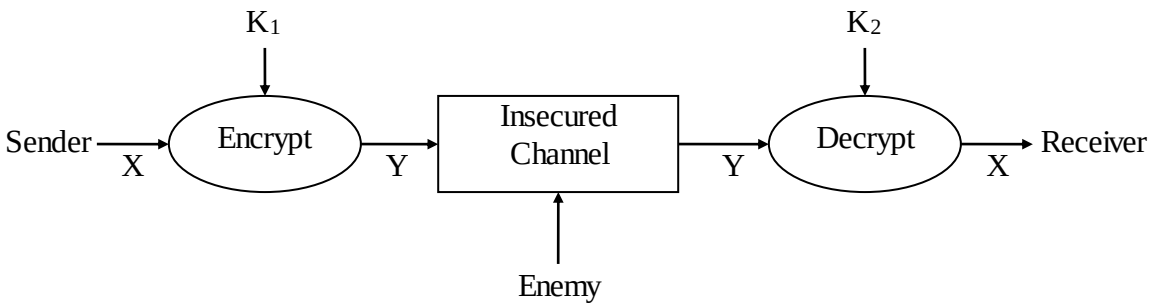
Tính chất 4 là tính chất quan trọng nhất của mã hoá. Nội dung của nó nói rằng nếu mã hoá bằng e_k và bản mã nhận được sau đó được giải mã bằng hàm d_k thì kết quả nhận được phải là bản rõ ban đầu x. Rõ ràng trong trường hợp này, hàm $e_k(x)$ phải là một đơn ánh, nếu không thì ta sẽ không giải mã được. Vì nếu tồn tại x_1 và x_2 sao cho $y = e_k(x_1) = e_k(x_2)$ thì khi nhận được bản mã y ta không biết nó được mã từ x_1 hay x_2.

Trong một hệ mật bất kỳ ta luôn có $|\mathcal{C}| \geq |\mathcal{P}|$ vì mỗi quy tắc mã hoá là một đơn ánh. Khi $|\mathcal{C}| = |\mathcal{P}|$ thì mỗi hàm mã hoá là một hoán vị.

7. Mô hình truyền tin cơ bản của mật mã học và luật Kirchoff

Mô hình truyền tin thông thường: Trong mô hình truyền tin thông thường thông tin được truyền (vận chuyển) từ người gửi đến người nhận được thực hiện nhờ một kênh vật lý (chẳng hạn như việc gửi thư) được coi là an toàn.

Mô hình truyền tin cơ bản của mật mã học:



Hình 1.1: Mô hình cơ bản của truyền tin bảo mật

Đây là mô hình cơ bản của truyền tin bảo mật. Khác với truyền tin thông thường, có các yếu tố mới được thêm vào như khái niệm kẻ địch (E-Enemy), các khoá mã hoá và giải mã K để đảm bảo tính bảo mật của thông tin cần truyền đi.

Trong mô hình này người gửi S (Sender) muốn gửi một thông điệp X (Message – là một bản rõ) tới người nhận R (Receiver) qua một kênh truyền không an toàn (Insecured Channel), kẻ địch E (Enemy) có thể nghe trộm, hay sửa đổi thông tin X. Vì vậy, S sử dụng phép biến đổi, tức mã hoá (E-Encryption) lên thông tin X ở dạng đọc được (Plaintext) để tạo ra một đoạn văn bản được mã hoá Y (C-Ciphertext) không thể hiểu được theo một quy luật thông thường sử dụng một thông tin bí mật được gọi là khoá K_1 (Key), khoá K_1 chính là thông số điều khiển cho phép biến đổi từ bản rõ X sang bản mã Y (chỉ các bên tham gia truyền tin S và R mới có thể biết khóa này). Giải mã (D-Decryption) là quá trình ngược lại cho phép người nhận thu được thông tin X ban đầu từ đoạn mã hoá Y sử dụng khóa giải mã K_2 (chú ý là khóa giải mã và khóa mã hóa có thể khác nhau hoặc là một tùy thuộc vào hệ mã sử dụng).

Các phép biến đổi được sử dụng trong mô hình truyền tin trên thuộc về một hệ mã mật (Cryptosytem) nào đó.

Quá trình mã hóa và giải mã yêu cầu các quá trình biến đổi dữ liệu từ dạng nguyên thuỷ thành in put cho việc mã hóa và chuyển output của quá trình giải mã thành bản rõ. Các quá trình này là các quá trình biến đổi không khóa và được gọi là các quá trình encode và decode.

Theo luật Kirchoff (1835 - 1903) (một nguyên tắc cơ bản trong mã hoá) thì: *toàn bộ cơ chế mã/giải mã trừ khoá là không bí mật đối với kẻ địch* [5]. Rõ ràng khi đối phương không biết được hệ mã mật đang sử dụng thuật toán mã hóa gì thì việc thám mã sẽ rất khó khăn. Nhưng chúng ta không thể tin vào độ an toàn của hệ mã mật chỉ dựa vào một giả thiết không chắc chắn là đối phương không biết thuật toán đang sử dụng. Vì vậy, khi trình bày một hệ mật bất kỳ, chúng ta đều giả thiết hệ mật đó được trình bày dưới luật Kirchoff.

Ý nghĩa của luật Kirchoff: sự an toàn của các hệ mã mật không phải dựa vào sự phức tạp của thuật toán mã hóa sử dụng.

8. Sơ lược về lịch sử mật mã học

Mật mã học là một ngành khoa học có một lịch sử khoảng 4000 năm. Các cổ vật của ngành khảo cổ học thu được đã cho thấy điều này. Những người Ai cập cổ đại đã sử dụng các chữ tượng hình như là một dạng mã hóa đơn giản nhất trên các bia mộ của họ. Các tài liệu viết tay khác cũng cho thấy các phương pháp mã hóa đơn giản đầu tiên mà loài người đã sử dụng là của người Ba Tư cổ và người Do Thái cổ.

Tuy vậy có thể chia lịch sử mật mã học thành hai thời kỳ như sau:

Thời kỳ tiền khoa học: Từ trước công nguyên cho tới năm 1949. Trong giai đoạn này mật mã học được coi là một nghệ thuật nhiều hơn là một môn khoa học mặc dù đã được ứng dụng trong thực tế.

Lịch sử của mật mã học được đánh dấu vào năm 1949 khi Claude Shannon đưa ra lý thuyết thông tin. Sau thời kỳ này một loạt các nghiên cứu quan trọng của nghành mật mã học đã được thực hiện chẳng hạn như các nghiên cứu về mã khối, sự ra đời của các hệ mã mật khóa công khai và chữ ký điện tử.

Qua nhiều thế kỷ phát triển của mật mã học chủ yếu được phục vụ cho các mục đích quân sự (gián điệp, ngoại giao, chiến tranh…). Một ví dụ điển hình là 2000 năm trước đây hoàng đế La mã Julius Caesar đã từng sử dụng một thuật toán thay thế đơn giản mà ngày nay được mang tên ông trong cuộc chiến tranh Gallic.

Tác phẩm “A manuscript on Deciphering Cryptography Messages” của Abu al-Kindi được viết vào thế kỷ thứ 9 được tìm thấy tại Istabul vào năm 1987 đã cho thấy những nhà khoa học Ả rập là những người đầu tiên đã phát triển các phương pháp thám mã dựa vào phân tích tần số xuất hiện của các ký tự đối với các hệ mã thay thế đơn âm (một phương pháp được sử dụng rộng rãi trong thời kỳ Trung cổ do đơn giản và khá hiệu quả).

Ở châu Âu thời kỳ Trung cổ là một khoảng thời gian u ám và tăm tối của lịch sử nên không có nhiều phát triển mạnh về văn hóa nói chung và mật mã học nói riêng. Một vài sự kiện được ghi lại bởi các vị linh mục nhưng chỉ có Roger Bacon là người thực sự đã viết về mật mã học trong tác phẩm “Secret Work of Art and the Nullity of Magic” vào giữa những năm 1200. Vào thời Trung cổ một trong những cái tên nổi tiếng nhất là Chaucer, người đã đưa ra các công trình nghiên cứu nghiêm túc đầu tiên về mật mã học trong các

tác phẩm của mình chẳng hạn như "Treatise on the Astrolabe". Trong thời kỳ Trung cổ ở phương Tây cuốn sách của Blaise De Vigenere (người phát minh ra thuật toán mã hóa thay thế đa âm tiết) được xem như là một tổng kết các kiến thức về mật mã học cho tới thời điểm bấy giờ, bao gồm cả thuật toán thay thế đa âm tiết và một vài sơ đồ khóa tự động.

Blaise De Vigenere cũng là tác giả của hệ mã mang tên ông, hệ mã này đã từng được xem là an toàn tuyệt đối và được sử dụng trong một thời gian dài, tuy nhiên Charles Babbages đã thực hiện thám mã thành công vào năm 1854 nhưng điều này được giữ bí mật. Một thuật toán thám mã được phát hiện độc lập bởi một nhà khoa học người Phổ (thuộc nước Đức ngày nay) có tên là Friedrich Kasiski. Tuy vậy do việc thiếu các thiết bị cải tiến nên các biến thể của thuật toán mã hóa này vẫn còn được sử dụng trong những năm đầu của thế kỷ 20 mà tiêu biểu nhất là việc thám mã thành công máy điện tín Zimmermann của quân Đức (một trong các sự kiện tiêu biểu của mật mã học) trong thế chiến thứ nhất và kết quả là sự tham gia của Mỹ vào cuộc chiến.

Với sự xuất hiện của các hệ thống máy tính cá nhân và mạng máy tính các thông tin văn bản ngày càng được lưu trữ và xử lý nhiều hơn trên các máy tính do đó nảy sinh yêu cầu về an toàn bảo mật đối với các thông tin được lưu trữ, xử lý và truyền giữa các máy tính.

Vào đầu những năm 1970 là sự phát triển của các thuật toán mã hóa khối đầu tiên: Lucipher và DES. DES sau đó đã có một sự phát triển ứng dụng rực rỡ cho tới đầu những năm 90.

Vào cuối những năm 1970 chứng kiến sự phát triển của các thuật toán mã hóa khóa công khai sau khi Whitfield Diffie và Martin Hellman công bố bài báo "New Directions in Cryptography" làm nền tảng cho sự ra đời của các hệ mã khóa công khai và các hệ chữ ký điện tử.

Do nhược điểm của các hệ mã mật khóa công khai là chậm nên các hệ mã khối vẫn tiếp tục được phát triển với các hệ mã khối mới ra đời để thay thế cho DES vào cuối thế kỷ 20 như IDEA, AES hoặc 3DES (một cải tiến của DES).

Gần đây nhất là các sự kiện liên quan tới các hàm băm MD5 (một hàm băm thuộc họ MD do Ron Rivest phát triển) và SHA1. Một nhóm các nhà khoa học người Trung Quốc (Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu) đã phát triển các phương pháp cho phép phát hiện ra các đụng độ của các hàm băm được sử dụng rộng rãi nhất trong số các hàm băm này. Đây là một sự kiện lớn đối với ngành mật mã học do sự ứng dụng rộng rãi và có thể xem là còn quan trọng hơn bản thân các hệ mã mật của các hàm băm. Do sự kiện này các hãng viết phần mềm lớn (như Microsoft) và các nhà mật mã học đã khuyến cáo các lập trình viên sử dụng các hàm băm mạnh hơn (như SHA-256, SHA-512) trong các ứng dụng.

Bruce Schneier (một trong những nhà mật mã học hàng đầu, tác giả của hệ mã Blowfish) đã từng nói rằng các hình thức tấn công đối với các hệ mã mật nói riêng và tấn công đối với các hệ thống máy tính nói chung sẽ ngày càng trở nên hoàn thiện hơn "Attacks always get better; they never get worse." và lịch sử phát triển của mật mã học chính là lịch sử phát triển của các hình thức tấn công đối với các hệ mã mật đang được sử dụng.

9. Phân loại các thuật toán mật mã học

Có nhiều cách khác nhau để chúng ta có thể phân loại các thuật toán mật mã học sẽ được học trong chương trình. Ở đây chúng ta sẽ phân loại các thuật toán mật mã học dựa vào hai loại tiêu chí.

Tiêu chí thứ nhất là dựa vào các dịch vụ an toàn bảo mật mà các thuật toán cung cấp, dựa vào số lượng khóa sử dụng (0, 1, 2) chúng ta có các thuật toán mã hóa sau:

1. Các thuật toán mã hóa khóa bí mật tương ứng với các hệ mã mật khóa bí mật hay khóa đối xứng SKC (Symmetric Key Cryptosytems), do vai trò của người nhận và người gửi là như nhau, cả hai đều có thể mã hóa và giải mã thông điệp, như Caesar, DES, AES … Khóa sử dụng cho các thuật toán này là 1 khóa cho cả việc mã hóa và giải mã.

2. Các thuật toán mã hóa khóa công khai tương ứng với các hệ mã khóa công khai PKC (Public Key Cryptosystems). Đôi khi các hệ mã này còn được gọi là các hệ mã khóa bất đối xứng (Asymmetric Key Cryptosytems). Khóa sử dụng cho các thuật toán này là 2 khóa, một cho việc mã hóa và một cho việc giải mã, khóa mã hóa được công khai hóa.

3. Các thuật toán tạo chữ ký điện tử (Digital Signature Algorithms). Các thuật toán tạo chữ ký điện tử tạo thành các hệ chữ ký điện tử. Thông thường mỗi hệ chữ ký điện tử có cùng cơ sở lý thuyết với một hệ mã mật khóa công khai nhưng với cách áp dụng khác nhau. Trong chương trình học chúng ta sẽ học một số hệ chữ ký điện tử phổ biến là RSA, ElGammma…

4. Các hàm băm (Hash functions). Các hàm băm là các thuật toán mã hóa không khóa hoặc có khóa và thường được sử dụng trong các hệ chữ ký điện tử hoặc các hệ mã khóa công khai.

Tiêu chí thứ hai phân loại các thuật toán mã hóa dựa trên cách thức xử lý input của thuật toán (tức là bản rõ), dựa trên tiêu chí này chúng ta có hai loại thuật toán mã hóa sau:

1. Các thuật toán mã hóa khối (chẳng hạn như DES, AES …) xử lý bản rõ dưới các đơn vị cơ bản là các khối có kích thước giống nhau.

2. Các thuật toán mã hóa dòng (RC4 …) coi bản rõ là một luồng bit, byte liên tục.

10. Một số ứng dụng của mật mã học

Ngày nay khó có thể tìm thấy các ứng dụng trên máy tính lại không sử dụng tới các thuật toán và các giao thức mật mã học. Từ các ứng dụng cho các máy tính cá nhân (Desktop Applications) cho tới các chương trình hệ thống như các hệ điều hành (Operating Systems) hoặc các ứng dụng mạng như Yahoo Messenger hoặc các hệ cơ sở dữ liệu đều có sử dụng các thuật toán mã hóa mật khẩu người dùng bằng một hệ mã hoặc một hàm băm nào đó. Đặc biệt với sự phát triển mạnh mẽ của thương mại điện tử các mô hình chữ ký điện tử ngày càng đóng vai trò tích cực cho một môi trường an toàn cho người dùng. Tuy vậy chúng ta vẫn có thể chia các lĩnh vực ứng dụng của mật mã học thành các lĩnh vực nhỏ như sau:

– Bảo mật (Confidentiality): che dấu nội dung của các thông điệp được trao đổi trong một phiên truyền thông hoặc giao dịch hoặc các thông điệp trên một hệ thống máy tính (các file, các dữ liệu trong một cơ sở dữ liệu …).

– Xác thực hóa (Authentication): đảm bảo nguồn gốc của một thông điệp, người dùng.

– Toàn vẹn (Integrity): đảm bảo chỉ có các tổ chức đã được xác thực hóa mới có thể thay đổi các tài sản của hệ thống cũng như các thông tin trên đường truyền.

– Dịch vụ không thể chối từ (Non-Repudiation): Các bên đã được xác thực không thể phủ nhận việc tham gia vào một giao dịch hợp lệ.

– Ngoài ra còn các dịch vụ quan trọng khác chẳng hạn như chữ ký điện tử, dịch vụ chứng thực danh tính (Identification) cho phép thay thế hình thức xác thực hóa người dùng dựa trên các mật khẩu bằng các kỹ thuật mạnh hơn hoặc dịch vụ thương mại điện tử cho phép tiến hành các giao dịch an toàn trên các kênh truyền thông không an toàn như Internet.

CHƯƠNG II: CƠ SỞ TOÁN HỌC

Để hiểu được những thuật toán sử dụng trong các hệ mã mật, trong các hệ chữ ký điện tử cũng như các giao thức mật mã, chúng ta phải có những kiến thức nền tảng cơ bản về toán học, lý thuyết thông tin … được sử dụng trong mật mã học. Chương này trình bày những khái niệm cơ bản về lý thuyết thông tin như Entropy, tốc độ của ngôn ngữ (Rate of Language), độ phức tạp của thuật toán, độ an toàn của thuật toán, và một số kiến thức toán học: đồng dư số học (modulo), số nguyên tố, định lý phần dư trung hoa, định lý Fermat . . . và các thuật toán kiểm tra số nguyên tố. Những vấn đề chính sẽ được trình bày trong chương này gồm :

- **Lý thuyết thông tin**
- **Lý thuyết độ phức tạp**
- **Lý thuyết số học.**

1. Lý thuyết thông tin

Những khái niệm mở đầu của lý thuyết thông tin được đưa ra lần đầu tiên vào năm 1948 bởi Claude Elmwood Shannon (một nhà khoa học được coi là cha để của lý thuyết thông tin). Trong phần này chúng ta chỉ đề cập tới một số chủ đề quan trọng của lý thuyết thông tin.

1.1. Entropy

Lý thuyết thông tin định nghĩa khối lượng thông tin trong một thông báo là số bít nhỏ nhất cần thiết để mã hoá tất cả những nghĩa có thể của thông báo đó.

Ví dụ, trường ngay_thang trong một cơ sở dữ liệu chứa không quá 3 bít thông tin, bởi vì thông tin ngày có thể mã hoá với 3 bít dữ liệu:

000 = Sunday

001 = Monday

010 = Tuesday

011 = Wednesday

100 = Thursday

101 = Friday

110 = Saturday

111 is unused

Nếu thông tin này được biểu diễn bởi chuỗi ký tự ASCII tương ứng, nó sẽ chiếm nhiều không gian nhớ hơn, nhưng cũng không chứa nhiều thông tin hơn. Tương tự như trường gioi_tinh của một cơ sở dữ liệu chỉ chứa 1 bít thông tin, nó có thể lưu trữ như một trong hai xâu ký tự ASCII : Nam, Nữ.

Khối lượng thông tin trong một thông báo M đo bởi Entropy của thông báo đó, ký hiệu là H(M). Entropy của thông báo gioi_tinh là 1 bít, ký hiệu H(gioi_tinh) = 1, Entropy của thông báo số ngày trong tuần là nhỏ hơn 3 bits.

Trong trường hợp tổng quát, **Entropy** của một thông báo là $\log_2 n$, với n là số khả năng có thể (ý nghĩa) của thông báo.

$$H(M) = \log_2 n$$

1.2. Tốc độ của ngôn ngữ. (Rate of Language)

Đối với một ngôn ngữ, tốc độ thực tế (actual rate) của ngôn ngữ là:

$$r = H(M)/N$$

trong trường hợp này N là độ dài của thông báo và M là một thông điệp có độ dài N. Tốc độ của tiếng Anh bình thường là 0.28 do đó mỗi chữ cái tiếng Anh có 1.3 bit nghĩa.

Tốc độ tuyệt đối (absolute rate) của một ngôn ngữ là số bits lớn nhất cần thiết để mã hóa các ký tự của ngôn ngữ đó. Nếu có L ký tự trong một ngôn ngữ, thì tốc độ tuyệt đối là :

$$R = \log_2 L$$

Đây là số Entropy lớn nhất của mỗi ký tự đơn lẻ. Đối với tiếng Anh gồm 26 chữ cái, tốc độ tuyệt đối là $\log_2 26 = 4.7$bits/chữ cái. Sẽ không có điều gì là ngạc nhiên đối với tất cả mọi người rằng thực tế tốc độ của tiếng Anh nhỏ hơn nhiều so với tốc độ tuyệt đối, và chúng ta vẫn thấy rằng đối với một thông báo bằng tiếng Anh có thể loại bỏ một số chữ cái nhưng người đọc vẫn có thể hiểu được. Hiện tượng này được gọi là **độ dư thừa của ngôn ngữ** (Redundancy) tự nhiên.

Không chỉ đối với tiếng Anh mà với hầu hết các ngôn ngữ tự nhiên, do cấu trúc của ngôn ngữ, do việc sử dụng ngôn ngữ dẫn tới có một số chữ cái được sử dụng với tần suất không đồng đều hoặc chỉ có thể xuất hiện với một cấu trúc nào đó làm cho chúng ta vẫn có thể đoán được nghĩa của các thông báo nếu loại bỏ các chữ cái này.

Độ dư thừa **(Redundancy)** của một ngôn ngữ ký hiệu là D và D = R – r. Đối với tiếng Anh:

D = 1 - .28 = .72 letters/letter

D = 4.7 – 1.3 = 3.4 bits/letter

Như vậy mỗi chữ cái có 1.3 bit nghĩa và 3.4 bit dư thừa (xấp xỉ 72%).

1.3. Tính an toàn của hệ thống mã hoá

Shannon định nghĩa rất rõ ràng, tỉ mỉ các mô hình toán học để đánh giá độ an toàn của các hệ mã mật sử dụng. Mục đích của người thám mã là phát hiện ra khoá sử dụng của hệ mã **(K-Key)**, bản rõ (**P-PlainText**), hoặc cả hai. Hơn nữa họ có thể hài lòng với một vài thông tin có khả năng về bản rõ **P** chẳng hạn như đó là âm thanh dạng số, hoặc là một văn bản tiếng Đức, hoặc là một bảng tính dữ liệu, v. v . . .

Trong hầu hết các lần thám mã, người thám mã thường cố gắng thu thập một số thông tin có khả năng về bản rõ **P** trước khi bắt đầu. Họ có thể biết ngôn ngữ đã được sử dụng để mã hoá. Ngôn ngữ này chắc chắn có sự dư thừa kết hợp với chính ngôn ngữ đó. Nếu nó là một thông báo gửi tới **Bob**, nó có thể bắt đầu với "Dear Bob". Đoạn văn bản

"Dear Bob" sẽ là một khả năng có thể hơn là một chuỗi không mang ý nghĩa gì chẳng hạn "tm*h&rf". Mục đích của việc thám mã là sửa những tập hợp khả năng có thể có của bản mã (**C-CipherText**) với mỗi khả năng có thể của bản rõ.

Shannon phát triển lý thuyết cho rằng, hệ thống mã hoá chỉ an toàn tuyệt đối nếu nếu số khoá có thể sử dụng ít nhất phải bằng số thông báo có thể. Hiểu theo một nghĩa khác, khoá tối thiểu của hệ mã phải dài bằng thông báo của hệ mã đó.

Ngoại trừ các hệ mã an toàn tuyệt đối, các bản mã thường chứa một số thông tin đúng với bản rõ, điều này là không thể tránh được. Một thuật toán mật mã tốt giữ cho thông tin bị tiết lộ ở mức nhỏ nhất và một người thám mã giỏi sẽ khai thác tốt những thông tin này để phát hiện ra bản rõ.

Người thám mã sử dụng sự dư thừa tự nhiên của ngôn ngữ để làm giảm số khả năng có thể có của bản rõ. Nhiều thông tin dư thừa của ngôn ngữ, sẽ dễ dàng hơn cho quá trình thám mã. Chính vì lý do này mà nhiều mô hình mã hóa sử dụng thuật toán nén bản rõ để giảm kích thước văn bản trước khi mã hoá chúng. Vì quá trình nén làm giảm sự dư thừa của thông báo. Entropy của một hệ mã mật là kích thước của không gian khoá (**Keyspace**).

$$\mathbf{H(K) = \log_2(number\ of\ keys)}$$

Shannon cũng đưa ra một khái niệm gọi là Unicity Distance (ký hiệu là U) để đánh giá độ an toàn của một hệ mã mật. Đối với một hệ mã mật U của nó là:

$$\mathbf{U = H(K)/D}$$

Đây là số nhỏ nhất các bản mã cần thiết để có thể tiến hành thám mã theo cách thử tất cả các khóa có thể (brute-force attack) thành công. Chẳng hạn đối với hệ mã thay thế đơn âm (như Caesar) trên bảng chữ cái tiếng Anh ta sẽ có:

$H(K) = \log_2 26! = 87$. $D = 3.4$ suy ra $U = 25.5$.

Điều này có nghĩa là nếu chúng ta có khoảng 25 chữ cái bản mã chúng ta chỉ có thể thử để khớp với một bản rõ.

Khái niệm Unicity Distance là một khái niệm mang tính xác suất nó cho chúng ta biết số lượng ít nhất các bản mã cần có để có thể xác định duy nhất 1 bản mã chứ không phải là số bản mã đủ để tiến hành thám mã (chắc chắn thành công). Nếu chúng ta có số bản mã ít hơn số U thì không thể nói là dự đoán (phép thử) của chúng ta là đúng. Dựa vào công thức này chúng ta thấy nếu như độ dư thừa của ngôn ngữ càng gần 0 thì càng khó thám mã mặc dù đó có thể là một hệ mã rất đơn giản. Cũng dựa vào công thức này suy ra để tăng tính an toàn của hệ mã có thể tăng không gian khóa của nó.

1.4. Kỹ thuật lộn xộn và rườm rà (Confusion and Diffusion)

Theo Shannon, có hai kỹ thuật cơ bản để che dấu sự dư thừa thông tin trong thông báo gốc, đó là: sự lộn xộn và sự rườm rà.

Kỹ thuật lộn xộn (Confusion): che dấu mối quan hệ giữa bản rõ và bản gốc. Kỹ thuật này làm thất bại các cố gắng nghiên cứu bản mã để tìm kiếm thông tin dư thừa và thống kê mẫu. Phương pháp dễ nhất để thực hiện điều này là thông qua **kỹ thuật thay thế**. Một hệ mã hoá thay thế đơn giản, chẳng hạn hệ mã dịch vòng Caesar, dựa trên nền

tảng của sự thay thế các chữ cái của bản rõ, nghĩa là chữ cái này được thay thế bằng chữ cái khác

Kỹ thuật rườm rà (Diffusion): làm mất đi sự dư thừa của bản rõ bằng cách tăng sự phụ bản mã vào bản rõ (và khóa). Công việc tìm kiếm sự dư thừa của người thám mã sẽ rất mất thời gian và phức tạp. Cách đơn giản nhất tạo ra sự rườm rà là thông qua việc đổi chỗ (hay còn gọi là **kỹ thuật hoán vị**).

Thông thường các hệ mã hiện đại thường kết hợp cả hai kỹ thuật thay thế và hoán vị để tạo ra các thuật toán mã hóa có độ an toàn cao hơn.

2. Lý thuyết độ phức tạp

Lý thuyết độ phức tạp cung cấp một phương pháp để phân tích độ phức tạp tính toán của thuật toán và các kỹ thuật mã hoá khác nhau. Nó so sánh các thuật toán mã hoá, kỹ thuật và phát hiện ra độ an toàn của các thuật toán đó. *Lý thuyết thông tin đã cho chúng ta biết rằng một thuật toán mã hoá có thể bị bại lộ. Còn lý thuyết độ phức tạp cho biết khả năng bị thám mã của một hệ mã mật.*

Độ phức tạp thời gian của thuật toán là một hàm của kích thước dữ liệu input của thuật toán đó. Thuật toán có độ phức tạp thời gian f(n) đối với mọi n và kích thước input n, nghĩa là số bước thực hiện của thuật toán lớn hơn f(n) bước.

Độ phức tạp thời gian thuật toán phụ thuộc vào mô hình của các thuật toán, số các bước nhỏ hơn nếu các hoạt động được tập trung trong một bước (chẳng hạn như các vòng lặp, các lời gọi hàm …).

Các lớp của thuật toán, với độ phức tạp thời gian là một hàm mũ đối với kích thước input được coi là "không có khả năng thực hiện". Các thuật toán có độ phức tạp giống nhau được phân loại vào trong các lớp tương đương. Ví dụ tất cả các thuật toán có độ phức tạp là n^3 được phân vào trong lớp n^3 và ký hiệu bởi $\boldsymbol{O}(n^3)$. Có hai lớp tổng quát sẽ được là lớp P (Polynomial) và lớp NP (NonPolynomial).

Các thuật toán thuộc lớp P có độ phức tạp là hàm đa thức của kích thước input. Nếu mỗi bước tiếp theo của thuật toán là duy nhất thì thuật toán gọi là đơn định. Tất cả thuật toán thuộc lớp P đơn định có thời gian giới hạn là P_time, điều này cho biết chúng sẽ thực hiện trong thời gian đa thức, tương đương với độ phức tạp đa thức của kích thước input.

Thuật toán mà ở bước tiếp theo việc tính toán phải lựa chọn giải pháp từ những giới hạn giá trị của hoạt động gọi là không đơn định. Lý thuyết độ phức tạp sử dụng các máy đặc biệt mô tả đặc điểm bằng cách đưa ra kết luận bởi các chuẩn. **Máy Turing** là một máy đặc biệt, máy hoạt động trong thời gian rời rạc, tại một thời điểm nó nằm trong khoảng trạng thái đầy đủ số của tất cả các trạng thái có thể là hữu hạn. Chúng ta có thể định nghĩa hàm độ phức tạp thời gian kết hợp với máy Turing A.

$$\mathbf{f_A(n) = max\{m/A \text{ kết thúc sau } m \text{ bước với đầu vào } w = n^3\}}$$

Ở đây chúng ta giả sử rằng A là trạng thái kết thúc đối với tất cả các đầu vào, vấn đề sẽ trở nên khó khăn hơn nếu các trạng thái không nằm trong P . Máy Turing không đơn định hoạt động với thuật toán NP. Máy Turing không đơn định có thể có một vài trạng

thái chính xác. S(w) là trạng thái đo sự thành công ngắn nhất của thuật toán, (Nghĩa là sự tính toán dẫn đến trạng thái cuối cùng)

Hàm số độ phức tạp thời gian của máy Turing không đơn định A được định nghĩa :

$$f_A(n)=max\{1,m/s(w) \text{ có m bước đối với } w/w=n\}$$

ở mỗi bước máy Turing không đơn định bố trí nhiều bản sao của chính nó như có một vài giải pháp và tính toán độc lập với mọi lời giải.

Các thuật toán thuộc lớp NP là không đơn định và có thể tính toán trên máy Turing không đơn định trong thời gian P.

Tuy nhiên không phải thuật toán mã hóa càng có độ phức tạp lớn thì hệ mã mật sử dụng thuật toán đó sẽ càng an toàn theo như phát biểu của luật Kierchoff.

Vậy có thể đánh giá độ an toàn của một hệ mã mật như thế nào? Vấn đề này đã được Claude Shannon trả lời với các khái niệm về độ an toàn của các hệ mã mật trong một bài báo có tiêu đề "Lý thuyết thông tin của các hệ thống bảo mật" (1949).

2.1. Độ an toàn tính toán

Định nghĩa:

Một hệ mật được gọi là an toàn về mặt tính toán nếu có một thuật toán tốt nhất để phá nó thì cần ít nhất N phép toán, với N là một số rất lớn nào đó. [10]

Tuy nhiên trong thực tế, không có một hệ mật nào chứng tỏ là an toàn theo định nghĩa trên. Vì vậy, trên thực tế, người ta gọi hệ mật là "an toàn tính toán" nếu có một thuật toán để phá nó nhưng đòi hỏi thời gian lớn đến mức không chấp nhận được (thuật toán có độ phức tạp hàm mũ hoặc thuộc lớp các bài toán có độ phức tạp NP).

Một cách tiếp cận khác về độ "an toàn tính toán" là quy nó về một bài toán đã được nghiên cứu kỹ và được coi là khó. Ví dụ như bài toán "phân tích ra thừa số nguyên tố của một số n cho trước" được coi là bài toán khó với n lớn, vì vậy ta có thể coi một hệ mật dựa trên bài toán "phân tích ra thừa số nguyên tố" là an toàn (tất nhiên đây chỉ là độ an toàn dựa vào chứng minh một bài toán khác chứ không phải chứng minh hoàn chỉnh về độ an toàn của hệ mật).

2.2. Độ an toàn không điều kiện

Định nghĩa 1:

Một hệ mật được coi là an toàn không điều kiện khi nó không thể bị phá ngay cả với khả năng tính toán không hạn chế. [10]

Rõ ràng là "độ an toàn không điều kiện" không thể nghiên cứu theo quan điểm độ phức tạp tính toán vì thời gian tính toán là không hạn chế. Vì vậy, ở đây lý thuyết xác suất sẽ được đề cập để nghiên cứu về "an toàn không điều kiện".

Định nghĩa 2:

Giả sử biến X và Y là các biến ngẫu nhiên. Ký hiệu xác suất để X nhận giá trị x là p(x) và để Y nhận giá trị y là p(y). Xác suất đồng thời p(x, y) là xác suất để đồng thời X nhận giá trị x và Y nhận giá trị y. Xác suất có điều kiện p(x/y) là xác suất để X nhận giá trị

x với điều kiện Y nhận giá trị y. Các biến X và Y được gọi là độc lập nếu p(x, y) = p(x)p(y) với mọi giá trị có thể có của X và Y.

Định lý Bayes:

Nếu p(y) ≠ 0 thì ta có:

$$p(x/y) = \frac{p(x)p(y/x)}{p(y)}$$

Hệ quả:

X, Y là biến độc lập khi và chỉ khi p(x/y) = p(x) với mọi x, y. [5]

Ở đây, ta giả thiết rằng một khoá cụ thể chỉ được dùng cho một bản mã. Ký hiệu xác suất tiên nghiệm để bản rõ xuất hiện là $p_P(x)$. Cũng giả thiết rằng khoá K được chọn theo một phân bố xác suất nào đó (thông thường khoá K được chọn ngẫu nhiên nên các khoá sẽ đồng khả năng). Ký hiệu xác suất khoá K được chọn là $p_K(K)$.

Giả thiết rằng khoá K và bản rõ x là các biến độc lập. Hai phân bố xác suất trên P và K sẽ tạo ra một phân bố xác suất trên C . Ký hiệu C(K) là tập các bản mã có thể nếu K là khoá.

$$C\,(K) = \{ e_K(x): x \in P \}$$

Khi đó với mỗi $y \in C$, ta có:

$$p_C(y) = \sum_{K, y \in C(K)} p_K(K).p_p(d_K(y))$$

Và xác suất có điều kiện $p_C(y/x)$ là xác suất để y là bản mã với điều kiện bản rõ là x được tính theo công thức sau:

$$p_C(y/x) = \sum_{K, x=d_K(y)} p_K(K)$$

Bây giờ ta có thể tính xác suất có điều kiện $p_P(x/y)$ là xác suất để x là bản rõ khi bản mã là y theo định lý Bayes:

$$p_P(x/y) = \frac{p_P(x)p_C(y/x)}{p_C(y)} = \frac{p_P(x)\sum_{K, x=d_K(y)} p_K(K)}{\sum_{K, y \in C(K)} p_K(K)p_P(d_K(y))}$$

Lúc này, ta có thể định nghĩa khái niệm về độ mật hoàn thiện. Nói một cách không hình thức, độ mật hoàn thiện nghĩa là đối phương với bản mã trong tay cũng không thể thu nhận được thông tin gì về bản rõ. Tuy nhiên ta sẽ nêu định nghĩa chính xác về độ mật hoàn thiện như sau:

Định nghĩa:

Một hệ mật hoàn thiện nếu $p_P(x/y) = p_P(x)$ với mọi $x \in P$ và mọi $y \in C$. Tức là xác suất hậu nghiệm để thu được bản rõ là x với điều kiện đã thu được bản mã là y đồng nhất với xác suất tiên nghiệm để bản rõ là x. [5]

Hay nói cách khác, độ mật hoàn thiện cũng tương đương với $p_C(y/x)= p_C(y)$).

Định lý Shannon:

Giả sử (P, C, K, E, D) là một hệ mật, khi đó hệ mật đạt được độ mật hoàn thiện khi và chỉ khi |K| ≥ |C|. Trong trường hợp |K| = |C| = |P|, hệ mật đạt độ mật hoàn thiện khi và chỉ khi mỗi khoá K được dùng với xác suất bằng nhau, bằng 1/|K| và với mỗi x ∈ P, mỗi y ∈ C có một khoá K duy nhất sao cho eK(x) = y. [5]

Như vậy ta thấy để đạt độ hoàn thiện đòi hỏi khoá phải rất dài, do vậy rất khó khăn trong việc chuyển giao khoá giữa hai bên truyền tin. Vì vậy trong thực tế, chúng ta không thể có an toàn không điều kiện mà chúng ta chỉ cần an toàn thực tế, tức là phụ thuộc vào thông tin và thời gian cần bảo mật bằng cách sử dụng các hệ mật khác nhau với độ bảo mật khác nhau.

3.3. Hệ mật tích

Một ý tưởng khác được Shannon đưa ra là ý tưởng tạo ra các hệ mật mới dựa trên các hệ mật cũ bằng cách tạo tích của chúng. Đây là một ý tưởng quan trọng trong việc thiết kế các hệ mật hiện đại ngày nay.

Để đơn giản, ở đây chúng ta chỉ xét các hệ mật trong đó C = P, các hệ mật loại này gọi là tự đồng cấu. Giả sử S1 = (P, C, K1, E1, D1) và S2 = (P, C, K2, E2, D2) là các hệ mật tự đồng cấu có cùng không gian bản rõ và bản mã. Khi đó hệ mật tích được định nghĩa là hệ mật S = (P, C, K1 × K2 ,E ,D). Khoá của hệ mật tích K = (K1, K2) trong đó K1 ∈ K1, K2 ∈ K2. Các hàm mã hoá và giải mã được xác định như sau:

$$e_{(K_1,K_2)}(x) = e_{K_2}(e_{K_1}(x))$$

$$d_{(K_1,K_2)}(x) = d_{K_1}(e_{K_2}(x))$$

Nếu chúng ta lấy tích của S với chính nó, ta có hệ mật (S×S) (ký hiệu S2). Nếu lấy tích n lần thì kết quả là Sn. Ta gọi Sn là một hệ mật lặp. Nếu S2 = S thì ta gọi hệ mật là luỹ đẳng. Nếu S là luỹ đẳng thì không nên lấy tích lặp vì độ bảo mật không tăng lên mà không gian khoá lại lớn hơn. Đương nhiên nếu S không luỹ đẳng thì ta có thể lặp lại S nhiều lần để tăng độ bảo mật. Ở đây nảy sinh một vấn đề là làm thế nào để có một hệ mật không luỹ đẳng?

Ta biết rằng nếu S1 và S2 là luỹ đẳng và giao hoán thì S1×S2 cũng luỹ đẳng, đơn giản vì:

$(S_1 \times S_2) \times (S_1 \times S_2) = S_1 \times (S_2 \times S_1) \times S_2$

$= S_1 \times (S_1 \times S_2) \times S_2$

$= (S_1 \times S_1) \times (S_2 \times S_2)$

$= (S_1 \times S_2)$

Vậy nếu muốn (S1×S2) không luỹ đẳng thì cần phải có S1 và S2 không giao hoán. Điều này có thể dễ dàng thực hiện bằng cách lấy tích của một hệ mật theo kiểu thay thế và một hệ mật theo kiểu hoán vị. Đây là kỹ thuật được dùng để thiết kế các hệ mã hiện đại như mã DES.

3. Lý thuyết toán học

3.1. Modulo số học

Về cơ bản $a \equiv b \pmod n$ nếu $a = b+kn$ trong đó k là một số nguyên. Nếu a và b dương và a nhỏ hơn n, chúng ta có thể gọi a là phần dư của b khi chia cho n. Nói chung a và b đều là phần dư khi chia cho n. Người ta còn gọ b là thặng dư của a theo modulo n, và a là đồng dư của b theo modulo n.

Modulo số học cũng giống như số học bình thường, bao gồm các phép giao hoán, kết hợp và phân phối. Mặt khác giảm mỗi giá trị trung gian trong suốt quá trình tính toán.

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

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

$(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n$

$(a \times (b + c)) \bmod n = (((a \times b) \bmod n) + ((a \times c) \bmod n)) \bmod n$

Các phép tính trong các hệ mã mật hầu hết đều thực hiện đối với một modulo N nào đó.

3.2. Số nguyên tố

Số nguyên tố là một số lớn hơn 1, nhưng chỉ chia hết cho 1 và chính nó, ngoài ra không còn số nào nó có thể chia hết nữa. Số 2 là một số nguyên tố đầu tiên và là số nguyên tố chẵn duy nhất. Do vậy 7, 17, 53, 73, 2521, 2365347734339 cũng là số nguyên tố. Số lượng số nguyên tố là vô tận. Hệ mật mã thường sử dụng số nguyên tố lớn cỡ 512 bits và thậm chí lớn hơn như vậy.

3.3. Ước số chung lớn nhất

Hai số a và n được gọi là hai số nguyên tố cùng nhau nếu chúng không có thừa số chung nào khác 1, hay nói một cách khác, nếu ước số chung lớn nhất của a và n là bằng 1. Chúng ta có thể viết như sau :

GCD(a,n)=1, (GCD-Greatest Common Divisor)

Số 15 và 28 là hai số nguyên tố cùng nhau, nhưng 15 và 27 thì không phải là hai số nguyên tố cùng nhau do có ước số chung là 1 và 3, dễ dàng thấy 13 và 500 cũng là một cặp số nguyên tố cùng nhau. Một số nguyên tố sẽ là nguyên tố cùng nhau với tất cả các số nguyên khác trừ các bội số của nó.

Một cách dễ nhất để tính toán ra ước số chung lớn nhất của hai số là nhờ vào thuật toán Euclid. Knuth mô tả thuật toán và một vài mô hình của thuật toán đã được sửa đổi.

Dưới đây là đoạn mã nguồn trong ngôn ngữ C:

```
/* Thuật toán tìm ước số chung lớn nhất của x và y, giả sử x,y>0 */
int  gcd(int x, int y)
{
  int  g;
  if(x<0)
```

```
            x=-x;
    if(y<0)
            y= -y;
    g=y;
    while(x>0){
            g=x;
            x=y%x;
            y=g;
    }
    return g;
}
```

3.4. Vành Z_N (vành đồng dư module N)

Tập các số nguyên Z_N = {0, 1, …, N-1} trong đó N là một số tự nhiên dương với hai phép toán cộng (+) và nhân (.) được định nghĩa như sau tạo thành một vành đồng dư modulo N (hay còn gọi là tập thặng dư đầy đủ theo modulo N):

Phép cộng:

$\forall$ a, b $\in Z_N$: a+b = (a+b) mod N.

Phép nhân:

$\forall$ a, b $\in Z_N$: a . b = (a * b) mod N.

Theo tính chất của modulo số học chúng ta dễ dàng nhận thấy Z_N là một vành giao hoán và kết hợp. Hầu hết các tính toán trong các hệ mã mật đều được thực hiện trên một vành Z_N nào đó.

Trên vành Z_N số 0 là phần tử trung hòa vì a + 0 = 0 + a = a, $\forall$ a $\in Z_N$, số 1 được gọi là phần tử đơn vị vì a . 1 = 1 . a = a $\forall$ a $\in Z_N$.

3.5. Phần tử nghịch đảo

Trên trường số thực R, số nghịch đảo của 5 là 1/5, bởi vì 5 $\times$ 1/5=1. Còn trên một vành số nguyên Z_N người ta đưa ra khái niệm về số nghịch đảo của một số như sau:

Giả sử a $\in Z_N$ và tồn tại b $\in Z_N$ sao cho a.b = (a*b) mod N = 1. Khi đó b được gọi là phần tử nghịch đảo của a trên Z_N và ký hiệu là a^{-1} = b.

Việc tìm phần tử nghịch đảo của một số a $\in Z_N$ cho trước thực chất tương đương với việc tìm hai số b và k sao cho: a.b = k.N + 1 trong đó b, k $\in Z_N$. Hay viết gọn lại là:

$$a^{-1} \equiv b \pmod N$$

Định lý về sự tồn tại của phần tử nghịch đảo: Nếu GCD(a, N) = 1 thì tồn tại duy nhất 1 số b $\in Z_N$ là phần tử nghịch đảo của a, nghĩa là thỏa mãn a.b = (a*b) mod N = 1.

3.6. Hàm phi Ơle

Với mỗi số nguyên N, giá trị của hàm phi Ơle của N là tổng số tất cả các số nguyên $\in Z_N$ và nguyên tố cùng nhau với N. Chẳng hạn nếu P là một số nguyên thì giá trị hàm phi Ơle của P: $\varnothing(P) = P - 1$ hoặc nếu N = p*q trong đó p và q là hai số nguyên tố thì $\varnothing(N) = (p-1)*(q-1)$.

Trong trường hợp tổng quát nếu dạng phân tích ra thừa số nguyên tố của N là:

$$N = p_1^{\alpha_1} p_2^{\alpha_2} ... p_k^{\alpha_k}$$

trong đó p_i là các số nguyên tố còn α_i là các số nguyên dương thì giá trị của hàm phi Ơle được tính như sau:

$$\phi(N) = (p_1 - 1) p_1^{\alpha_1 - 1} (p_2 - 1) p_2^{\alpha_2 - 1} ... (p_k - 1) p_k^{\alpha_k - 1}$$

Liên quan tới khái niệm về hàm phi Ơle chúng ta có định lý Ơle phát biểu như sau:

$\forall$ a $\in$ Z^*_N = Z_N – {0} và GCD(a, N) = 1 ta có $a^{\phi(N)} \equiv 1 (\mathrm{mod}\, N)$. Có nghĩa là $a^{\phi(N)}$ chính là giá trị nghịch đảo của a trên Z_N.

Một trường hợp riêng của định lý Ơle chính là định lý Fermat nhỏ: Nếu P là một số nguyên tố thì $\forall$ a $\in$ Z^*_P ta có $a^{P-1} \equiv 1 (\mathrm{mod}\, P)$. Đây là một trong những định lý đẹp nhất của số học.

Với mỗi số nguyên N vành Z^*_N gồm các phần tử thuộc Z_N và nguyên tố cùng nhau với N, hay nói cách khác: $Z^*_N = \{x: x \in Z_N, (x, N) = 1\} = \{x: x \in Z_N, x^{\phi(N)} = 1\}$.

Với mỗi phần tử a $\in$ Z_N, bậc t của a (ký hiệu là ord(a)) là số nhỏ nhất sao cho: $a^t = 1$. Theo định lý Ơle ta suy ra $\phi(N)$ chia hết cho t.

Cụ thể với N = 21 ta có bảng sau:

a$\in Z^*_{21}$	1	2	4	5	8	10	11	13	16	17	19	20
Ord(a)	1	6	3	6	2	6	6	2	3	6	6	2

Bảng 2.1: Bảng bậc của các phần tử trên Z^*_{21}

Nếu bậc của a $\in$ Z^*_N bằng $\phi(N)$ thì a được gọi là phần tử sinh hay phần tử nguyên thủy của tập Z^*_N. Và nếu tập Z^*_N chỉ có một phần tử sinh thì nó được gọi là một cyclic.

3.7. Thặng dư bậc hai

Giả sử a $\in$ Z^*_N, khi đó a được gọi là thặng dư bậc 2 theo modulo N nếu tồn tại x $\in$ Z^*_N sao cho $x^2 = a \pmod N$. Tập các phần tử thặng dư theo modulo N được ký hiệu là Q_N, tập các phần tử không thặng dư theo modulo N được gọi là bất thặng dư theo modulo N và ký hiệu là $\overline{Q}_N$.

Định lý: nếu p là một số nguyên tố lẻ và α là một phần tử sinh của Z^*_N, khi đó a là một thặng dư bậc 2 theo modulo N khi và chỉ khi $a = \alpha^i \bmod p$, trong đó i là số nguyên lẻ. Từ định lý này suy ra $\left|\overline{Q}_N\right| = (p-1)/2 = \left|Q_N\right|$.

Ví dụ với p = 13, $\alpha = 6 \in Z_{13}$ ta có bảng sau:

i	0	1	2	3	4	5	6	7	8	9	10	11
$\alpha^i \bmod 13$	1	6	10	8	9	2	12	7	3	5	4	11

Bảng 2.2: Bảng lũy thừa trên Z_{13}

Do đó $Q_{13} = \{1, 3, 4, 9, 10, 12\}$ và $\overline{Q}_{13} = \{2, 5, 6, 7, 8, 11\}$.

Với $a \in Q_N$. Nếu $x \in Z^*_N$ thỏa mãn $x^2 = a \pmod N$ thì a được gọi là căn bậc hai của x theo modulo N.

3.8. Thuật toán lũy thừa nhanh

Để có thể tìm phần tử nghịch đảo của một số nguyên a trên một vành Z_N cho trước chúng ta có thể sử dụng định lý Ơle để tính giá trị lũy thừa của a với số mũ là giá trị hàm phi Ơle của N. Tuy nhiên để có thể nhanh chóng tính được giá trị lũy thừa này chúng ta cần có một thuật toán hiệu quả và một trong các thuật toán đó (còn nhiều thuật toán khác phức tạp hơn) là thuật toán lũy thừa nhanh. Thuật toán này do Chivers đưa ra vào năm 1984. Các bước của thuật toán như sau:

Input: a, m, N.

Output: $a^m \bmod N$.

```
Begin
  Phân tích m thành dạng nhị phân m = bk bk-1 … b0.
  j = 0, kq = a;
  while (k>=j)
  {
        if (bj==1)
              kq = (kq * a) mod N;
        a = (a * a) mod N;
        j = j + 1;
  }
  return kq;
end
```

Một cài đặt khác bằng ngôn ngữ C như sau:

```
long modexp(long a, long x, long n)
{
```

```
        long r = 1;
        while (x > 0){
            if (x % 2 == 1) /* is x odd? */
                r = (r * a) % n;
            a = (a*a) % n;
            x /= 2;
        }
        return r;
}
```

Thuật toán này chạy không quá $\log_2(m+1)$ bước.

3.9. Thuật toán Ơclit mở rộng

Trong phần 3.3 chúng ta đã biết thuật toán Ơclit được dùng để tìm ước số chung lớn nhất của hai số nguyên và trong phần 3.7 chúng ta đã biết cách tìm một phần tử nghịch đảo của một số bằng cách sử dụng thuật toán lũy thừa nhanh tuy nhiên vẫn có một thuật toán hiệu quả khác để tìm phần tử nghịch đảo gọi là thuật tóan Ơclit mở rộng (do dựa trên thuật toán Ơclit). Các bước của thuật toán như sau:

input: a, N với i GCD(a, N) = 1

output: a^{-1}

begin

g_0=n, g_1 = a, u_0 = 1, u_1 = 0, v_0 = 0, v_1 = 1, i = 1;

while (g_i <>0)

{

$y = g_{i-1}$ div g_i;

$g_{i+1} = g_{i-1} – y*g_i$;

$u_{i+1} = u_{i-1} – y*u_i$;

$v_{i+1} = v_{i-1} – v*u_i$;

i = i + 1;

}

$x = v_{i-1}$;

if(x>0) then

return x;

else

return (N+x);

end;

3.10. Phương trình đồng dư bậc nhất 1 ẩn

Phương trình đồng dư bậc nhất 1 ẩn là phương trình có dạng:

$ax \equiv b \pmod N$ trong đó $a, b \in Z_N$ là các hệ số còn x là ẩn số.

Nếu như GCD(a, N) = 1 chúng ta có thể tìm a^{-1} sau đó nhân vào 2 vế của phương trình và tìm ra nghiệm một cách dễ dàng tuy nhiên nếu g = GCD(a, N) là một giá trị khác 1 thì sao? Khi đó bài toán có thể vô nghiệm hoặc có nhiều nghiệm. Chúng ta xét định lý sau:

Giả sử g = GCD(a, N) và nếu b chia hết cho g thì phương trình đồng dư bậc nhất 1 ẩn:

$ax \equiv b \pmod N$

sẽ có g nghiệm có dạng

$x \equiv ((b/g)x_0 + t(n/g)) \pmod N$ trong đó t = 0, …, g-1,

và x_0 là nghiệm của phương trình $(a/g)x \equiv 1 \pmod{N/g}$.

3.11. Định lý phần dư Trung Hoa.

Định lý phần dư Trung Hoa là một định lý quan trọng của số học được các nhà toán học Trung Quốc khám phá ra vào thế kỷ thứ nhất. Định lý phát biểu như sau:

Nếu $d_1, d_2, \ldots, d_k$ là các số nguyên đôi một nguyên tố cùng nhau và $N = d_1d_2\ldots d_k$ thì hệ phương trình đồng dư:

$x \equiv x_i \pmod{d_i}$, i=1, 2, …, k

sẽ có một nghiệm thuộc vào Z_N. Nghiệm của hệ có tính theo công thức sau:

$$x = \sum_{i=1}^{k} (N / d_i) y_i x_i \pmod N$$

trong đó y_i là các nghiệm của các phương trình đồng dư $(N/d_i)\, y_i \equiv 1 \pmod{d_i}$.

Dưới đây là đoạn mã định lý phần dư trung hoa trong ngôn ngữ C :

```
int chinese_remainder(int r, int *m, int *u)
{
  int i;
  int modulus;
  int n;
  modulus = 1;
  for ( i=0; i<r:++i )
        modulus *=m[i];
        n=0;
  for ( i=0; i<r:++i )
```

```
    {
        n+=u[i]*modexp(modulus/m[i],totient(m[i]),m[i]);
        n%=modulus;
    }
    return n;
}
```

4. Các thuật toán kiểm tra số nguyên tố.

Hàm **một phía (one-way functions)** là một khái niệm cơ bản của mã hoá công khai. Việc nhân hai số nguyên tố là một ví dụ về hàm một phía, nhân các số nguyên tố lớn để tạo thành một hợp số là dễ, nhưng công việc ngược lại phân tích một số nguyên lớn thành dạng thừa số nguyên tố lại là một bài toán khó (chưa có một thuật toán tốt).

Các thuật toán mã hoá khóa công khai đều cần phải sử dụng các số nguyên tố. Có một số phương pháp để sinh ra số nguyên tố và hầu hết chúng đều dựa trên các thuật toán kiểm tra tính nguyên tố của một số nguyên. Tuy nhiên có một số vấn đề được đặt ra đối với số nguyên tố như sau

- Trong một hệ thống có thể đảm bảo hai người dùng sẽ được sử dụng hai số nguyên tố khác nhau hay không? Câu trả lời là có thể vì có tới 10^{150} số nguyên tố có độ dài 512 bits hoặc nhỏ hơn.

- Khả năng hai người dùng sẽ lựa chọn cùng một số nguyên tố là bao nhiêu. Với sự lựa chọn từ 10^{150} số nguyên tố, điều kỳ xảy ra với xác xuất nhỏ hơn so với sự tự bốc cháy của máy tính.

Các loại thuật toán kiểm tra số nguyên tố được chia làm hai loại: thuật toán tất định và thuật toán xác suất. Các thuật toán tất định cho chúng ta biết chính xác câu trả lời một số nguyên có phải là một số nguyên tố hay không còn một thuật toán xác suất cho biết xác suất của một số nguyên là một số nguyên tố là bao nhiêu. Trong phần này sẽ trình bày một số thuật toán kiểm tra số nguyên tố phổ biến.

4.1. Một số ký hiệu toán học

4.1.1. Ký hiệu Lagrăng (Legendre Symbol)

Ký hiệu L(a,p) được định nghĩa với a là một số nguyên và p là một số nguyên tố lớn hơn 2. Nó nhận ba giá trị 0, 1, -1 :

L(a,p) = 0 nếu a chia hết cho p.

L(a,p) = 1 nếu a $\in Q_N$ (a là thặng dư bậc 2 modulo p).

L(a,p) = -1 nếu a $\in \overline{Q}_N$ (a không là thặng dư bậc 2 modulo p).

Một phương pháp dễ dàng để tính toán ra L(a,p) là :

L(a,p) = $a^{(p-1)/2} \bmod p$

4.1.2. Ký hiệu Jacobi (Jacobi Symbol)

Ký hiệu Jacobi được viết là J(a,n), nó là sự khái quát hoá của ký hiệu Lagrăng, nó định nghĩa cho bất kỳ cặp số nguyên a và n nào. Ký hiệu Jacobi là một chức năng trên tập hợp số thặng dư thấp của ước số n và có thể tính toán theo công thức sau:

- Nếu n là số nguyên tố, thì J(a,n) = 1 nếu a là thặng dư bậc hai modulo n .
- Nếu n là số nguyên tố, thì J(a,n) = -1 nếu a không là thặng dư bậc hai modulo n .
- Nếu n không phải là số nguyên tố thì Jacobi(a,n) sẽ được tính theo công thức sau:
- $J(a,n)=J(h,p_1) \times J(h,p_2) \times \ldots \times J(h,p_m)$

với $p_1,p_2 \ldots,p_m$ là các thừa số lớn nhất của n.

Thuật toán này tính ra số Jacobi tuần hoàn theo công thức sau :

1. $J(1,k) = 1$
2. $J(a \times b,k) = J(a,k) \times J(b,k)$
3. $J(2,k) =1$ Nếu $(k^2-1)/8$ là chia hết và $J(2,k) = -1$ trong các trường hợp khác.
4. $J(b,a) = J((b \bmod a),a)$
5. Nếu GCD(a,b)=1 :

a. $J(a,b) \times J(b,a) = 1$ nếu (a-1)(b-1)/4 là chia hết.

b. $J(a,b) \times J(b,a) = -1$ nếu (a-1)(b-1)/4 là còn dư.

Sau đây là thuật toán trong ngôn ngữ C :

```
int jacobi(int a,int b)
{
  int a1,a2;
  if(a>=b)
        a%=b;
  if(a==0)
        return 0;
  if(a==1)
        return 1;
  if(a==2)
        if(((b*b-1)/8)%2==0)
              return 1;
        else
              return -1;
```

```
if(a&b&1)  (cả a và b đều là số dư)
        if(((a-1)*(b-1)/4)%2==0)
                return +jacobi(b,a);
        else
                return -jacobi(b,a);
if(gcd(a,b)==1)
        if(((a-1)*(b-1)/4)%2==0)
        return +jacobi(b,a);
        else
                return -jacobi(b,a);
return jacobi(a1,b) * jacobi(a2,b);
}
```

Trên thực tế có thể tính được ký hiệu Jacobi một cách thuận lợi hơn nếu dựa vào 1 trong các tính chất sau, giả sử m, n là các số nguyên lẻ, a, b $\in$ Z:

(i) J(a*b, n) = J(a, n) * J(b, n) do đó $J(a^2, n) = 1$.

(ii) J(a, m*n) = J(a, m) * J(a, n).

(iii) nếu $a \equiv b \pmod n$ thì J(a, n) = J(b, n).

(iv) J(1, n) = 1.

(v) $J(-1, n) = (-1)^{(n-1)/2}$

(vi) $J(m, n) = J(n, m) * (-1)^{(m-1)*(n-1)/4}$

4.2. Thuật toán Soloway-Strassen

Soloway và Strassen đã phát triển thuật toán có thể kiểm tra số nguyên tố. Thuật toán này sử dụng hàm Jacobi.

Thuật toán kiểm tra số p là số nguyên tố:

1. Chọn ngẫu nhiên một số a nhỏ hơn p.
2. Nếu ước số chung lớn nhất gcd(a,p) $\neq$ 1 thì p là hợp số.
3. Tính $j = a^{(p-1)/2} \bmod p$.
4. Tính số Jacobi J(a,p).
5. Nếu j $\neq$ J(a,p), thì p không phải là số nguyên tố.
6. Nếu j = J(a,p) thì nói p có thể là số nguyên tố với chắc chắn 50%.

Lặp lại các bước này n lần, mỗi lần với một giá trị ngẫu nhiên khác nhau của a. Phần dư của hợp số với n phép thử là không quá 2^n.

Thực tế khi thực hiện chương trình, thuật toán chạy với tốc độ khá nhanh.

4.3. Thuật toán Rabin-Miller

Thuật toán này được phát triển bởi Rabin, dựa trên một phần ý tưởng của Miller. Thực tế những phiên bản của thuật toán đã được giới thiệu tại NIST. (National Institute of Standards and Technology).

Đầu tiên là chọn ngẫu nhiên một số p để kiểm tra. Viết p dưới dạng $p = 1+2^b m$ trong đó m là một số lẻ.

Sau đây là thuật toán :

1. Chọn một số ngẫu nhiên a, và giả sử a nhỏ hơn p.
2. Đặt j=0 và $z=a^m \bmod p$.
3. Nếu z=1, hoặc z=p-1 thì p đã qua bước kiểm tra và có thể là số nguyên tố.
4. Nếu j > 0 và z=1 thì p không phải là số nguyên tố.
5. Đặt j = j+1. Nếu j < b và $z \neq p-1$ thì đặt $z=z^2 \bmod p$ và trở lại bước 4.
6. Nếu j = b và $z \neq p-1$, thì p không phải là số nguyên tố.

4.4. Thuật toán Lehmann.

Một phương pháp đơn giản hơn kiểm tra số nguyên tố được phát triển độc lập bởi Lehmann. Sau đây là thuật toán với số bước lặp là 100.

1. Chọn ngẫu nhiên một số n để kiểm tra.
2. Chắc chắn rằng n không chia hết cho các số nguyên tố nhỏ như 2,3,5,7 và 11.
3. Chọn ngẫu nhiên 100 số $a_1, a_2, \ldots, a_{100}$ giữa 1 và n-1.
4. Tính $a_i^{(n-1)/2} \pmod n$ cho tất cả $a_i = a_1 \ldots a_{100}$. Dừng lại nếu bạn tìm thấy a_i sao cho phép kiểm tra là sai.
5. Nếu $a_i^{(n-1)/2} = 1 \pmod n$ với mọi i, thì n có thể là hợp số.

   Nếu $a_i^{(n-1)/2} \neq 1$ hoặc $-1 \pmod n$ với i bất kỳ, thì n là hợp số.

   Nếu $a_i^{(n-1)/2} = 1$ hoặc $-1 \pmod n$ với mọi $i \neq 1$, thì n là số nguyên tố.

5. Bài tập

Bài tập 2.1: hãy tính $17^{53} \bmod 29$, hỏi cần dùng ít nhất là bao nhiêu phép nhân để tìm ra kết quả.

Bài tập 2.2: Tính $876^{611} \bmod 899$.

Sử dụng một trong các ngôn ngữ lập trình C, C++, Java hoặc C# để làm các bài tập sau:

Bài tập 2.3: Viết chương trình cài đặt thuật toán tìm phần tử nghịch đảo.

Bài tập 2.4: Viết chương trình cài đặt thuật toán lũy thừa nhanh.

Bài tập 2.5: Viết chương trình giải hệ phương trình đồng dư bậc nhất hai ẩn.

Bài tập 2.6: Viết chương trình cài đặt thuật toán kiểm tra số nguyên tố với input là một số nguyên nhỏ hơn 2000000000.

Bài tập 2.7: Viết chương trình cài đặt thư viện số nguyên lớn với các thao tác tính toán cơ bản: nhân, chia, cộng trừ, lấy modulo.

Bài tập 2.8: Sử dụng thư viện số lớn (ở bài tập 2.5 hoặc một thư viện mã nguồn mở) cài đặt các thuật toán kiểm tra số nguyên tố được trình bày trong phần 4 của chương 2.

CHƯƠNG III: CÁC HỆ MÃ KHÓA BÍ MẬT

1. Các hệ mã cổ điển

1.1. Hệ mã hoá thay thế (substitution cipher)

Hệ mã hoá thay thế là hệ mã hoá trong đó mỗi ký tự của bản rõ được thay thế bằng ký tự khác trong bản mã (có thể là một chữ cái, một số hoặc một ký hiệu).

Có 4 kỹ thuật thay thế sau đây:

1. Thay thế đơn (A simple substitution cipher): là hệ trong đó một ký tự của bản rõ được thay bằng một ký tự tương ứng trong bản mã. Một ánh xạ 1-1 từ bản rõ tới bản mã được sử dụng để mã hoá toàn bộ thông điệp.
2. Thay thế đồng âm (A homophonic substitution cipher): giống như hệ thống mã hoá thay thế đơn, ngoại trừ một ký tự của bản rõ có thể được ánh xạ tới một trong số một vài ký tự của bản mã: sơ đồ ánh xạ 1-n (one-to-many). Ví dụ, “A” có thể tương ứng với 5, 13, 25, hoặc 56, “B” có thể tương ứng với 7, 19, 31, hoặc 42, v.v.
3. Thay thế đa mẫu tự (A polyalphbetic substitution cipher): được tạo nên từ nhiều thuật toán mã hoá thay thế đơn. Ánh xạ 1-1 như trong trường hợp thay thế đơn, nhưng có thể thay đổi trong phạm vi một thông điệp. Ví dụ, có thể có năm thuật toán mã hoá đơn khác nhau được sử dụng; đặc biệt thuật toán mã hoá đơn được sử dụng thay đổi theo vị trí của mỗi ký tự trong bản rõ.
4. Thay thế đa sơ đồ (A polygram substitution cipher): là thuật toán trong đó các khối ký tự được mã hoá theo nhóm. Đây là thuật toán tổng quát nhất, cho phép thay thế các nhóm ký tự của văn bản gốc. Ví dụ, “ABA” có thể tương ứng với “RTQ”, “ABB” có thể tương ứng với “SLL”, v.v.

1.2. Hệ mã Caesar

Hệ mã Caesar là một hệ mã hoá thay thế đơn âm làm việc trên bảng chữ cái tiếng Anh 26 ký tự (A, B, ... , Z). Đây là hệ mã cổ điển và đơn giản nhất đã từng được dùng trong thực tế bởi hoàng đế La mã Caesar nên được đặt theo tên của vị hoàng đế này.

Không gian các bản rõ $\mathcal{P}$ là các thông điệp được tạo từ bảng chữ cái $\mathcal{A}$ (để tiện trình bày chúng ta xem đây là một bảng chữ cái tổng quát). Tương tự không gian các bản mã $\mathcal{C} \equiv \mathcal{P}$. Giả sử số phần tử của bảng chữ cái $|\mathcal{A}| = N$.

Để mã hóa người ta đánh số các chữ cái từ 0 tới N-1. Không gian khóa $\mathcal{K} = Z_N$. Với mỗi khóa $K \in \mathcal{K}$ hàm mã hóa và giải mã một ký tự có số thứ tự là i sẽ được thực hiện như sau:

Mã hóa: $E_K(i) = (i + k) \bmod N$.

Giải mã: $D_K(i) = (i – k) \bmod N$.

Hệ mã Caesar với bảng chữ cái tiếng Anh sẽ có N = 26 chữ cái, bảng chữ cái được đánh số như sau:

A	B	C	D	...	L	M	N	...	W	X	Y	Z
0	1	2	3	...	11	12	13	...	22	23	23	25

Bảng 3.1: Bảng đánh số các chữ cái tiếng Anh

Các phép tính toán số học được thực hiện trên vành Z_{26}, số khóa có thể sử dụng là 26 nhưng trên thực tế chỉ có 25 khóa có ích.

Ví dụ: với k=3 (trường hợp đã được hoàng đế Caesar sử dụng), ký tự A được thay bằng D, B được thay bằng E, ... , W được thay bằng Z, ... , X được thay bằng A, Y được thay bằng B, và Z được thay bằng C.

Bảng chữ cái gốc:

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

Bảng chữ cái dùng để mã hoá:

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

Do đó chẳng hạn xâu “ANGLES” sẽ được mã hóa thành “DQJOHV”.

Hệ mã Caesar sử dụng phương pháp thay thế đơn âm nên có hiện tượng gọi là phụ thuộc tần suất xuất hiện của ngôn ngữ tự nhiên. Trong ngôn ngữ tự nhiên một số chữ cái xuất hiện nhiều hơn so với các chữ cái khác (chẳng hạn trong tiếng Anh các chữ cái xuất hiện nhiều là e, t, i, h …) nên các chữ cái dùng để thay thế cho chúng cũng xuất hiện nhiều. Điều này có thể dẫn tới hệ quả là người thám mã có thể sử dụng phương pháp thử thay thế các ký tự xuất hiện nhiều trong bản mã bằng các ký tự xuất hiện nhiều trên các văn bản thực tế.

Trên thực tế hệ mã Caesar có số khóa ít nên hoàn toàn có thể thám mã bằng cách thử tất cả các khóa có thể (kiểu tấn công Brute force).

1.3. Hệ mã Affine

Không gian các bản rõ và bản mã của hệ mã là các xâu được hình thành từ một bảng chữ cái $\mathcal{A}$, giả sử $|\mathcal{A}|$ = N. Khi đó không gian khóa của hệ mã được xác định như sau:

$\mathcal{K}$ = { (a, b): a, b $\in Z_N$, (a, N) = 1}

Để mã hóa người ta đánh số các chữ cái của bảng chữ cái từ 0 tới N – 1 và tiến hành mã hóa, giải mã từng ký tự (thay thế) theo các công thức sau:

Mã hóa:

$E_K(x) = (a*x + b) \bmod N$. Ký tự bản rõ có số thứ tự là x sẽ được chuyển thành ký tự có số thứ tự là (a*x+b) mod N trong bảng chữ cái.

Để giải mã ta cần tìm a^{-1} (do (a, N) = 1 nên luôn tìm được) và tiến hành công thức giải mã sau:

$D_K(y) = a*(y - b) \bmod N$. Ký tự bản mã có số thứ tự là y sẽ được thay thế bằng ký tự có số thứ tự là a*(y - b) mod N trong bảng chữ cái.

Có thể thấy rằng đối với một hệ mã Affine thì số khóa có thể sử dụng sẽ là:

$|\mathcal{K}| = \varnothing(N) * N$. Ví dụ với N = 26 tương ứng với bảng chữ cái tiếng Anh chúng ta sẽ có $\varnothing(26) * 26 = 12 * 26 = 312$ khóa. Con số này là tương đối nhỏ.

1.4. Hệ mã Vigenere

Hệ mã này được đặt theo tên của một nhà mật mã học người Pháp Blaise de Vigenère (1523-1596).

Đối với hệ mã này không gian các bản mã và bản rõ cũng là các thông điệp được tạo thành từ một bảng chữ cái $\mathcal{A}$ như trong hệ mã Caesar, các chữ cái được đanh số từ 0 tới N-1 trong đó N là số phần tử của bảng chữ cái.

Không gian khóa $\mathcal{K}$ được xác định như sau:

Với mỗi số nguyên dương M, khóa có độ dài M là một xâu ký tự có độ dài M, $K = k_1k_2\ldots k_M$.

Để mã hóa một bản rõ P người ta chia P thành các đoạn độ dài M và chuyển thành số thứ tự tương ứng của chúng trong bảng chữ cái, chẳng hạn $X = x_1x_2\ldots x_M$. Khi đó việc mã hóa và giải mã được thực hiện như sau:

$E_K(X) = (x_1 + k_1, x_2 + k_2, \ldots, x_M + k_M) \bmod N$

$D_K(Y) = (y_1 - k_1, y_2 - k_2, \ldots, y_M - k_M) \bmod N$ với N là số phần tử của bảng chữ cái và $Y = y_1y_2\ldots y_M$ là bản mã.

Ví dụ: xét $\mathcal{A}$ là bảng chữ cái tiếng Anh, ta có N = 26 giả sử khóa có độ dài 6 và K = "CIPHER", bản rõ P = "THIS CRYPTOSYSTEM IS NOT SECURE". Ta có K = 2 8 15 7 4 17, P = 19 7 8 18 2 17 | 24 15 19 14 18 23 | 18 19 4 12 8 18 | 13 14 19 18 4 2 | 20 17 4. Quá trình mã hóa thực hiện như sau:

P = 19 7 8 18 2 17 | 24 15 19 14 18 23 | 18 19 4 12 8 18 | 13 14 19 18 4 2 | 20 17 4

K = 2 8 15 7 4 17 | 2 8 15 7 4 17 | 2 8 15 7 4 17 | 2 8 15 7 4 17 | 2 8 15

C = 21 15 23 25 6 8 | 0 23 8 21 22 14 | 20 1 19 19 12 9 | 15 22 8 25 8 19 | 22 25 19

Vậy bản mã là C = "VPXZGI AXIVWO UBTTMJ PWIZIT WZT".

Về thực chất hệ mã này là kết hợp của nhiều mã Caesar, trong hệ mã Caesar chúng ta thay thế từng ký tự đơn lẻ thì trong hệ mã Vigenere này thay thế từng bộ M ký tự liên tiếp. Với mỗi M chúng ta có số khóa có thể sử dụng là N^M, cụ thể là với bảng chữ cái tiếng Anh sẽ có 26^M khóa có thể sử dụng.

1.5. Hệ mã Hill

Hệ mã hoá này dựa trên lý thuyết về đại số tuyến tính do Lester S.Hill đưa ra năm 1929.

Cả không gian bản rõ và bản mã đều là các xâu được thành lập từ một bảng chữ cái $\mathcal{A}$ như trong hệ mã Vigenere.

Với mỗi số nguyên M khóa của hệ mã là một ma trận K vuông kích thước MxM gồm các phần tử là c ác số nguyên thuộc Z_N trong đó N là số phần tử của bảng chữ cái. Điều kiện để ma trận K có thể sử dụng làm khóa của hệ mã là K phải là một ma trận không suy biến trên Z_N hay nói cách khác là tồn tại ma trận nghịch đảo của ma trận K trên Z_N.

Các ký tự của bảng chữ cái cũng được đánh số từ 0 tới N-1.

Để mã hóa một bản rõ người ta cũng chia bản rõ đó thành các xâu có độ dài M, chuyển các xâu này thành số thứ tự của các chữ cái trong bảng chữ cái dưới dạng một vectơ hàng M chiều và tiến hành mã hóa, giải mã theo công thức sau:

Mã hóa:

$C = P * K.$

Giải mã:

$P = C * K^{-1}.$

Ví dụ: cho hệ mã Hill có M = 2 (khóa là các ma trận vuông cấp 2) và bảng chữ cái là bảng chữ cái tiếng Anh, tức là N = 26. Cho khóa

$$K = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix}$$

Hãy mã hóa xâu P = “HELP” và giải mã ngược lại bản mã thu được.

Để mã hóa chúng ta chia xâu bản rõ thành hai vecto hàng 2 chiều “HE” (7 4) và “LP” (11 15) và tiến hành mã hóa lần lượt.

Với $P_1 = (7\ 4)$ ta có $C_1 = P_1 * K = (7\ 4)\begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} = (3\ 15) = (\mathrm{D\ P})$

Với $P_2 = (11\ 15)$ ta có $C_2 = P_2 * K = (11\ 15)\begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} = (11\ 4) = (\mathrm{L\ E})$

Vậy bản mã thu được là C = “DPLE”.

Để giải mã ta tính khóa giải mã là ma trận nghịch đảo của ma trận khóa trên Z_{26} theo công thức sau:

Với $K = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$ và $\det(K) = (k_{11}*k_{22} - k_{21}*k_{12}) \bmod N$ là một phần tử có phần tử nghịch đảo trên Z_N (ký hiệu là $\det(K)^{-1}$) thì khóa giải mã sẽ là

$$K^{-1} = \det(K)^{-1} * \begin{pmatrix} k_{22} & -k_{12} \\ -k_{21} & k_{11} \end{pmatrix}$$

Áp dụng vào trường hợp trên ta có det(K) = (15 - 6) mod 26 = 9. GCD(9, 26) =1 nên áp dụng thuật toán Ơclit mở rộng tìm được $det(K)^{-1}$ = 3. Vậy K^{-1} = 3 * $\begin{pmatrix} 5 & 23 \\ 24 & 3 \end{pmatrix}$ = $\begin{pmatrix} 15 & 17 \\ 20 & 9 \end{pmatrix}$.

Quá trình giải mã tiến hành giống như quá trình mã hóa với khóa mã hóa thay bằng khóa giải mã.

Giải mã C = "DP" = (3 15), P = C * K^{-1} = (3 15) * $\begin{pmatrix} 15 & 17 \\ 20 & 9 \end{pmatrix}$ = $(3\ 15)$ = "HE".

Tương tự giải mã xâu C = "LE" kết quả sẽ được bản rõ P = "LP".

Chú ý là trong ví dụ trên chúng ta sử dụng khóa K có kích thước nhỏ nên dễ dàng tìm được khóa để giải mã còn trong trường hợp tổng quát điều này là không dễ dàng.

1.6. Hệ mã đổi chỗ (transposition cipher)

Một hệ mã hoá đổi chỗ là hệ mã hoá trong đó các ký tự của bản rõ vẫn được giữ nguyên, nhưng thứ tự của chúng được đổi chỗ cho nhau.

Ví dụ một hệ mã hoá đổi chỗ cột đơn giản, bản rõ được viết theo hàng ngang trên trang giấy với độ dài cố định, và bản mã được đọc theo hàng dọc.

Bản rõ: COMPUTER GRAPHICS MAY BE SLOW BUT AT LEAST IT'S EXPENSIVE		
	COMPUTERGR	
	APHICSMAYB	
	ESLOWBUTAT	
	LEASTITSEX	
	PENSIVE	
Bản mã: CAELPOPSEEMHLANPIOSSUCWTITSBIUEMUTERATSGYAERBTX		

Bảng 3.2: Mã hoá thay đổi vị trí cột

Phương pháp này có các kỹ thuật sau:

1. **Đảo ngược toàn bộ bản rõ**: nghĩa là bản rõ được viết theo thứ tự ngược lại để tạo ra bản mã. Đây là phương pháp mã hoá đơn giản nhất vì vậy không đảm bảo an toàn.

Ví dụ: bản rõ "TRANSPOSITION CIPHER" được mã hoá thành "REHPICNOITISOPSNART".

2. **Mã hoá theo mẫu hình học**: bản rõ được sắp xếp lại theo một mẫu hình học nào đó, thường là một mảng hoặc một ma trận hai chiều.

Ví dụ: bản rõ "LIECHTENSTEINER" được viết thành ma trận 3×5 theo hàng như sau:

Cột	1	2	3	4	5
Bản rõ	L	I	E	C	H

	T	E	N	S	T
	E	I	N	E	R

Bảng 3.3: Mã hóa theo mẫu hình học

Nếu lấy các ký tự ra theo số thứ tự cột 2, 4, 1, 3, 5 thì sẽ có bản mã "IEICSELTEENNHTR".

Đổi chỗ cột: Đầu tiên đổi chỗ các ký tự trong bản rõ thành dạng hình chữ nhật theo cột, sau đó các cột được sắp xếp lại và các chữ cái được lấy ra theo hàng ngang

Ví dụ: bản rõ gốc là "NGAY MAI BAT DAU CHIEN DICH XYZ" được viết dưới dạng ma trận 5×5 theo cột như sau:

Cột	1	2	3	4	5
Bản rõ	N	A	D	I	C
	G	I	A	E	H
	A	B	U	N	X
	Y	A	C	D	Y
	M	T	H	I	Z

Bảng 3.4: Ví dụ mã hóa theo mẫu hình học

Vì có 5 cột nên chúng có thể được sắp lại theo 5!=120 cách khác nhau. Để tăng độ an toàn có thể chọn một trong các cách sắp xếp lại đó.

Nếu ta chuyển vị các cột theo thứ tự 3, 5, 2, 4, 1 rồi lấy các ký tự ra theo hàng ngang ta sẽ được bản mã là "DCAINAHIEGUXBNACYADY HZTIM". Lưu ý rằng các ký tự cách được bỏ đi.

Hạn chế của phương pháp này là toàn bộ các ma trận ký tự phải được sinh để mã hoá và giải mã.

3. **Hoán vị các ký tự của bản rõ theo chu kỳ cố định d**: Nếu hàm f là một hoán vị của một khối gồm d ký tự thì khoá mã hoá được biểu diễn bởi K(d,f).

Do vậy, bản rõ:

$$M = m_1 m_2 ... m_d m_{d+1} ... m_{2d}$$

Với m_i là các ký tự , và bản rõ sẽ được mã hoá thành

$$E_k(M) = m_{f(1)} m_{f(2)} ... m_{f(d)} m_{f(d)+1} ... m_{d+f(d)}$$

Trong đó $m_{f(1)} m_{f(2)} ... m_{f(d)}$ là một hoán vị của $m_1 m_2 ... m_d$.

Ví dụ: giả sử d=5 và f hoán vị dãy i=12345 thành f(i)=35142

Vị trí đầu	Vị trí hoán vị	Từ	Mã hoá
1	3	G	O
2	5	R	P

3	1	O	G
4	4	U	U
5	2	P	R

Bảng 3.5: Mã hóa hoán vị theo chu kỳ

Theo bảng trên, ký tự đầu trong khối 5 ký tự được chuyển tới vị trí thứ 3, ký tự thứ hai được chuyển tới vị trí thứ 5, ... Chẳng hạn từ gốc GROUP được mã hoá thành OPGUR. Bằng cách đó, bản rõ "I LOVE BEETHOVENS MUSIC" sẽ được chuyển thành "OEIVLEHBTEESONVSCMIU".

Hệ mã ADFGV của Đức, được sử dụng trong suốt chiến tranh thế giới lần thứ I, là một hệ mã hoá đổi chỗ (có sử dụng phương pháp thay thế đơn giản). Nó được coi là một thuật toán mã hoá phức tạp vào thời ấy nhưng nó đã bị phá bởi Georges Painvin, một nhà thám mã người Pháp. Trên thực tế có rất nhiều hệ thống mã hoá sử dụng phương pháp đổi chỗ, nhưng chúng rất rắc rối vì thường đòi hỏi không gian nhớ lớn.

2. Các hệ mã khối

Trong phần này chúng ta sẽ học về các hệ mã khối điển hình là chuẩn mã hóa dữ liệu DES (Data Encryption Standard), một trong số các hệ mã khối được sử dụng rộng rãi nhất và là nền tảng cho rất nhiều các hệ mã khối khác.

Chuẩn mã hóa dữ liệu DES là một chuẩn mã hoá được công bố bởi Uỷ ban Tiêu chuẩn quốc gia Hoa Kỳ vào 15/02/1977. Hệ mã này được xây dựng dựa trên một hệ mã khối phổ biến có tên là LUCIFER và được phát triển bởi IBM.

DES có nhiều ưu điểm (nhanh, thuật toán công khai, dễ cài đặt) và đã từng được sử dụng trên thực tế trong một thời gian rất dài (cho đến trước đầu những năm 90) tuy nhiên theo thời gian năng lực của các máy tính phát triển cùng với các kỹ thuật thám mã mới được đưa ra đã cho thấy nhu cầu về một hệ mã khối mạnh hơn và chuẩn mã hóa cao cấp AES đã ra đời. Chuẩn này ra đời dựa trên một cuộc thi về thiết kế một hệ mã khối an toàn hơn (vào năm 1997) thay thế cho DES của Ủy ban Tiêu chuẩn quốc gia của Hoa Kỳ (NIST). Có rất nhiều hệ mã đã được gửi đến làm ứng cử viên cho AES nhưng cuối cùng hệ mã Rijndael của hai tác giả người Bỉ là tiến sĩ Joan Daemen và tiến sĩ Vincent Rijmen (vào năm 2001).

2.1. Mật mã khối

Các hệ mã cổ điển mà chúng ta xem xét ở phần đầu chương này đều có đặc điểm chung là từng ký tự của bản rõ được mã hoá tách biệt. Điều này làm cho việc phá mã trở nên dễ dàng hơn. Chính vì vậy, trên thực tế người ta hay dùng một kiểu mật mã khác, trong đó từng khối ký tự của bản rõ được mã hoá cùng một lúc như là một đơn vị mã hoá đồng nhất. Trong kiểu mã hoá này, các tham số quan trọng là kích thước (độ dài) của mỗi khối và kích thước khoá.

Điều kiện để mã hoá khối an toàn:

- Kích thước khối phải đủ lớn để chống lại phương án tấn công bằng phương pháp thống kê. Tuy nhiên điều này sẽ dẫn đến thời gian mã hoá sẽ tăng lên.

- Không gian khoá, tức chiều dài khoá phải đủ lớn để chống lại phương án tấn công bằng vét cạn. Tuy nhiên khoá phải đủ ngắn để việc tạo khoá, phân phối và lưu trữ khoá được dễ dàng.

Khi thiết kế một hệ mã khối, phải đảm bảo hai yêu cầu sau:

- Sự hỗn loạn (confusion): sự phụ thuộc giữa bản rõ và bản mã phải thực sự phức tạp để gây khó khăn đối với việc tìm quy luật thám mã. Mối quan hệ này tốt nhất là phi tuyến.
- Sự khuếch tán (diffusion): Mỗi bit của bản rõ và khóa phải ảnh hưởng lên càng nhiều bit của bản mã càng tốt.

Trong khi sự hỗn loạn (confusion) được tạo ra bằng kỹ thuật thay thế thì sự khuếch tán (diffusion) được tạo ra bằng các kỹ thuật hoán vị. Các hệ mã khối mà chúng ta xem xét trong phần này đều thỏa mãn các yêu cầu đó.

Ngoài các hệ mã khối được trình bày trong phần này còn rất nhiều các hệ mã khối khác đã phát triển qua thời gian (tại các quốc gia khác nhau và ứng dụng trong các lĩnh vực khác nhau), có thể kể ra đây một số hệ mã nổi tiếng như: Lucifer (1969), DES (1977), Madryga (1984), NewDES (1985), FEAL, REDOC, LOKI (1990), Khufu and Khafre (1990), RC2, RC4, IDEA (1990), MMB, CA-1.1, Shipjack, GOST, CAST, Blowfish, SAFER, 3-Way, Crab, SXAL8/MBAL, SAFER, RC5, RC6 ...

Đặc điểm chung của các hệ mã khối là quá trình mã hóa làm việc với các khối dữ liệu (thường ở dạng xâu bit) có kích thước khác nhau (tối thiểu là 64 bit), khóa của hệ mã cũng là một xâu bit có độ dài cố định (56 bit với DES, các hệ mã khác là 128, 256, hoặc thậm chí 512 bit). Tất cả các hệ mã này đều dựa trên lý thuyết của Shannon đưa ra năm 1949 và nếu mang mã hóa hai bản rõ giống nhau sẽ thu được cùng một bản mã. Hoạt động của các hệ mã khối thường được thực hiện qua một số lần lặp, mỗi lần sẽ sử dụng một khóa con được sinh ra từ khóa chính.

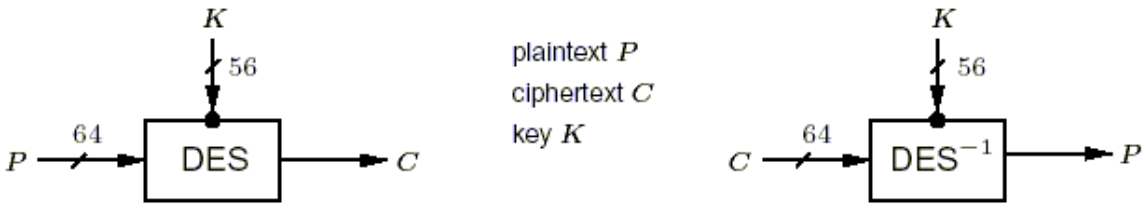
2.2. Chuẩn mã hoá dữ liệu DES (Data Encryption Standard)

Vào cuối thập niên 60, hệ mã Lucifer đã được đưa ra bởi Horst Feistel. Hệ mã này gắn liền với hãng IBM nổi tiếng. Sau đó Uỷ ban Tiêu chuẩn Hoa Kỳ đã dàn xếp với IBM để thuật toán mã hóa này thành miễn phí và phát triển nó thành chuẩn mã hóa dữ liệu và công bố vào ngày 15/02/1977.

2.2.1. Mô tả sơ đồ mã hoá DES

Mô tả tổng quan:

DES là thuật toán mã hóa với input là khối 64 bit, output cũng là khối 64 bit. Khóa mã hóa có độ dài 56 bit, thực ra chính xác hơn phải là 64 bit với các bit ở vị trí chia hết cho 8 có thể sử dụng là các bit kiểm tra tính chẵn lẻ. Số khóa của không gian khóa K là 2^{56}.



Hình 3.1: Chuẩn mã hóa dữ liệu DES

Thuật toán thực hiện 16 vòng. Từ khóa input K, 16 khóa con 48 bit K_i sẽ được sinh ra, mỗi khóa cho một vòng thực hiện trong quá trình mã hóa. Trong mỗi vòng, 8 ánh xạ thay thế 6 bit thành 4 bit S_i (còn gọi là hộp S_i) được chọn lựa kỹ càng và cố định, ký hiệu chung là S sẽ được sử dụng. Bản rõ 64 bit sẽ được sử dụng chia thành hai nửa L_0 và R_0. Các vòng có chức năng giống nhau, nhận input là L_{i-1} và R_{i-1} từ vòng trước và sinh ra output là các xâu 32 bit L_i và R_i như sau:

$L_i = R_{i-1}$; (1)

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$ trong đó $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$; (2)

Trong đó:

- $\oplus$ là ký hiệu của phép tuyển loại trừ (XOR) của hai xâu bit theo modulo 2.
- Hàm f là một hàm phi tuyến.
- E là hoán vị mở rộng ánh xạ R_{i-1} từ 32 bit thành 48 bit (đôi khi tất cả các bit sẽ được sử dụng hoặc một bit sẽ được sử dụng hai lần).
- P là hoán vị cố định khác của 32 bit.

Một hoán vị bit khởi đầu (IP) được sử dụng cho vòng đầu tiên; sau vòng cuối cùng nửa trái và phải sẽ được đổi cho nhau và cuối cùng xâu kết quả sẽ được hoán vị bit lần cuối bởi hoán vị ngược của IP (IP^{-1}).

Quá trình giải mã diễn ra tương tự nhưng với các khoá con ứng dụng vào các vòng trong theo thứ tự ngược lại.

Có thể hình dung đơn giản là phần bên phải trong mỗi vòng (sau khi mở rộng input 32 bit thành 8 ký tự 6 bit – xâu 48 bit) sẽ thực hiện một tính toán thay thế phụ thuộc khóa trên mỗi một ký tự trong xâu 48 bit, và sau đó sử dụng một phép chuyển bit cố định để phân bố lại các bit của các ký tự kết quả hình thành nên output 32 bit.

Các khoá con K_i (chứa 48 bit của K) được tính bằng cách sử dụng các bảng PC1 và PC2 (Permutation Choice 1 và 2). Trước tiên 8 bit (k_8, k_{16},…,k_{64}) của K bị bỏ đi (áp dụng PC1). 56 bit còn lại được hoán vị và gán cho hai biến 28 bit C và D, và sau đó trong 16 vòng lặp cả C và D sẽ được quay 1 hoặc 2 bit, và các khóa con 48 bit K_i được chọn từ kết quả của việc ghép hai xâu với nhau.

Như vậy, ta có thể mô tả toàn bộ thuật toán sinh mã DES dưới dạng công thức như sau:

$$\mathbf{Y = IP^{-1} \bullet f_{16} \bullet T \bullet f_{15} \bullet T \bullet \ ... \ \bullet f_2 \bullet T \bullet f_1 \bullet IP(x)}$$

Trong đó:

- T mô tả phép hoán vị của các khối L_iR_i ($1 \le i \le 15$).
- f_i mô tả việc dùng hàm f với khoá K_i ($1 \le i \le 16$).

Thuật toán chi tiết:

Input: bản rõ $M = m_1m_2…m_{64}$, khóa 64 bit $K = k_1k_2…k_{64}$ (bao gồm cả 8 bit chẵn lẻ, việc thêm bit chẵn lẻ sao cho các đoạn khóa 8 bit có số bit 1 là lẻ)

Output: bản mã 64 bit $C = c_1c_2\ldots c_{64}$

1. Sinh khóa con. Tính các khóa con theo thuật toán sinh khóa con bên dưới

2. $(L_0,R_0) \leftarrow IP(m_1m_2\ldots m_{64})$ (Sử dụng bảng hoán vị IP để hoán vị các bit, kết quả nhận được chia thành hai nửa là $L_0 = m_{58}m_{50}\ldots m_8$, $R_0 = m_{57}m_{49}\ldots m_7$.)

3. (16 vòng) for i = 1 to 16

Tính các L_i và R_i theo các công thức (1) và (2), việc tính

$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ được thực hiện như sau:

a) Mở rộng $R_{i-1} = r_1r_2\ldots r_{32}$ từ 32 bit thành 48 bit bằng cách sử dụng hoán vị mở rộng E.

   $T \leftarrow E(R_{i-1})$. (Vì thế $T = r_{32}r_1r_2\ldots r_{32}r_1$)

b) $T' \leftarrow T \oplus K_i$. Biểu diễn T' như là các xâu gồm 8 ký tự 6 bit $T' = (B_1,\ldots,B_8)$

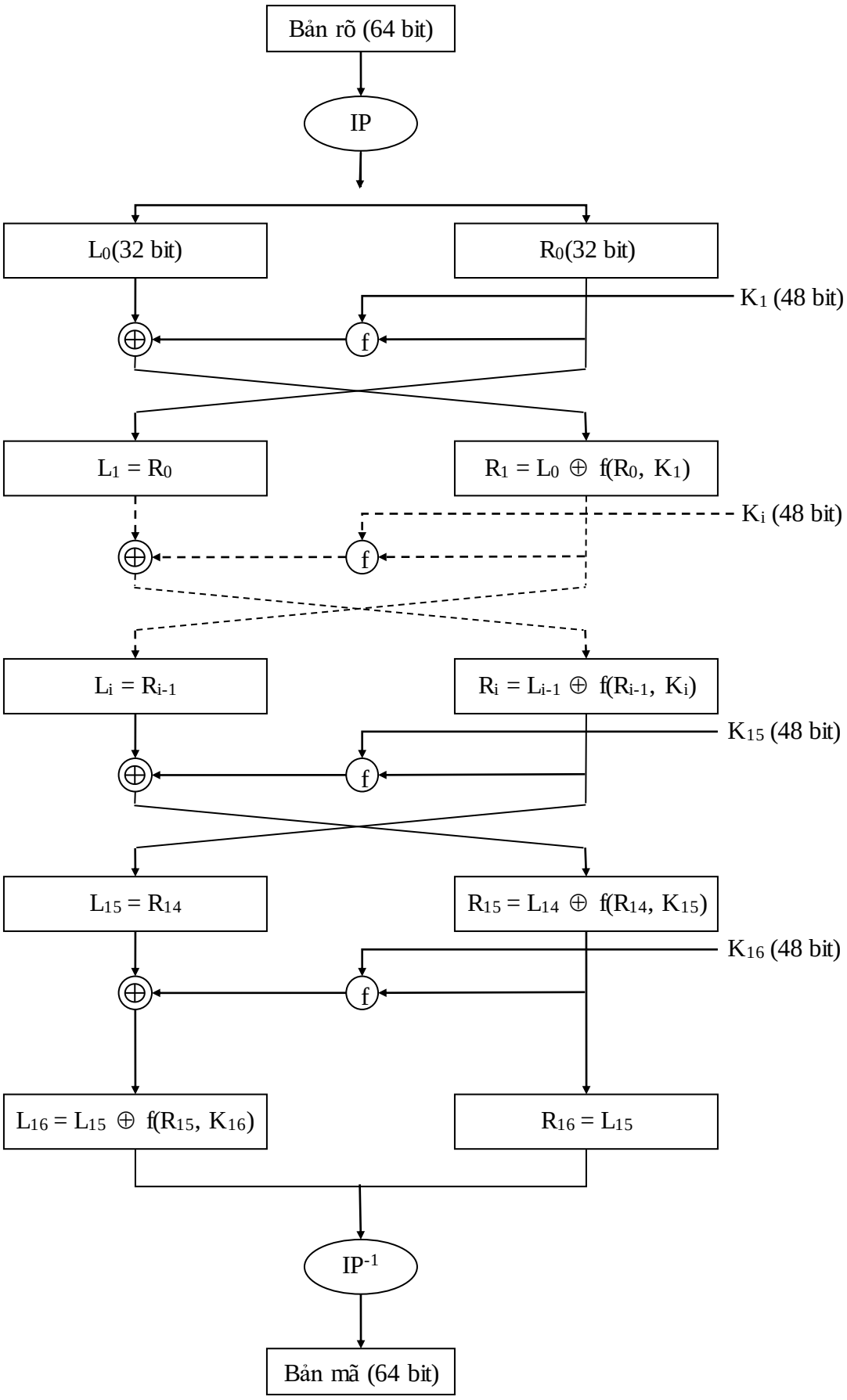
c) $T'' \leftarrow (S_1(B_1), S_2(B_2),\ldots,S_8(B_8))$. Trong đó $S_i(Bi)$ ánh xạ $b_1b_2\ldots b_6$ thành các xâu 4 bit của phần tử thuộc hàng r và cột c của các bảng S_i (S box) trong đó $r = 2 * b_1 + b_6$ và $c = b_2b_3b_4b_5$ là một số nhị phân từ 0 tới 15. Chẳng hạn $S_1(011011)$ sẽ cho r = 1 và c = 13 và kết quả là 5 biểu diễn dưới dạng nhị phân là 0101.

d) $T''' \leftarrow P(T'')$ trong đó P là hoán vị cố định để hoán vị 32 bit của $T'' = t_1t_2\ldots t_{32}$ sinh ra $t_{16}t_7\ldots t_{25}$.

4. $b_1b_2\ldots b_{64} \leftarrow (R_{16}, L_{16})$ (đổi vị trí các khối cuối cùng L_{16}, R_{16}

5. $C \leftarrow IP^{-1}(b_1b_2\ldots b_{64})$ (Biến đổi sử dụng IP^{-1}, $C = b_{40}b_8\ldots b_{25}$)

Sơ đồ 16 vòng lặp của DES:



Hình 3.2: Sơ đồ mã hoá DES

2.2.2. Hoán vị IP và hoán vị ngược IP^{-1}

Bảng hoán vị IP được đưa ra trong bảng dưới đây:

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
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

Bảng 3.6: Bảng hoán vị IP

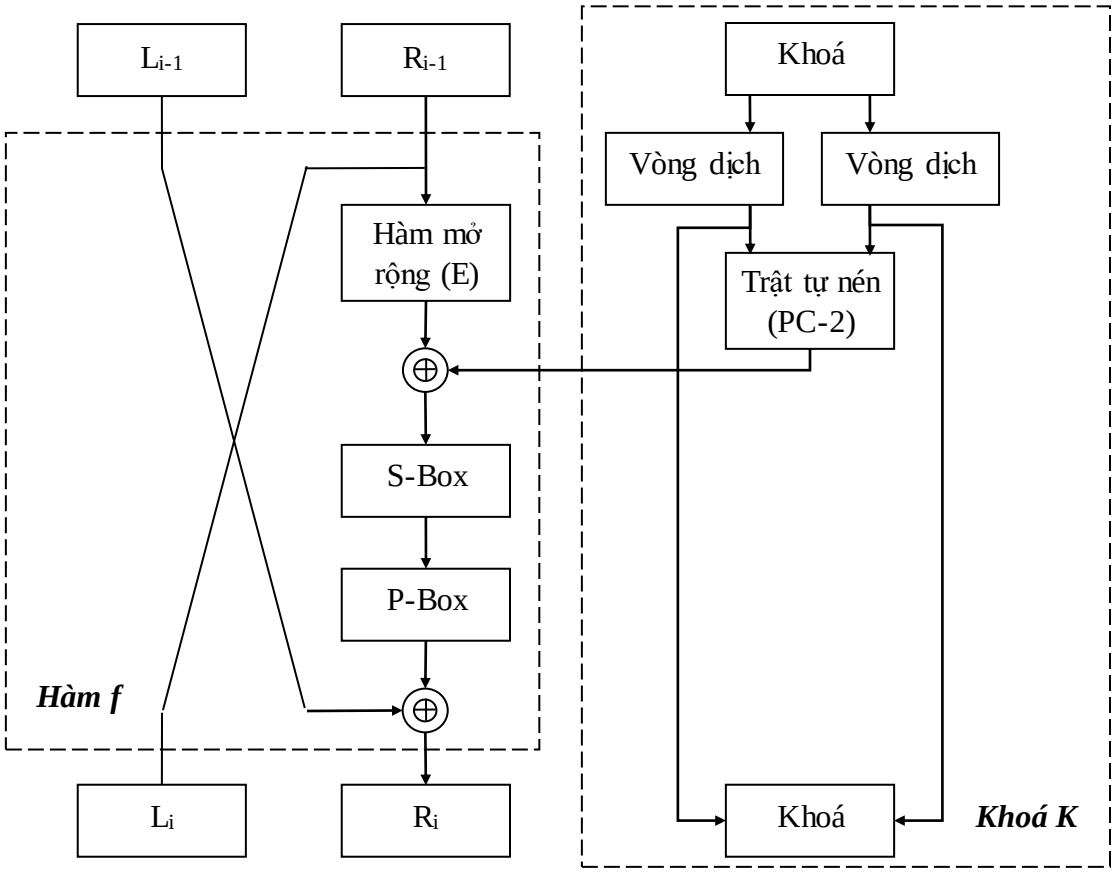
Bảng hoán vị ngược IP^{-1}:

40	8	48	16	56	24	64	32	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

Bảng 3.7: Bảng hoán vị ngược IP^{-1}

Hai hoán vị IP và IP^{-1} không có ý nghĩa gì về mặt mật mã mà hoàn toàn nhằm tạo điều kiện cho việc "chip hoá" thuật toán DES.

Sơ đồ cấu trúc một vòng DES:



Hình 3.3: Sơ đồ một vòng DES

2.2.3. Thuật toán sinh khóa con

Mười sáu vòng lặp của DES chạy cùng thuật toán như nhau nhưng với 16 khoá con khác nhau. Các khoá con đều được sinh ra từ khoá chính của DES bằng một thuật toán sinh khoá con. Khoá chính K (64 bit) đi qua 16 bước biến đổi, tại mỗi bước biến đổi này một khoá con được sinh ra với độ dài 48 bit.

Có thể mô tả thuật toán sinh các khóa con chi tiết như sau:

Input: khóa 64 bit K = k1k2...k64 (bao gồm cả 8 bit kiểm tra tính chẵn lẻ)

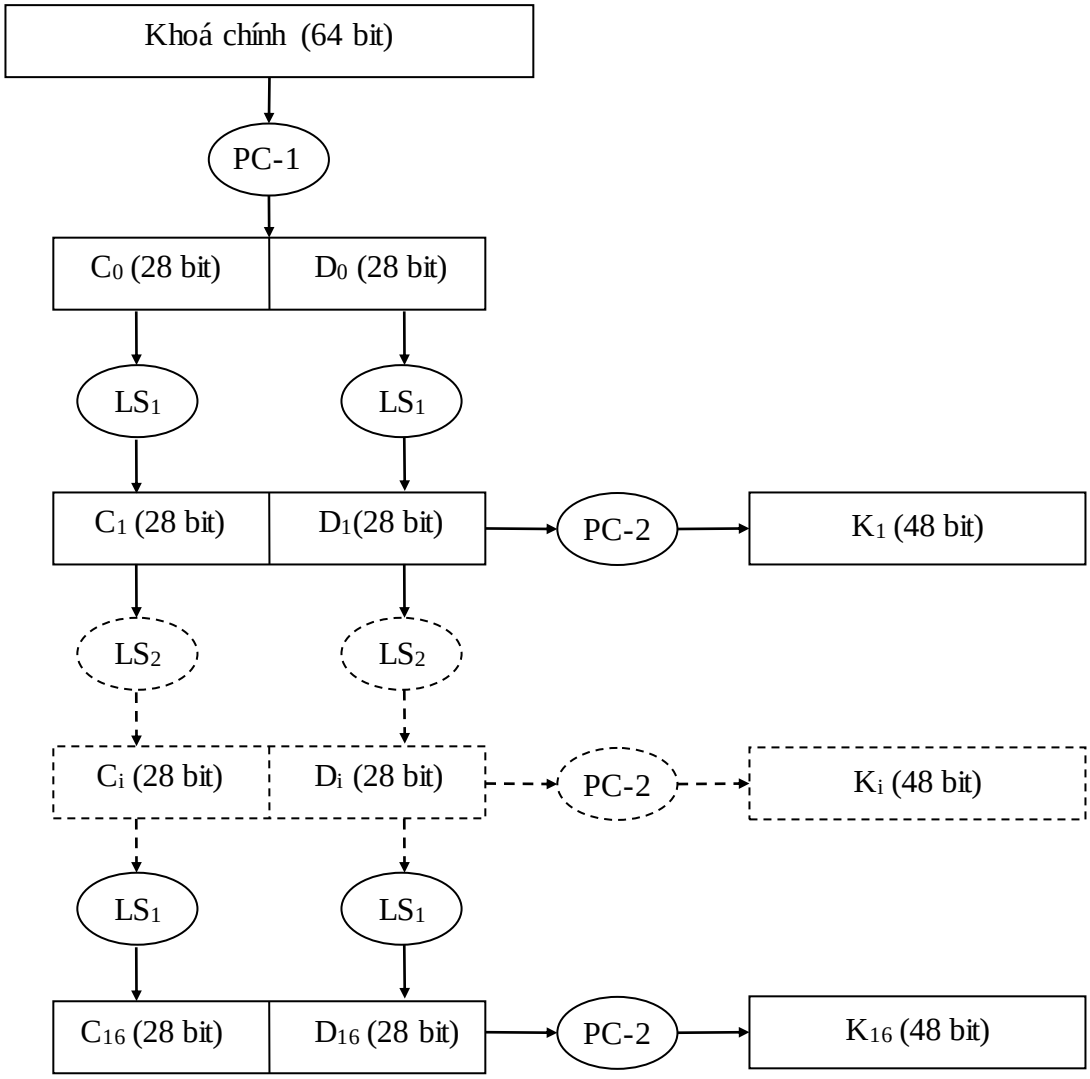
Output: 16 khóa con 48 bit K_i, $1 \le i \le 16$.

1) Định nghĩa v_i, $1 \le i \le 16$ như sau: $v_i = 1$ đối với $i \in \{1,2,9,16\}$; $v_i = 2$ cho các trường hợp khác (Đây là các giá trị dịch trái cho các quay vòng 28 bit bên dưới).

2) $T \leftarrow PC1(K)$; biểu diễn T thành các nửa 28 bit (C_0, D_0) (Sử dụng bảng PC1 để chọn các bit từ K: $C_0 = k_{57}k_{49}...k_{36}$, $D_0 = k_{63}k_{55}...k_4$.)

3) For i from 1 to 16, tính các K_i như sau: $C_i \leftarrow (C_{i-1} \leftarrow v_i)$, $D_i \leftarrow (D_{i-1} \leftarrow v_i)$, $K_i \leftarrow PC2(C_i, D_i)$. (Sử dụng bảng PC2 để chọn 48 bit từ xâu ghép $b_1b_2...b_{56}$ của C_i và D_i: $K_i = b_{14}b_{17}...b_{32}$. '← ' là ký hiệu dịch vòng trái.)

Sơ đồ sinh các khóa con của DES:



Hình 3.4: Sơ đồ tạo khoá con của DES

64 bit đầu vào sẽ giảm xuống còn 56 bit bằng cách bỏ đi 8 bit (ở các vị trí chia hết cho 8), các bit này dùng để kiểm tra bit chẵn lẻ. Sau đó 56 bit này lại được trích lấy 48 bit để sinh ra cho 16 vòng khoá của DES.

Bảng trật tự khoá (PC-1):

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

Bảng 3.8: Bảng PC-1

Đầu tiên 56 bit khóa được chia ra thành hai nửa 28 bit. Sau đó, hai nửa 28 bit này được dịch vòng trái hoặc 1 hoặc 2 bit phụ thuộc vào số bit dịch tương ứng với vòng đó.

Số bit dịch của các vòng (LS):

Vòng lặp	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Số bit dịch	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Bảng 3.9: Bảng dịch bit tại các vòng lặp của DES

Sau khi dịch vòng, một bảng chọn 48 bit được sử dụng. Vì cách hoán vị này của các bit được chọn như một tổ hợp con của các bit nên được gọi là "hoán vị nén" hay "trật tự nén".

Bảng trật tự nén(PC-2):

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Bảng 3.10: Bảng PC-2

Ví dụ như chúng ta có thể nhận thấy bit ở vị trí 33 của khoá sẽ dịch sang vị trí 35 ra ngoài, còn bit ở vị trí 18 của khoá sẽ bị bỏ qua. Chính việc dịch vòng này, tạo nên một tập hợp con của khoá được sử dụng trong mỗi tổ hợp khoá. Mỗi bit được sử dụng khoảng 14 lần trong tổng số 16 tổ hợp khoá, dù không phải tất cả các bít được sử dụng một cách chính xác cùng một lúc trong mỗi lần sử dụng.

2.2.4. Mô tả hàm f

Hàm $f(R_{i-1},K_i)$ là một hàm có hai biến vào: biến thứ nhất R_{i-1} là một xâu bit có độ dài 32 bit, biến thứ hai khoá K_i là một xâu bít có độ dài 48 bit. Đầu ra của f là một xâu bit có độ dài 32 bit. Hàm f có thể là hàm bất kỳ tuy nhiên vì nguồn gốc "sức mạnh" của DES nằm trong hàm f nên việc chọn hàm f phải cẩn thận để tránh bị phá mã một cách dễ dàng. Thông thường hàm f được chọn thường là hàm có tính chất $f = f^{-1}$, tức $f(f(x)) = x$.

Trong sơ đồ mô tả mã hoá của DES được công bố bởi Uỷ ban Tiêu chuẩn Quốc gia Hoa Kỳ (The Untied States Nation Bureau of Standard), hàm f thực hiện các việc sau:

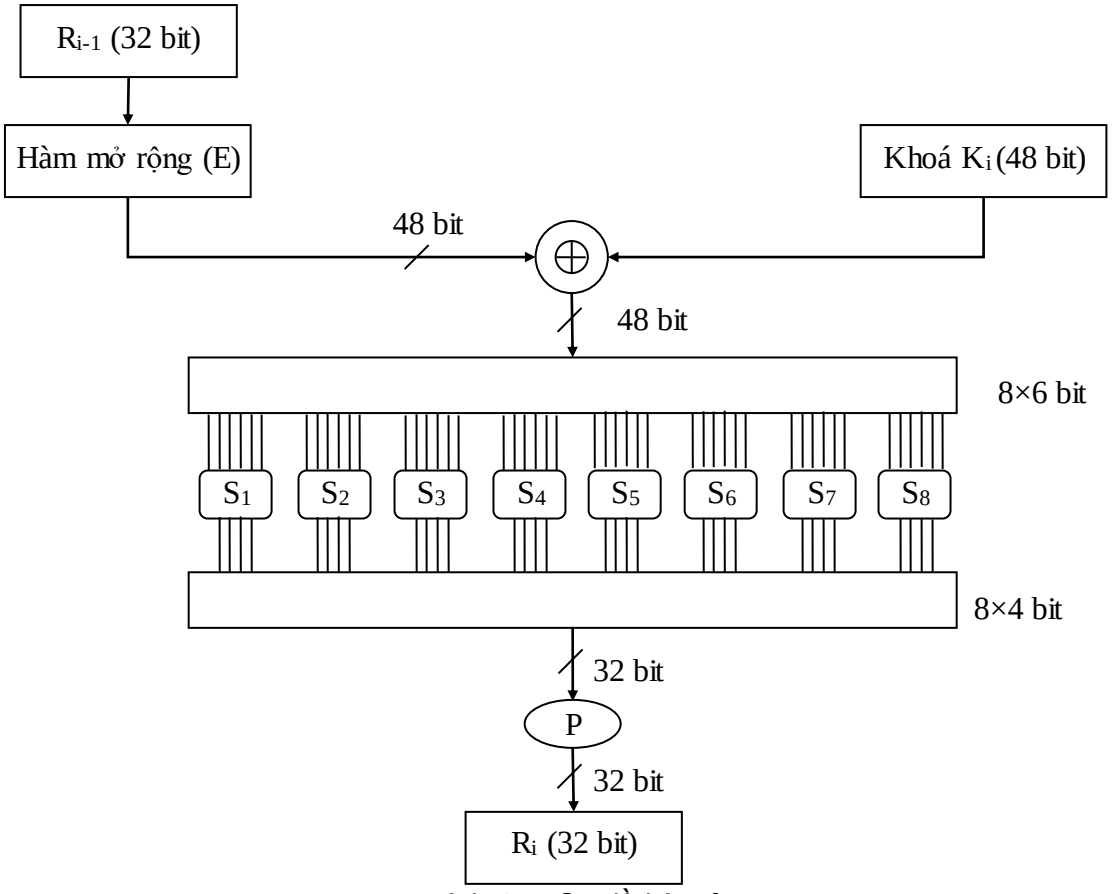
– Biến thứ nhất R_{i-1} được mở rộng thành một xâu bit có độ dài 48 bit theo một hàm mở rộng cố định E. Thực chất hàm mở rộng $E(R_{i-1})$ là một hoán vị có lặp trong đó lặp lại 16 bit của R_{i-1}.

– Tính $E(R_{i-1}) \oplus K_i$ và viết kết quả thành 8 xâu 6 bit $B_1B_2B_3B_4B_5B_6B_7B_8$.

– Đưa 8 khối B_i vào 8 bảng S_1, S_2, ..., S_8 (được gọi là các hộp S-Box). Mỗi hộp S-Box là một bảng 4*16 cố định có các cột từ 0 đến 15 và các hàng từ 0 đến 3. Với mỗi xâu 6 bit $B_i = b_1b_2b_3b_4b_5b_6$, ta tính được $S_i(B_i)$ như sau: hai bit b_1b_6 xác định hàng r trong hộp S_i, bốn bit $b_2b_3b_4b_5$ xác định cột c trong hộp S_i. Khi đó, $S_i(B_i)$ sẽ xác định phần tử $C_i = S_i(r,c)$, phần tử này viết dưới dạng nhị phân 4 bit. Như vậy, 8 khối 6 bit B_i ($1 \le i \le 8$) sẽ cho ra 8 khối 4 bit C_i với ($1 \le i \le 8$).

– Xâu bit $C = C_1C_2C_3C_4C_5C_6C_7C_8$ có độ dài 32 bit được hoán vị theo phép hoán vị P (hộp P-Box). Kết quả P(C) sẽ là kết quả của hàm $f(R_{i-1}, K_i)$, và cũng chính là R_i cho vòng sau.

Hàm f cũng có thể mô tả bằng hình vẽ sau:



Hình 3.5: Sơ đồ hàm f

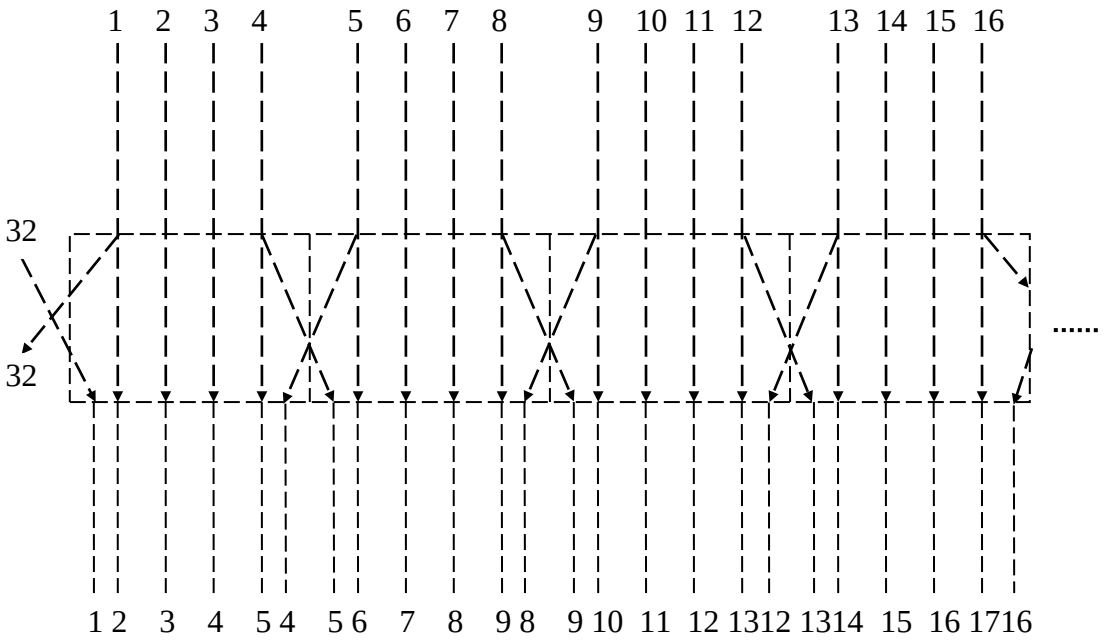
2.2.5. Hàm (ánh xạ) mở rộng (E)

Hàm mở rộng (E) sẽ tăng độ dài của Ri từ 32 bit lên 48 bit bằng cách thay đổi các thứ tự của các bit cũng như lặp lại các bit. Việc thực hiện này nhằm hai mục đích:

- Làm độ dài của R_i cùng cỡ với khoá K để thực hiện việc cộng modulo XOR.
- Cho kết quả dài hơn để có thể được nén trong suốt quá trình thay thế.

Tuy nhiên, cả hai mục đích này đều nhằm một mục tiêu chính là bảo mật dữ liệu. Bằng cách cho phép 1 bit có thể chèn vào hai vị trí thay thế, sự phụ thuộc của các bit đầu ra với các bit đầu vào sẽ trải rộng ra. DES được thiết kế với điều kiện là mỗi bit của bản mã phụ thuộc vào mỗi bit của bản rõ và khoá.

Sơ đồ hàm mở rộng:



Hình 3.6: Sơ đồ hàm mở rộng (E)

Đôi khi nó được gọi là hàm E-Box, mỗi 4 bit của khối vào, bit thứ nhất và bit thứ tư tương ứng với 2 bit của đầu ra, trong khi bit thứ 2 và 3 tương ứng với 1 bit ở đầu ra. Bảng sau đây miêu tả vị trí của bit ra so với bit vào.

Bảng mô tả hàm mở rộng (E):

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Bảng 3.11: Bảng mô tả hàm mở rộng E

Ví dụ như bit ở vị trí số 3 của khối vào sẽ di chuyển đến vị trí số 4 của khối ra và bit ở vị trí 21 ở đầu vào sẽ di chuyển đến vị trí 30 và 32 ở đầu ra.

2.2.6. Mô tả hộp S - Box

Đối với sơ đồ mã hoá DES, mọi tính toán đều là tuyến tính, tức là việc tính phép tuyển loại trừ XOR của hai đầu ra cũng giống với phép tuyển loại trừ XOR của hai đầu vào rồi tính toán đầu ra. Chỉ duy nhất có các tính toán với hộp S là phi tuyến. Chính vì vậy các hộp S-Box (chứa đựng các thành phần phi tuyến của hệ mật) là quan trọng nhất đối với độ mật của hệ mã, chính các hộp S tạo nên sự hỗn loạn (confusion) và sự khuếch tán (diffusion) của DES. Năm 1976, NSA đã đưa ra tiêu chuẩn thiết kế hộp S như sau:

- Mỗi hàng trong mỗi hộp S là một hoán vị của các số nguyên từ 0 đến 15.
- Không có hộp S nào là hàm Affine hay tuyến tính đối với các đầu vào của nó.
- Sự thay đổi của một bit đầu vào sẽ dẫn đến sự thay đổi ít nhất hai bit đầu ra.

– Đối với hộp S bất kỳ và với đầu vào x (một xâu bit có độ dài bằng 6) bất kỳ, thì S(x) và S(x⊕001100) phải khác nhau ít nhất là 2 bit.

NSA cũng tiết lộ 3 thuộc tính của hộp S, những thuộc tính này đảm bảo tính confusion và diffusion của thuật toán:

– Các bit vào luôn phụ thuộc không tuyến tính với các bit ra.

– Sửa đổi ở một bit vào làm thay đổi ít nhất là hai bit ra.

– Khi một bit vào được giữ cố định và 5 bit còn lại cho thay đổi thì hộp S thể hiện một tính chất được gọi là "phân bố đồng nhất": so sánh số lượng bit số 0 và 1 ở các đầu ra luôn ở mức cân bằng. Tính chất này khiến cho việc phân tích theo lý thuyết thống kê để tìm cách phá hộp S là vô ích.

Sau khi cộng modulo với khoá K, kết quả thu được chuỗi 48 bit chia làm 8 khối đưa vào 8 hộp S-Box. Mỗi hộp S-Box có 6 bit đầu vào và 4 bit đầu ra (tổng bộ nhớ yêu cầu cho 8 hộp S-Box chuẩn DES là 256 bytes). Kết quả thu được là một chuỗi 32 bit tiếp tục vào hộp P-Box.

Ta có thể xây dựng các hộp S của riêng mình, tuy nhiên cũng có thể dùng các hộp S chuẩn đã được công bố:

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Bảng 3.12: Hộp S_1

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Bảng 3.13: Hộp S_2

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	15	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Bảng 3.14: Hộp S_3

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9

10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Bảng 3.15: Hộp S_4

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Bảng 3.16: Hộp S_5

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Bảng 3.17: Hộp S_6

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Bảng 3.18: Hộp S_7

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Bảng 3.19: Hộp S_8

Ví dụ:

Giả sử đầu vào của hộp S_6 là chuỗi bit 110011 từ 31 đến 36 . Bit đầu tiên và bit cuối cùng kết hợp lại thành 11 tương ứng với hàng 3 của hộp S_6. Bốn bit giữa có giá trị 1001, tương ứng với cột 9. Như vậy, giá trị nhận được là 14 (số đếm của cột, hàng bắt đầu từ 0) và giá trị 1110 được thay thế cho giá trị 110110 ở đầu ra.

2.2.7. Hộp P-Box

Việc hoán vị này mang tính đơn ánh, nghĩa là một bit đầu vào sẽ cho một bit ở đầu ra, không bit nào được sử dụng hai lần hay bị bỏ qua. Hộp P-Box thực chất chỉ làm chức năng sắp xếp đơn thuần theo bảng sau:

Bảng mô tả hộp P-Box (P):

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Bảng 3.20: Bảng hoán vị P

Ví dụ như bit 21 sẽ dịch chuyển đến bit thứ 4, trong khi bit thứ 4 lại dịch chuyển đến bit 31. Kết quả cuối cùng của hộp P-Box lại được XOR với nửa trái của khối 64 bit của chính nó (tức Li-1 để tạo ra Ri) và sau đó nửa trái và nửa phải đảo cho nhau và bắt đầu một vòng khác.

2.2.8. Ví dụ về mã hoá DES

Để có thể hiểu rõ hơn về phương pháp mã hoá DES, chúng ta hãy xét ví dụ sau:

– Một bản rõ mang nội dung: "***0123456789ABCDEF***".

– Sử dụng khoá (ở dạng thập phân): "***133457799BBCDFFf***". Khoá này ở dạng nhị phân là một chuỗi bit như sau (không có bit kiểm tra):

0001001001101001010110111100100110110111101011111111000

– Chuyển đổi IP, chúng ta lấy ra L_0 và R_0:

L_0 = 11001100000000001100110011111111

$L_0 = R_0$ = 11110000101010101111000010101010

– 16 vòng mã hoá được thực hiện như sau:

$E(R_0)$	=	011110100001010101010101011110100001010101010101
K_1	=	000110110000001011101111111111000111000001110010
$E(R_0) \oplus K_1$	=	011000010001011110111010100001100110010100100111
Đầu ra S-Box	=	010111001000001010110101100101 11
$f(R_0,K_1)$	=	001000110100101010100110111011
$L_2=R_1$	=	11101111010010100110010101000100

$E(R_1)$	=	011101011110101001010100001100001010101010000 01001
K_2	=	011110011010111011011001110110111100100111100101
$E(R_1) \oplus K_2$	=	000011000100010010001101111010110110001111101100
Đầu ra S-Box	=	111110001101000000111010101011 10
$f(R_1,K_2)$	=	001111001010101110000111101000 11
$L_3=R_2$	=	11001100000000010111011100001001

$E(R_2)$	=	111001011000000000000010101110101110100001010011
K_3	=	010101011111110010001010010000101100111110011001
$E(R_2) \oplus K_3$	=	101100000111110010001000111110000010011111001010
Đầu ra S-Box	=	001001110001000011100001011011111
$f(R_2,K_3)$	=	010011010001011001101110101100000
$L_4=R_3$	=	101000100101110000001011111110100

$E(R_3)$	=	010100000100001011111000000001010111111110101001
K_4	=	011100101010110111010110110110110011010100011101
$E(R_3) \oplus K_4$	=	001000101110111100101110110111100100101010110100
Đầu ra S-Box	=	001000011110110110011111100111010
$f(R_3,K_4)$	=	101110110010001101110111010011100
$L_5=R_4$	=	011101110010001000000000001000101

$E(R_4)$	=	101110101110100100000100000000000000000001000001010
K_5	=	011111001110110000000111111010110101001110101000
$E(R_4) \oplus K_5$	=	110001100000001010000000111110101101010001101000010
Đầu ra S-Box	=	010100001100100000110001111010111
$f(R_4,K_5)$	=	001010000001001110101101110000011
$L_6=R_5$	=	100010100100111110100110001101111

$E(R_5)$	=	110001010100001001011111110100001100000110101111
K_6	=	011000111010010100111110010100000111101100101111
$E(R_5) \oplus K_6$	=	101001101110011101100001100000000101110101000000
Đầu ra S-Box	=	010000011111001101001100000111101
$F(R_5,K_6)$	=	100111100100010111001101001011100
$L_7=R_6$	=	111010010110011111001101011010011

$E(R_6)$	=	111101010010101100001111111001011010101101010011
K_7	=	111011001000010010110111111101100001100010111100
$E(R_6) \oplus K_7$	=	000110011010111110111000000100111011001111101111
Đầu ra S-Box	=	000100000111010101000000010101101
$F(R_6,K_7)$	=	100011000000010100011100000100111

$L_8=R_7$	=	00000110010010101011101000010000

$E(R_7)$	=	000000001100001001010101010111110100000010100000
K_8	=	111101111000101000111010110000010011101111111011
$E(R_7) \oplus K_8$	=	111101110100100001101111100111100111101101011011
Đầu ra S-Box	=	01101100000110000111110010101110
$F(R_7,K_8)$	=	00111100000011101000011011111001
$L_9=R_8$	=	11010101011010010100101110010000

$E(R_8)$	=	011010101010101101010010101001010111110010100001
K_9	=	111000001101101111101011111011011110011110000001
$E(R_8) \oplus K_9$	=	100010100111000010111001010010001001101100100000
Đầu ra S-Box	=	00010001000011000101011101110111
$F(R_8,K_9)$	=	00100010001101100111110001101010
$L_{10}=R_9$	=	00100100011111001100011001111010

$E(R_9)$	=	000100001000001111111001011000001100001111110100
K_{10}	=	101100011111001101000111101110100100011001001111
$E(R_9) \oplus K_{10}$	=	101000010111000010111110110110101000010110111011
Đầu ra S-Box	=	11011010000001000101001001110101
$F(R_9,K_{10})$	=	01100010101111001001110000100010
$L_{11}=R_{10}$	=	10110111110101011101011110110010

$E(R_{10})$	=	010110101111111010101011111010101111110110100101
K_{11}	=	001000010101111111010011110111101101001110000110
$E(R_{10}) \oplus K_{11}$	=	011110111010000101111000001101000010111000100011
Đầu ra S-Box	=	01110011000001011101000100000001
$f(R_{10},K_{11})$	=	11100001000001001111101000000010
$L_{12}=R_{11}$	=	11000101011110000011110001111000

$E(R_{11})$	=	011000001010101111110000000111111000001111110001
K_{12}	=	011101010111000111110101100101000110011111101001
$E(R_{11}) \oplus K_{12}$	=	000101011101101000000101100010111110010000011000

Đầu ra S-Box	=	01111011100010110010011000110101
$f(R_{11},K_{12})$	=	11000010011010001100111111101010
$L_{13}=R_{12}$	=	01110101101111010001100001011000

$E(R_{12})$	=	00111010101111011111101010001111000000101111 0000
K_{13}	=	10010111110001011101000111111010101110100100 0001
$E(R_{12}) \oplus K_{13}$	=	10101101011110000010101101110101101110001011 0001
Đầu ra S-Box	=	10011010110100011000101101001111
$f(R_{12},K_{13})$	=	11011101101110110010100100100010
$L_{14}=R_{13}$	=	00011000110000110001010101011010
$E(R_{13})$	=	00001111000101100000011010001010101010101111 0100
K_{14}	=	01011111010000111011011111110010111001110011 1010
$E(R_{13}) \oplus K_{14}$	=	01010000010101011011000101111000010011011100 1110
Đầu ra S-Box	=	01100100011110011001101011110001
$f(R_{13},K_{14})$	=	10110111001100011000111001010101
$L_{15}=R_{14}$	=	11000010100011001001011000001101

$E(R_{14})$	=	11100000010101000101100101001010110000000 1011011
K_{15}	=	10111111100100011000110100111101001111110000 1010
$E(R_{14}) \oplus K_{15}$	=	01011111110001011101010001110111111111110101 0001
Đầu ra S-Box	=	10110010111010001000110100111100
$f(R_{14},K_{15})$	=	01011011100000010010011101101110
$L_{16}=R_{15}$	=	01000011010000100011001000110100

$E(R_{15})$	=	00100000011010100000010000011010010000011010 1000
K_{16}	=	11001011001111011000101100001110000010111111 10101
$E(R_{15}) \oplus K_{16}$	=	11101011010101111000111100010100010101100101 1101
Đầu ra S-Box	=	10100111100000110010010000101001
$f(R_{15},K_{16})$	=	11001000110000000100111110011000
R_{16}	=	00001010010011001101100110010101

Bảng 3.21: Ví dụ về các bước thực hiện của DES

– Cuối cùng, chuyển đổi IP^{-1}, ta thu được bản mã (ở dạng Hecxa): "***85E813540F0AB405***".

2.3. Các yếu điểm của DES

2.3.1. Tính bù

Nếu ta ký hiệu $\bar{u}$ là phần bù của u (ví dụ như: 0100101 là phần bù của 1011010) thì DES có tính chất sau:

$$y = DES(x,k) \rightarrow \bar{y} = DES(\bar{x}, \bar{k})$$

Cho nên nếu ta biết mã y được mã hoá từ thông tin x với khoá K thì ta suy ra được bản mã $\bar{y}$ được mã hoá từ bản rõ $\bar{x}$ với khoá $\bar{k}$. Tính chất này chính là một yếu điểm của DES bởi vì qua đó đối phương có thể loại bỏ đi một số khoá phải thử khi tiến hành thử giải mã theo kiểu vét cạn.

2.3.2. Khoá yếu

Khoá yếu là các khoá mà theo thuật toán sinh khoá con thì tất cả 16 khoá con đều như nhau:

$K_1 = K_2 = ... = K_{15} = K_{16}$

Điều đó khiến cho việc mã hóa và giải mã đối với khoá yếu là giống hệt nhau.

Có tất cả 4 khoá yếu sau:

Khoá yếu (Hex)				C_0	D_0
0101	0101	0101	0101	$\{0\}^{28}$	$\{0\}^{28}$
FEFE	FEFE	FEFE	FEFE	$\{1\}^{28}$	$\{1\}^{28}$
1F1F	1F1F	0E0E	0E0E	$\{0\}^{28}$	$\{1\}^{28}$
E0E0	E0E0	F1F1	F1F1	$\{1\}^{28}$	$\{0\}^{28}$

Bảng 3.22: Các khóa yếu của DES

Đồng thời còn có 6 cặp khoá nửa yếu (semi-weak key) khác với thuộc tính như sau:

$$y = DES(x,k_1) \text{ và } y = DES(x,k_2)$$

nghĩa là với 2 khoá khác nhau nhưng mã hoá ra cùng một bản mã từ cùng một bản rõ:

C_0	D_0	Semi-weak key (Hex)								C_0	D_0
$\{01\}^{14}$	$\{01\}^{14}$	01FE	01FE	01FE	01FE	FE01	FE01	FE01	FE01	$\{10\}^{14}$	$\{10\}^{14}$
$\{01\}^{14}$	$\{10\}^{14}$	1FE0	1FE0	0EF1	0EF1	E01F	E01F	F10E	F10E	$\{10\}^{14}$	$\{01\}^{14}$
$\{01\}^{14}$	$\{0\}^{28}$	01E0	01E0	01F1	01F1	E001	E001	F101	F101	$\{10\}^{14}$	$\{0\}^{28}$
$\{01\}^{14}$	$\{1\}^{28}$	1FFE	1FFE	0EFE	0EFE	FE1F	FE1F	FE0E	FE0E	$\{10\}^{14}$	$\{1\}^{28}$
$\{0\}^{28}$	$\{01\}^{14}$	011F	011F	010E	010E	1F01	1F01	0E01	0E01	$\{0\}^{28}$	$\{10\}^{14}$
$\{1\}^{28}$	$\{01\}^{14}$	E0FE	E0FE	F1FE	F1FE	FEE0	FEE0	FEF1	FEF1	$\{1\}^{28}$	$\{10\}^{14}$

Bảng 3.23: Các khóa nửa yếu của DES

2.3.3. DES có cấu trúc đại số

Với 64 bit khối bản rõ có thể được ánh xạ lên tất cả vị trí của 64 bit khối bản mã trong 2^{64} cách. Trong thuật toán DES, với 56 bit khoá, có thể cho chúng ta 2^{56} (khoảng 10^{17}) vị trí ánh xạ. Với việc đa mã hoá thì không gian ánh xạ còn lớn hơn. Tuy nhiên điều này chỉ đúng nếu việc mã hoá DES là không có cấu trúc.

Với DES có cấu trúc đại số thì việc đa mã hoá sẽ được xem ngang bằng với việc đơn mã hoá. Ví dụ như có hai khoá bất kỳ K_1 và K_2 thì sẽ luôn được khoá thứ K_3 như sau:

$$E_{K2}(E_{K1}(x)) = E_{K3}(x)$$

Nói một cách khác, việc mã hoá DES mang tích chất "nhóm", đầu tiên mã hoá bản rõ bằng khoá K_1 sau đó là khoá K_2 sẽ giống với việc mã hoá ở khoá K_3. Điều này thực sự quan trọng nếu sử dụng DES trong đa mã hoá. Nếu một "nhóm" được phát với cấu trúc hàm quá nhỏ thì tính an toàn sẽ giảm.

2.3.4. Không gian khóa K

DES có $2^{56} = 10^{17}$ khoá. Nếu chúng ta biết được một cặp "tin/mã" thì chúng ta có thể thử tất cả 10^{17} khả năng này để tìm ra khoá cho kết quả khớp nhất. Giả sử như một phép thử mất 10^{-6}s, thì chúng sẽ mất 10^{11}s, tức 7300 năm. Nhưng với các máy tính được chế tạo theo xử lý song song. Chẳng hạn với 10^7 con chipset mã DES chạy song song thì bây giờ mỗi một con chipset chỉ phải chịu trách nhiệm tính toán với 10^{10} phép thử. Chipset mã DES ngày nay có thể xử lý tốc độ 4.5×10^7 bit/s tức có thể làm được hơn 10^5 phép mã DES trong một giây.

Vào năm 1976 và 1977, Diffie và Hellman đã ước lượng rằng có thể chế tạo được một máy tính chuyên dụng để vét cạn không gian khoá DES trong ½ ngày với cái giá 20 triệu đô la. Năm 1984, chipset mã hoá DES với tốc độ mã hoá 256000 lần/giây. Năm 1987, đã tăng lên 512000 lần/giây. Vào năm 1993, Michael Wiener đã thiết kế một máy tính chuyên dụng với giá 1 triệu đô la sử dụng phương pháp vét cạn để giải mã DES trung bình trong vòng 3,5 giờ (và chậm nhất là 7 giờ).

Đến năm 1990, hai nhà toán học người Do Thái - Biham và Shamir - đã phát minh ra phương pháp phá mã vi sai (diferential cryptanalyis), đây là một kỹ thuật sử dụng những phỏng đoán khác nhau trong bản rõ để đưa ra những thông tin trong bản mã. Với phương pháp này, Biham và Shamir đã chứng minh rằng nó hiệu quả hơn cả phương pháp vét cạn.

Phá mã vi sai là thuật toán xem xét những cặp mã hoá khác nhau, đây là những cặp mã hoá mà bản rõ của chúng là khác biệt. Người ta sẽ phân tích tiến trình biến đổi của những cặp mã này thông qua các vòng của DES khi chúng được mã hoá với cùng một khoá K. Sau đó sẽ chọn hai bản rõ khác nhau một cách ngẫu nhiên hợp lý nhất. Sử dụng sự khác nhau của kết quả mã hoá và gán cho những khoá khác nhau một cách phù hợp nhất. Khi phân tích nhiều hơn những cặp bản mã, chúng ta sẽ tìm ra một khoá được xem là đúng nhất.

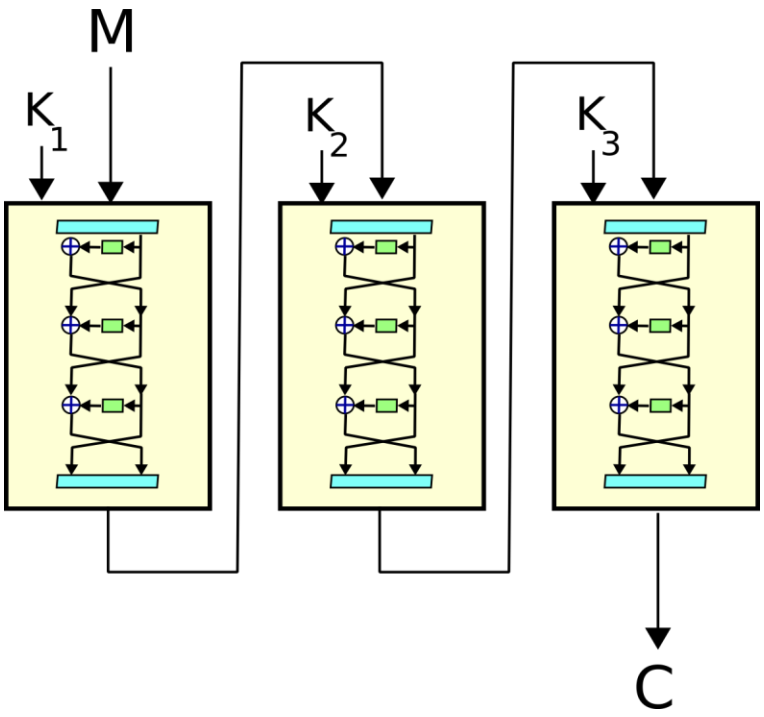
2.4. Triple DES (3DES)

Như đã trình bày ở các phần trên, hệ mã DES (hay chuẩn mã hóa dữ liệu) với không gian khóa vẻn vẹn có 2^{54} khóa nên thực tế hiện nay có thể bị thám mã trong khoảng thời gian vài giờ đồng hồ. Vì vậy việc tìm kiếm các hệ mã khác thay thế cho DES là một điều cần thiết. Một trong những cách thức được xem xét đầu tiên là tận dụng DES nhưng sử dụng mã hóa nhiều lần. Cách thứ nhất là sử dụng hai khóa để mã hóa hai lần như sau:

$$C = E_{K2}(E_{K1}(P))$$

Cách này gọi là double DES hay 2DES, khóa của hệ mã theo mô hình này là 112 bit, có vẻ an toàn hơn so với DES, ít nhất là trên nguyên tắc. Tuy nhiên các chứng minh về mặt lý thuyết (không nằm trong phạm vi của tài liệu này) đã cho thấy rằng hệ mã này không hề an toàn hơn DES (thuật toán thám mã theo kiểu vét cạn brute-force yêu cầu số phép tính gấp đôi để thám mã 2DES so với DES).

Cách thức thứ hai và hiện nay đang được sử dụng rộng rãi là mã hóa DES ba lần, cách này gọi là Triple DES (TDES) hay 3DES, hoặc một cách chuẩn mực hơn là TDEA (Triple Data Encryption Algorithm). Mô hình sử dụng đơn giản nhất của Triple DES là mã hóa 3 lần sử dụng 3 khóa K1, K2, K3 như hình minh họa sau:



Hình 3.7: Triple DES

Bản mã $C = DES_{K3}(DES_{K2}(DES_{K1}(M))$, mô hình này gọi là EEE vì cả ba bước sử dụng ba khóa ở đây đều sử dụng thuật toán mã hóa chuẩn của DES, một biến thể khác của mô hình này gọi là EDE với bước ở giữa sử dụng thuật toán giải mã của DES:

$$C = DES_{K3}(DES^{-1}_{K2}(DES_{K1}(M)).$$

Việc lựa chọn mã hóa hay giải mã ở bước thứ hai không làm thay đổi tính an toàn của Triple DES. Khóa của Triple DES là 168 bit, một số biến thể của Triple DES sử dụng

khóa có độ dài 112 bit (K1=K3) nhưng khác với double DES, khi đó phương pháp này có tên gọi là Two key Triple DES. Các chứng minh về mặt lý thuyết và các tấn công đối với Triple DES cho thấy hệ mã này vẫn sẽ còn được sử dụng trong một tương lai dài nữa mặc dù trên trên thực tế nó chậm hơn so với AES 6 lần.

2.5. Chuẩn mã hóa cao cấp AES

2.5.1. Giới thiệu

Chuẩn mã hóa dữ liệu cao cấp AES là một hệ mã khóa bí mật có tên là Rijndael (Do hai nhà mật mã học người Bỉ là Joan Daemen và Vincent Rijmen đưa ra và trở thành chuẩn từ năm 2002) cho phép xử lý các khối dữ liệu input có kích thước 128 bit sử dụng các khóa có độ dài 128, 192 hoặc 256 bit. Hệ mã Rijndael được thiết kế để có thể làm việc với các khóa và các khối dữ liệu có độ dài lớn hơn tuy nhiên khi được chọn là một chuẩn do Ủy ban tiêu chuẩn của Hoa Kỳ đưa ra vào năm 2001, nó được qui định chỉ làm việc với các khối dữ liệu 128 bit và các khóa có độ dài 128, 192 hoặc 256 bit (do đó còn đặt cho nó các tên AES-128, AES-192, AES-256 tương ứng với độ dài khóa sử dụng).

2.5.2. Các khái niệm và định nghĩa (Definitions)

2.5.2.1. Các khái niệm và ký hiệu

Các khái niệm và định nghĩa được sử dụng để trình bày về chuẩn mã hóa cao cấp:

AES	Chuẩn mã hóa cao cấp
Biến đổi Affine	Phép biến đổi bao gồm một phép nhân với một ma trận sau đó là một phép cộng của một vectơ
Bit	Một số nhị phân nhận giá trị 0 hoặc 1
Block	Một dãy các bit nhị phân tạo thành input, output, trạng thái (state) và các khóa sử dụng tại các vòng lặp (Round Key) của hệ mã. Độ dài của dãy (khối) là số lượng các bit mà nó chứa. Các khối cũng có thể được xem là một dãy các byte
Byte	Một nhóm 8 bit
Cipher	Thuật toán mã hóa
Cipher Key	Khóa của hệ mã, có thể được biểu diễn dưới dạng một mảng 2 chiều gồm 4 hàng và Nk cột
Ciphertext	Bản mã
Inverse Cipher	Thuật toán giải mã
Thủ tục sinh khóa (Key Expansion)	Thủ tục được sử dụng để sinh ra các khóa sử dụng tại các vòng lặp của thuật toán mã hóa, giải mã từ khóa chính ban đầu
Round Key	Là các giá trị sinh ra từ khóa chính bằng cách sử dụng thủ tục sinh khóa. Các khóa này được sử dụng tại các vòng lặp của thuật toán
Trạng thái (State)	Các giá trị mã hóa trung gian có thể biểu diễn dưới dạng một mảng 2 chiều gồm 4 hàng và Nb cột
S-box	Một bảng thế phi tuyến được sử dụng trong thủ tục sinh khóa và trong các biến đổi thay thế các byte để thực hiện các thay thế 1-1 đối với một giá trị 1 byte
Word	Một nhóm 32 bit có thể được xem như 1 đơn vị tính toán độc lập hoặc là một mảng 4 byte

Bảng 3.24: Qui ước một số từ viết tắt và thuật ngữ của AES

2.5.2.2. Các hàm, ký hiệu và các tham số của thuật toán

Các tham số thuật toán, các ký hiệu và các hàm được sử dụng trong mô tả thuật toán:

AddRoundKey()	Hàm biến đổi được sử dụng trong thuật toán mã hóa và giải mã trong đó thực hiện phép toán XOR bit giữa một trạng thái trung gian (State) và một khóa của vòng lặp (Round Key). Kích thước của một Round Key bằng kích thước của trạng thái (chẳng hạn với Nb = 4 độ dài của một Round Key sẽ là 128 bit hay 16 byte)
InvMixColumns()	Hàm biến đổi được sử dụng trong thuật toán giải mã, là hàm ngược của hàm MixColumns()
InvShiftRows()	Hàm biến đổi trong thuật toán giải mã, là hàm ngược của hàm ShiftRows()
InvSubBytes()	Hàm biến đổi trong thuật toán giải mã, là hàm ngược của hàm SubBytes()
K	Khóa mã hóa
MixColumns()	Hàm biến đổi trong thuật toán mã hóa nhận tất cả các cột của một trạng thái (State) và trộn với dữ liệu của nó (không phụ thuộc lẫn nhau) để nhận được một cột mới
Nb	Số lượng các cột (là các word 32 bit) tạo thành một trạng thái, Nb = 4)
Nk	Số lượng các word 32 bit tạo thành khóa mã hóa K (Nk = 4, 6, hoặc 8)
Nr	Số lượng các vòng lặp của thuật toán, là một hàm của Nk và Nb (là các giá trị cố định) (Nr = 10, 12 hoặc 14 tương ứng với các giá trị khác nhau của Nk)
Rcon[]	Mảng word hằng số sử dụng trong các vòng lặp
RotWord()	Hàm sử dụng trong thủ tục sinh khóa nhận một word 4-byte và thực hiện một hoán vị vòng
ShiftRows()	Hàm sử dụng trong quá trình mã hóa, xử lý các trạng thái bằng cách dịch vòng ba hàng cuối của trạng thái với số lần dịch khác nhau
SubBytes()	Hàm biến đổi sử dụng trong quá trình mã hóa, xử lý một trạng thái bằng cách sử dụng một bảng thế phi tuyến các byte (S-box) thao tác trên mỗi byte một cách độc lập
SubWord()	Hàm sử dụng trong thủ tục sinh khóa nhận một word input 4-byte và sử dụng một S-box trên mỗi giá trị 4-byte này để thu được 1 word output
XOR	Phép or bit tuyệt đối
$\oplus$	Phép or bit tuyệt đối
$\otimes$	Phép nhân 2 đa thức (bậc nhỏ hơn 4) theo modulo ($x^4 + 1$)
$\bullet$	Phép nhân trên trường hữu hạn

2.5.3. Các ký hiệu và qui ước

2.5.3.1. Input và Output

Input và Output của chuẩn mã hóa cao cấp đều là các dãy 128 bit, còn gọi là các khối (block), độ dài của mỗi khối này là số bit dữ liệu mà nó chứa. Khóa của chuẩn mã hóa cao cấp là một dãy có độ dài 128, 192 hoặc 256 bit. Chuẩn mã hóa dữ liệu cao cấp không làm việc với các giá trị input, output và khóa có các độ dài khác (mặc dù thuật toán cơ sở của nó cho phép điều này).

Các bit của input, output và khóa của hệ mã được đánh số từ 0.

2.5.3.2. Đơn vị Byte

Đơn vị cơ bản để xử lý trong AES là một byte tức là một dãy 8 bit được xem như là một đối tượng đơn. Các giá trị input, output và khóa của hệ mã (được qui định trong phần 3.1) được xem là một mảng các byte. Các giá trị input, output và khóa của hệ mã được ký hiệu bởi tên mảng a và biểu diễn dưới dạng a_n hoặc a[n] trong đó n nhận các giá trị trong các khoảng sau:

Nếu độ dài khóa bằng 128 bit: $0 \le n < 16$;

Nếu độ dài khóa bằng 192 bit: $0 \le n < 24$;

Nếu độ dài khóa bằng 256 bit: $0 \le n < 32$;

Tất cả các giá trị **Byte** sử dụng trong thuật toán của AES đều được biểu diễn dưới dạng một dãy các bit 0 hoặc 1 theo định dạng $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. Các Byte này sau được hiểu là các phần tử trên trường hữu hạn bằng cách sử dụng biểu diễn thành dạng đa thức:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0 = \sum_{i=0}^{7} b_i x^i .$$

Chẳng hạn giá trị {01100011} tương đương với phần tử trên trường hữu hạn $x^6 + x^5 + x + 1$.

Để thuận tiện, các giá trị Byte được biểu diễn sử dụng các ký hiệu của hệ Hexa, sử dụng 4 bit cho một ký tự và hai ký tự cho một Byte như bảng sau:

Bit	Ký tự
0000	0
0001	1
0010	2
0011	3

Bit	Ký tự
0100	4
0101	5
0110	6
0111	7

Bit	Ký tự
1000	8
1001	9
1010	a
1011	b

Bit	Ký tự
1100	c
1101	d
1110	e
1111	f

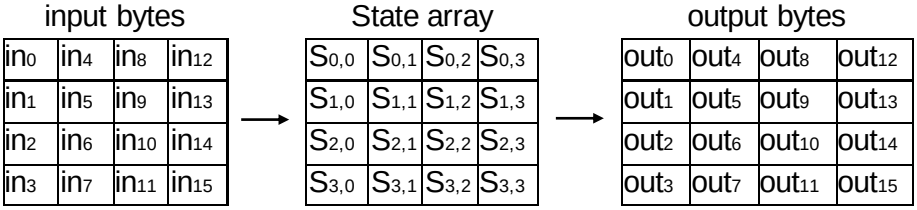
Bảng 3.25: Bảng biểu diễn các xâu 4 bit

Khi đó các Byte (8 bit) sẽ được biểu diễn bằng hai ký tự, chẳng hạn {01100011} sẽ được biểu diễn thành {63}.

2.5.3.4. Trạng thái (State)

Các thao tác bên trong của AES được thực hiện trên một mảng 2 chiều các byte được gọi là trạng thái. Một trạng thái gồm bốn hàng các byte, mỗi hàng có Nb byte trong đó Nb là kích thước của khối chia cho 32. Mảng trạng thái ký hiệu là s trong đó mỗi byte của mảng có 2 chỉ số hàng r và cột c ($0 \le r, c < 4$).

Tại thời điểm bắt đầu input của thuật toán – mảng các byte in_0, in_1, …, in_{15} được copy vào mảng trạng thái theo qui tắc được minh họa bằng hình vẽ:



Hình 3.8: Các trạng thái của AES

trong đó các giá trị của mảng s và mảng output được tính như sau:

$s[r, c] = in[r + 4c] \forall\ 0 \le r, c < 4$

$out[r + 4c] = s[r, c] \forall\ 0 \le r, c < 4$

2.5.3.5. Biểu diễn của trạng thái

Bốn cột của mảng trạng thái của thuật toán tạo thành 4 word 32-bit w_0, w_1, …, w_3 được biểu diễn như sau:

$w_0 = s_{0,0}\ s_{1,0}\ s_{2,0}\ s_{3,0}$ $w_1 = s_{0,1}\ s_{1,1}\ s_{2,1}\ s_{3,1}$

$w_2 = s_{0,2}\ s_{1,2}\ s_{2,2}\ s_{3,2}$ $w_3 = s_{0,3}\ s_{1,3}\ s_{2,3}\ s_{3,3}$

2.5.4. Thuật toán

Độ dài của input, output và các trạng thái (state) của chuẩn mã hóa cao cấp AES là 128 bit tương ứng với giá trị của Nb = 4 (là số lượng các word 32-bit và cũng là số cột của mỗi trạng thái). Khóa của AES có độ dài là 128, 192 hoặc 256 bit tương ứng với các giá trị của Nk là 4, 6, hoặc 8 và cũng là số cột của khóa mã hóa.

Tương ứng với độ dài của khóa sử dụng số vòng lặp của thuật toán Nr nhận các giá trị 10 (Nk = 4), 12 (Nk = 6) hoặc 14 (Nk = 8). Chúng ta có thể minh họa qua bảng sau:

	Độ dài khóa (Nk)	Kích thước khối (Nb)	Số lần lặp (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Bảng 3.26: Bảng độ dài khóa của AES

Cả quá trình mã hóa và giải mã AES sử dụng một hàm lặp là kết hợp của bốn hàm biến đổi (đơn vị xử lý là byte) sau: 1) biến đổi thay thế byte sử dụng một bảng thế (S-box), 2) dịch các hàng của mảng trạng thái với số lần dịch của mỗi hàng là khác nhau, 3) kết hợp dữ liệu của mỗi cột trong mảng trạng thái và 4) cộng một khóa Round Key vào trạng thái. Các biến đổi này (và các hàm ngược của chúng) được mô tả trong các phần 4.1.1-4.1.4 và 4.3.1-4.3.4.

2.5.4.1. Thuật toán mã hóa

Bắt đầu thuật toán bản rõ (input) được copy vào mảng trạng thái sử dụng các qui ước được mô tả trong phần 3.4. Sau khi cộng với khóa Round Key khởi tạo mảng trạng thái được biến đổi bằng các thực hiện một hàm vòng (round function) Nr lần (10, 12, hoặc 14 phụ thuộc vào độ dài khóa) trong đó lần cuối cùng thực hiện khác các lần trước đó. Trạng thái sau lần lặp cuối cùng sẽ được chuyển thành output của thuật toán theo qui tắc được mô tả trong phần 3.4.

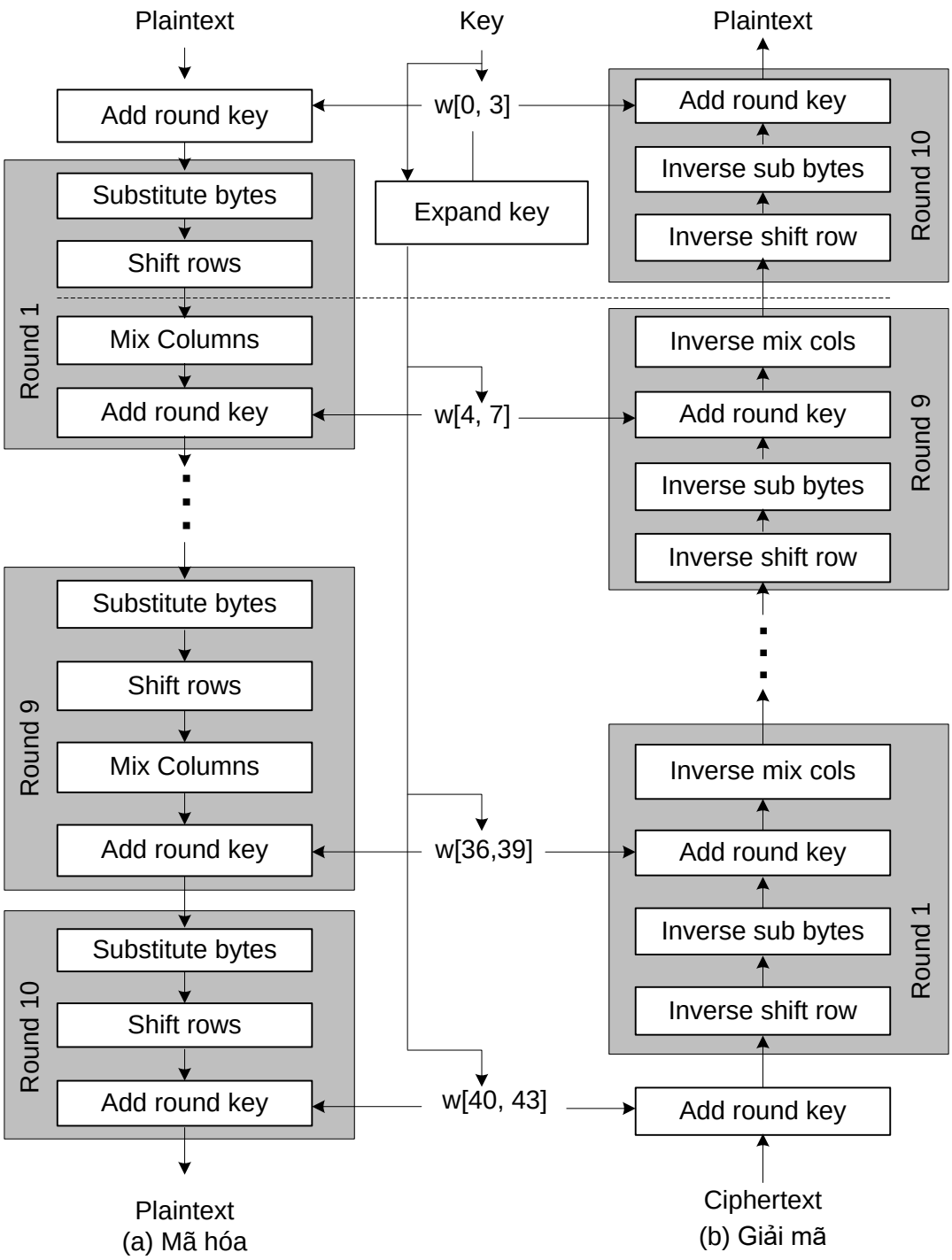
Hàm vòng được tham số hóa sử dụng một (key schedule) dãy các khóa được biểu diễn như là một mảng 1 chiều của các word 4-byte được sinh ra từ thủ tục sinh khóa (Key Expansion) được mô tả trong phần 5.2.

Chúng ta có thể thấy tất cả các vòng đều thực hiện các công việc giống nhau dựa trên 4 hàm (theo thứ tự) SubBytes(), ShiftRows(), MixColumns() và AddRoundKey() trừ vòng cuối cùng bỏ qua việc thực hiện hàm MixColumns().

Thuật toán được mô tả chi tiết qua đoạn giả mã lệnh sau:

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
        byte state[4,Nb]
        state = in
        AddRoundKey(state, w[0, Nb-1]) // See Sec. 5.1.4
        for round = 1 step 1 to Nr–1
                SubBytes(state) // See Sec. 5.1.1
                ShiftRows(state) // See Sec. 5.1.2
                MixColumns(state) // See Sec. 5.1.3
                AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
        end for
        SubBytes(state)
        ShiftRows(state)
        AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
        out = state
end
```

Sơ đồ thuật toán:



Hình 3.9: Thuật toán mã hóa và giải mã của AES

2.5.4.1.1 Hàm SubBytes()

Hàm SubBytes() thực hiện phép thay thế các byte của mảng trạng thái bằng cách sử dụng một bảng thế S-box, bảng thế này là khả nghịch và được xây dựng bằng cách kết hợp hai biến đổi sau:

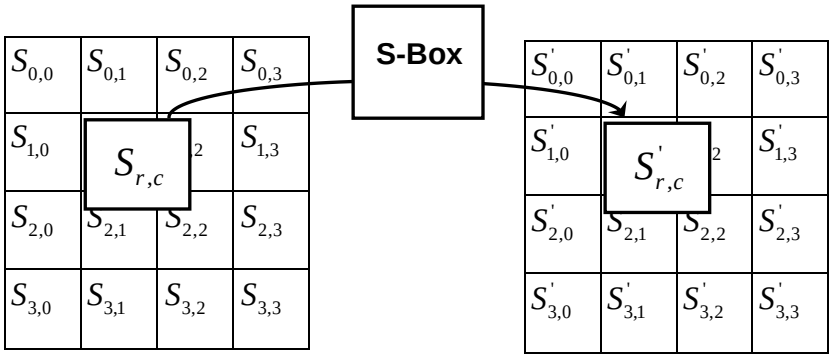
1. Nhân nghịch đảo trên trường hữn hạn GF(2^8) (mô tả trong phần 4.2), phần tử {00} được ánh xạ thành chính nó
2. Áp dụng biến đổi Affine sau (trên GF(2)):

$b_i^{'} = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$ trong đó 0 ≤ i <8 là bit thứ i của byte b tương ứng và c_i là bit thứ i của byte c với giá trị {63} hay {01100011}.

Các phần tử biến đổi affine của S-box có thể được biểu diễn dưới dạng ma trận như sau:

$$\begin{bmatrix} b_0^{'} \\ b_1^{'} \\ b_2^{'} \\ b_3^{'} \\ b_4^{'} \\ b_5^{'} \\ b_6^{'} \\ b_7^{'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Hình sau minh họa kết quả của việc áp dụng hàm biến đổi SubBytes() đối với mảng trạng thái:



Bảng thế S-box được sử dụng trong hàm SubBytes() có thể được biểu diễn dưới dạng hexa như sau:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Bảng 3.27: Bảng thế S-Box của AES

trong đó chẳng hạn nếu $s_{1,1}$ = {53} có nghĩa là giá trị thay thế sẽ được xác định bằng giao của hàng có chỉ số 5 với cột có chỉ số 3 trong bảng trên điều này tương ứng với việc $s'_{1,1}$ = {ed}.

2.5.4.1.2. Hàm ShiftRows()

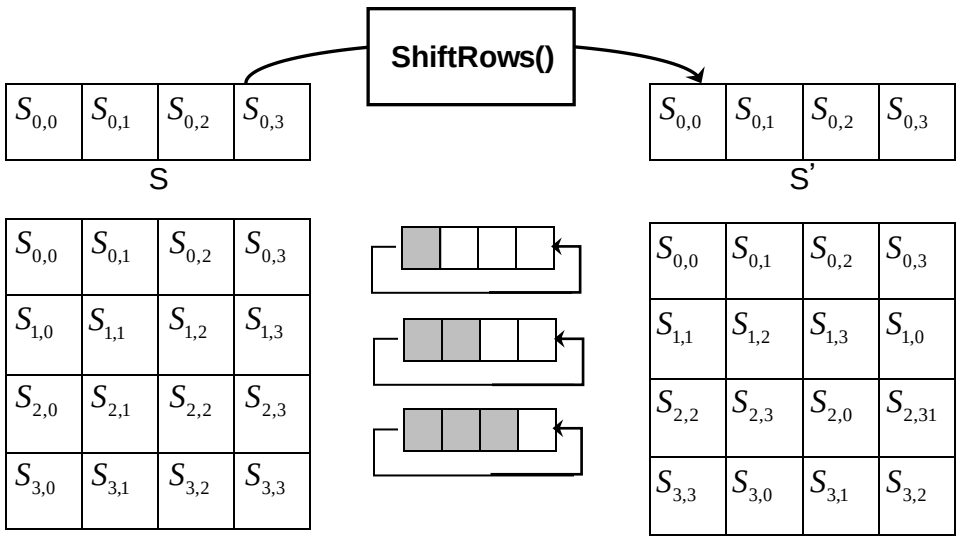
Trong hàm này các byte trong 3 hàng cuối của mảng trạng thái sẽ được dịch vòng với số lần dịch (hay số byte bị dịch) khác nhau. Hàng đầu tiên r = 0 không bị dịch.

Cụ thể hàm này sẽ tiến hành biến đổi sau:

$s'_{r,c} = s_{r,(c+shift(r,Nb)) \bmod Nb} (Nb = 4)$ trong đó giá trị dịch shift(r, Nb) phụ thuộc vào số hàng r như sau:

shift(1, 4) = 1, shift(2, 4) = 2, shift(3, 4) = 3.

Thao tác này sẽ chuyển các byte tới các vị trí thấp hơn trong các hàng, trong khi các byte thấp nhất sẽ được chuyển lên đầu của hàng. Tất các các mô tả trên có thể minh họa qua hình vẽ sau:



Hình 3.10: Hàm ShifftRows()

2.5.4.1.3. Hàm MixColumns()

Hàm này làm việc trên các cột của bảng trạng thái, nó coi mỗi cột của mảng trạng thái như là một đa thức gồm 4 hạng tử như được mô tả trong phần 4.3. Các cột sẽ được xem như là các đa thức trên GF(2^8) và được nhân theo modulo $x^4 + 1$ với một đa thức cố định a(x):

$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

Như đã mô tả trong phần 4.3 điều này có thể biểu diễn bằng một phép nhân ma trận:

s'(x) = a(x)⊗s(x):

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

với mọi $0 \le c < Nb = 4$.

Kết quả là bốn byte trong mỗi cột sẽ được thay thế theo công thức sau:

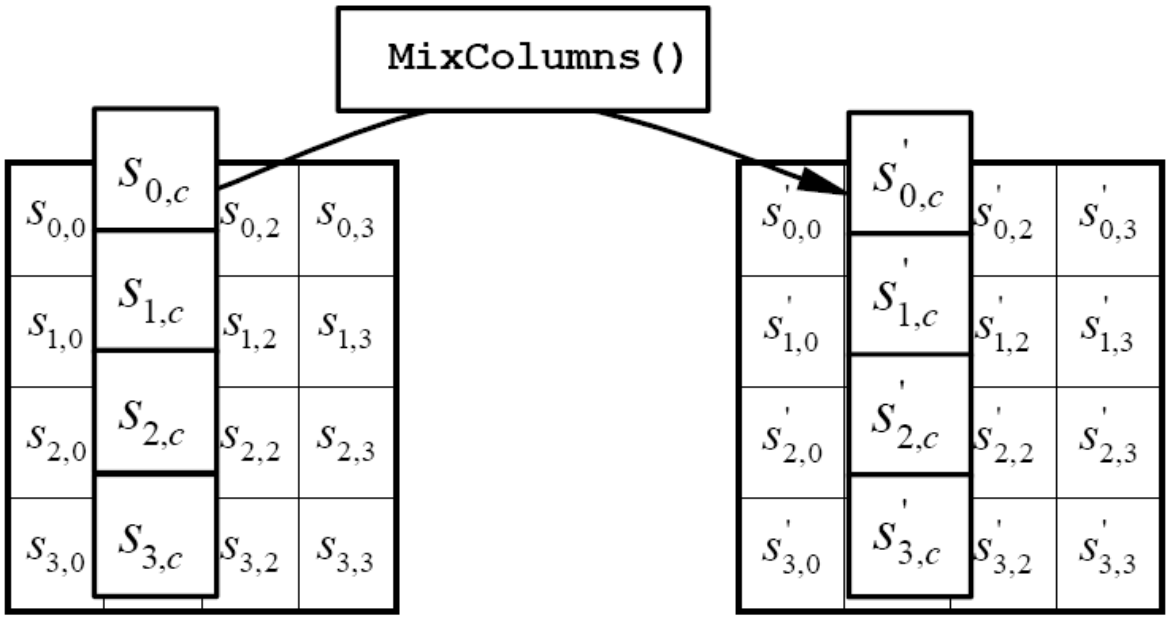
$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Có thể minh họa việc thực hiện của hàm này bằng hình vẽ sau:



Hình 3.11: Hàm MixColumns của AES

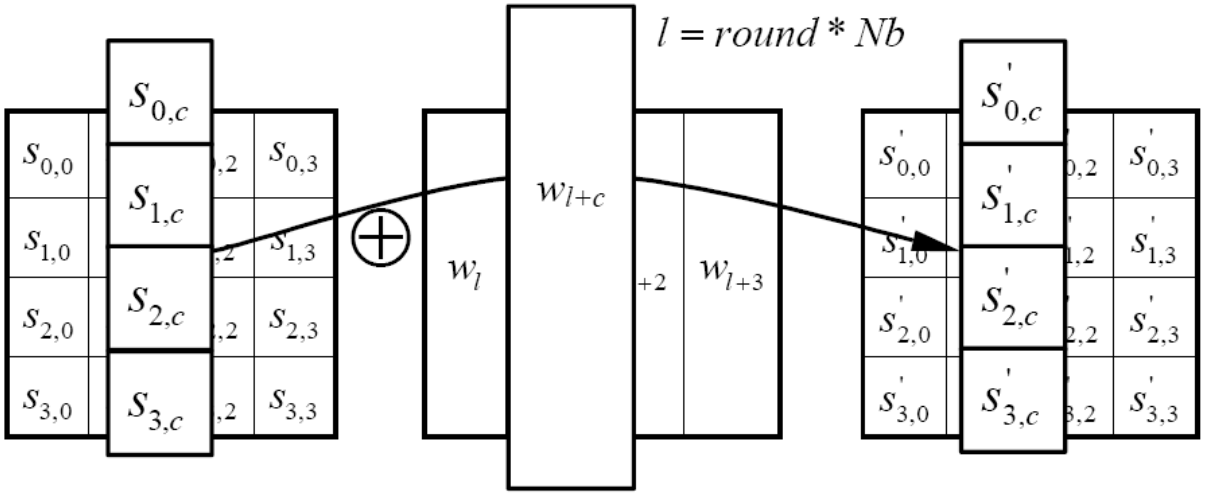
2.5.4.1.4. Hàm AddRoundKey()

Trong hàm này một khóa vòng (Round Key) sẽ được cộng vào mảng trạng thái bằng một thao tác XOR bit. Mỗi khóa vòng gồm Nb word được sinh ra bởi thủ tục sinh khóa (phần 5.2). Các word này sẽ được cộng vào mỗi cột của mảng trạng thái như sau:

$$\left[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c} \right] = \left[s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c} \right] \oplus \left[w_{round*Nb+c} \right] \forall 0 \le c \le Nb = 4$$

trong đó [w_i] là các word của khóa được mô tả trong phần 5.2 và round là lần lặp tương ứng với qui ước 0 ≤ round ≤ Nr. Trong thuật toán mã hóa phép cộng khóa vòng khởi tạo xảy ra với round = 0 trước khi các vòng lặp của thuật toán được thực hiện. Hàm AddRoundKey() được thực hiện trong thuật toán mã hóa khi 1 ≤ round ≤ Nr.

Việc thực hiện của hàm này có thể minh họa qua hình vẽ tring đó l = round * Nb. Địa chỉ byte trong các word của dãy khóa được mô tả trong phần 3.1.



Hình 3.12: Hàm AddRoundKey của AES

2.5.4.2. Thuật toán sinh khóa (Key Expansion)

Thuật toán sinh khóa của AES nhận một khóa mã hóa K sau đó thực hiện một thủ tục sinh khóa để sinh một dãy các khóa cho việc mã hóa. Thủ tục này sẽ sinh tổng số Nb*(Nr+1) word, thủ tục sử dụng một tập khởi tạo Nb word và mỗi một lần lặp trong số Nr lần sẽ cần tới Nb word của dữ liệu khóa. Dãy khóa kết quả là một mảng tuyến tính các word 4-byte được ký hiệu là [w_i] trong đó 0 ≤ i < Nb(Nr+1).

Sự mở rộng khóa thành dãy khóa được mô tả qua đoạn giả mã sau:

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
  word temp
  i = 0
  while (i < Nk)
```

```
        w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while
    i = Nk
    while (i < Nb * (Nr+1)]
        temp = w[i-1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
        else if (Nk > 6 and i mod Nk = 4)
            temp = SubWord(temp)
        end if
        w[i] = w[i-Nk] xor temp
        i = i + 1
    end while
end
```

SubWord() là một hàm nhận một input 4-byte và áp dụng bảng thế S-box lên input để nhận được một word output. Hàm RotWord() nhận một word input [a_0, a_1, a_2, a_3] thực hiện một hoán vị vòng và trả về [a_1, a_2, a_3, a_0]. Các phần tử của mảng hằng số Rcon[i] chứa các giá trị nhận được bởi [x^{i-1}, {00}, {00}, {00}] trong đó x^{i-1} là mũ hóa của x (x được biểu diễn dưới dạng {02} trên GF(2^8) và i bắt đầu từ 1).

Theo đoạn giả mã trên chúng ta có thể nhận thấy rằng Nk word của khóa kết quả sẽ được điền bởi khóa mã hóa. Các word sau đó w[i] sẽ bằng XOR với word đứng trước nó w[i-1] với w[i-Nk]. Với các word ở vị trí chia hết cho Nk một biến đổi sẽ được thực hiện với w[i-1] trước khi thực hiện phép XOR bit, sau đó là phép XOR với một hằng số Rcon[i]. Biến đổi này gồm một phép dịch vòng các byte của một word (RotWord()), sau đó là áp dụng một bảng tra lên tất cả 4 byte của word (SubWord()).

Chú ý là thủ tục mở rộng khóa đối với các khóa có độ dài 256 hơi khác so với thủ tục cho các khóa có độ dài 128 hoặc 192. Nếu Nk = 8 và i – 4 là một bội số của Nk thì SubWord() sẽ được áp dụng cho w[i-1] trước khi thực hiện phép XOR bit.

2.5.4.3. Thuật toán giải mã

Thuật toán giải mã khá giống với thuật toán mã hóa về mặt cấu trúc nhưng 4 hàm cơ bản sử dụng là các hàm ngược của các hàm trong thuật toán giải mã. Đoạn giả mã cho thuật toán giải mã như sau:

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
  byte state[4,Nb]
  state = in
```

```
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1]) // See Sec. 5.1.4
    for round = Nr-1 step -1 downto 1
        InvShiftRows(state) // See Sec. 5.3.1
        InvSubBytes(state) // See Sec. 5.3.2
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
        InvMixColumns(state) // See Sec. 5.3.3
    end for
    InvShiftRows(state)
    InvSubBytes(state)
    AddRoundKey(state, w[0, Nb-1])
    out = state
end
```

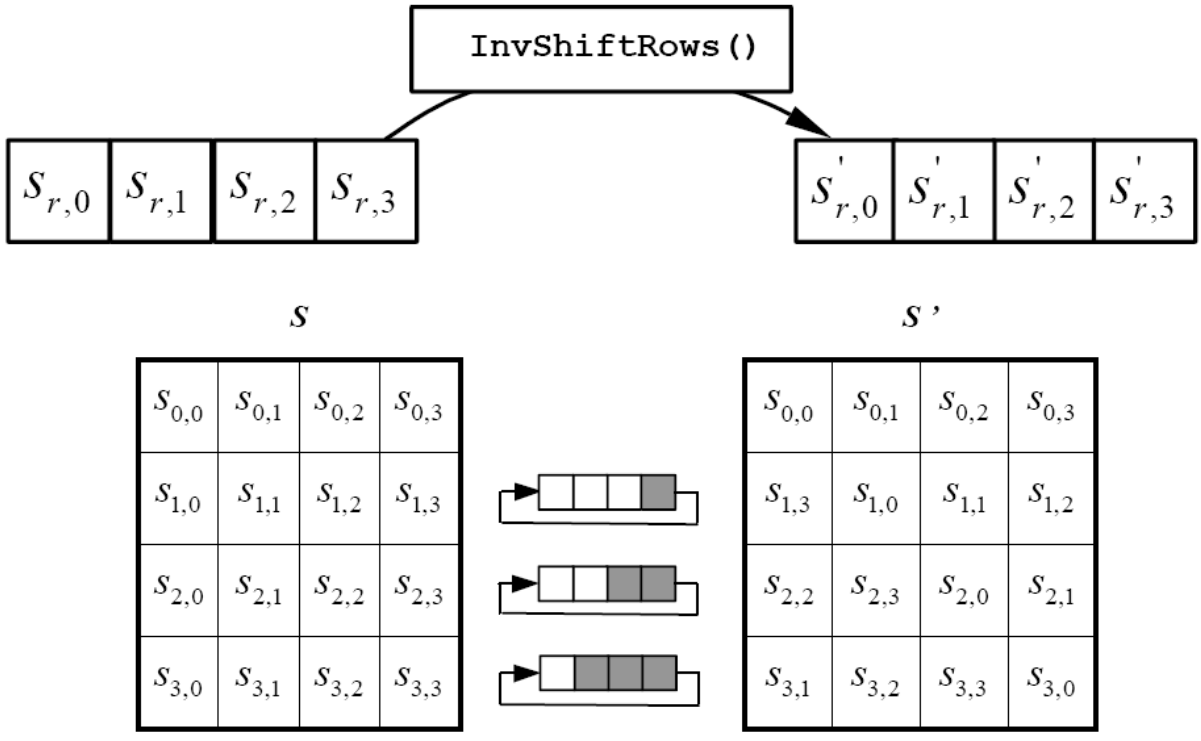
2.5.4.3.1. Hàm InvShiftRows()

Hàm này là hàm ngược của hàm ShiftRows() . Các byte của ba hàng cuối của mảng trạng thái sẽ được dịch vòng với các vị trí dịch khác nhau. Hàng đầu tiên không bị dịch, ba hàng cuối bị dịch đi Nb – shift(r, Nb) byte trong đó các giá trị shift(r, Nb) phụ thuộc vào số hàng như trong phần 5.1.2.

Cụ thể hàm này tiến hành xử lý sau:

$$s'_{r,(c+shift(r,Nb))\bmod Nb} = s_{r,c} \forall 0 < r < 4, 0 \le c < Nb(Nb = 4)$$

Hình minh họa:



Hình 3.13: Hàm InvShiftRows() của AES

2.5.4.3.2. Hàm InvSubBytes()

Hàm này là hàm ngược của hàm SubBytes(), hàm sử dụng nghịch đảo của biến đổi Affine bằng cách thực hiện nhân nghịch đảo trên GF(2^8).

Bảng thế được sử dụng trong hàm là:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Bảng 3.28: Bảng thế cho hàm InvSubBytes()

2.5.4.3.3. Hàm InvMixColumns()

Hàm này là hàm ngược của hàm MixColumns(). Hàm làm việc trên các cột của mảng trạng thái, coi mỗi cột như là môtô đa thức 4 hạng tử được mô tả trong phần 4.3. Các cột được xem là các đa thức trên GF(2^8) và được nhân theo modulo x^4+1 với một đa thức cố định là $a^{-1}(x)$:

$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$

Và có thể mô tả bằng phép nhân ma trận như sau:

$s'(x) = a^{-1}(x) \otimes s(x)$:

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

trong đó $0 \le c < Nb$.

Kết quả là bốn byte trong mỗi cột sẽ được thay thế theo công thức sau:

$$s'_{0,c} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$

$$s'_{1,c} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

2.5.4.3.4. Hàm nghịch đảo của hàm AddRoundKey()

Thật thú vị là hàm này tự bản thân nó là nghịch đảo của chính nó là do hàm chỉ có phép toán XOR bit.

2.5.4.3.5. Thuật toán giải mã tương đương

Trong thuật toán giải mã được trình bày ở trên chúng ta thấy thứ tự của các hàm biến đổi được áp dụng khác so với thuật toán mã hóa trong khi dạng của danh sách khóa cho cả 2 thuật toán vẫn giữ nguyên. Tuy vậy một số đặc điểm của AES cho phép chúng ta có một thuật toán giải mã tương đương có thứ tự áp dụng các hàm biến đổi giống với thuật toán mã hóa (tất nhiên là thay các biến đổi bằng các hàm ngược của chúng). Điều này đạt được bằng cách thay đổi danh sách khóa.

Hai thuộc tính sau cho phép chúng ta có một thuật toán giải mã tương đương:

1. Các hàm SubBytes() và ShiftRows() hoán đổi cho nhau; có nghĩa là một biến đổi SubBytes() theo sau bởi một biến đổi ShiftRows() tương đương với một biến đổi ShiftRows() theo sau bởi một biến đổi SubBytes(). Điều này cũng đúng với các hàm ngược của chúng

2. Các hàm trộn cột – MixColumns() và InvMixColumns() là các hàm tuyến tính đối với các cột input, có nghĩa là:

InvMixColumns(state XOR Round Key) = InvMixColumns(state) XOR InvMixColumns(Round Key).

Các đặc điểm này cho phép thứ tự của các hàm InvSubBytes() và InvShiftRows() có thể đổi chỗ. Thứ tự của các hàm AddRoundKey() và InvMixColumns() cũng có thể đổi chỗ miễn là các cột của danh sách khóa giải mã phải được thay đổi bằng cách sử dụng hàm InvMixColumns().

Thuật toán giải mã tương đương được thực hiện bằng cách đảo ngược thứ tự của hàm InvSubBytes() và InvShiftRows(), và thay đổi thứ tự của AddRoundKey() và InvMixColumns() trong các lần lặp sau khi thay đổi khóa cho giá trị round = 1 to Nr-1 bằng cách sử dụng biến đổi InvMixColumns(). Các word đầu tiên và cuối cùng của danh sách khóa không bị thay đổi khi ta áp dụng phương pháp này.

Thuật toán giải mã tương đương cho một cấu trúc hiệu quả hơn so với thuật toán giải mã trước đó.

Đoạn giả mã cho thuật toán giải mã tương đương:

EqInvCipher(byte in[4*Nb], byte out[4*Nb], word dw[Nb*(Nr+1)])

begin

```
    byte state[4,Nb]
    state = in
    AddRoundKey(state, dw[Nr*Nb, (Nr+1)*Nb-1])
    for round = Nr-1 step -1 downto 1
        InvSubBytes(state)
        InvShiftRows(state)
        InvMixColumns(state)
        AddRoundKey(state, dw[round*Nb, (round+1)*Nb-1])
    end for
    InvSubBytes(state)
    InvShiftRows(state)
    AddRoundKey(state, dw[0, Nb-1])
    out = state
  end
```

Các thay đổi sau cần thực hiện trong thuật toán sinh khóa để thuật toán trên có thể hoạt động được:

```
for i = 0 step 1 to (Nr+1)*Nb-1
  dw[i] = w[i]
end for
for round = 1 step 1 to Nr-1
  InvMixColumns(dw[round*Nb, (round+1)*Nb-1]) // note change of type
end for
```

2.6. Các cơ chế, hình thức sử dụng của mã hóa khối (Mode of Operation)

2.6.1. Các hình thức sử dụng

Như chúng ta đã biết các mã hóa khối mã hóa các khối thông tin có độ dài cố định, chẳng hạn DES với các khối bit 64, sử dụng khóa là xâu bít có độ dài bằng 56. Tuy nhiên để sử dụng các hệ mã này trên thực tế vẫn cần có một qui định về qui cách sử dụng chúng để mã hóa các dữ liệu cần mã hóa. Cách thức sử dụng một thuật toán mã hóa khối trong thực tế được gọi là Mode of Use hay Mode Of Operation. Có 4 hình thức sử dụng các hệ mã khối được định nghĩa trong các chuẩn ANSI (ví dụ ANSI X3.106-1983 dành cho DES). Dựa vào việc xử lý dữ liệu input của hệ mã người ta chia thành hai loại cơ chế sử dụng các hệ mã khối sau:

1. **Các chế độ khối** (Block Mode): xử lý các thông điệp theo các khối (ECB, CBC)
2. **Các chế độ luồng, dòng** (Stream Modes): xử lý các thông điệp như là một luồng bit/byte (CFB, OFB).

Các chế độ khối thường được sử dụng để mã hóa các dữ liệu mà chúng ta biết trước về vị trí, độ lớn trước khi mã hóa (chẳng hạn như các file, các email trước khi cần gửi đi) trong khi các chế độ luồng thường được sử dụng cho việc mã hóa các dữ liệu không được biết trước về độ lớn cũng như vị trí chẳng hạn như các tín hiệu gửi về từ vệ tinh hoặc các tín hiệu do một bộ cảm biến đọc từ bên ngoài vào.

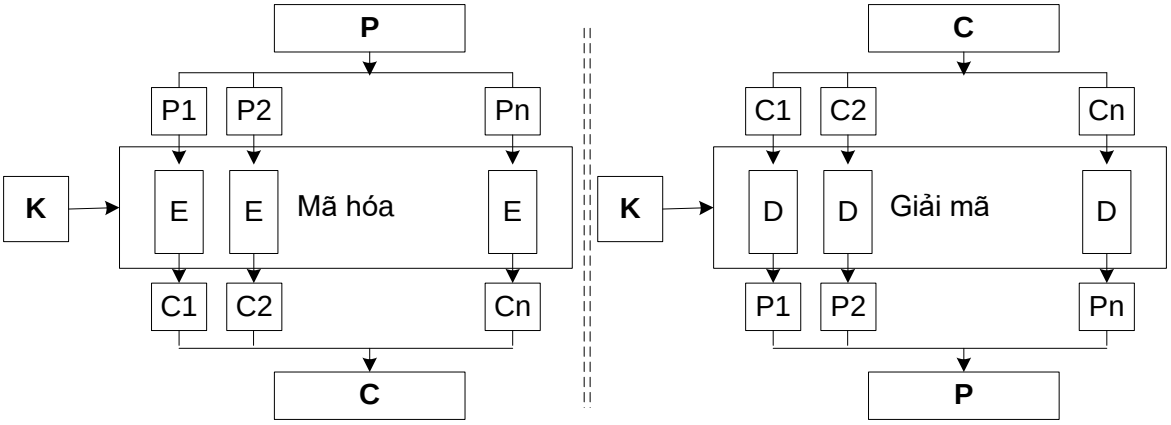
Chú ý: *DES, 3DES, AES (hay bất kỳ một thuật toán mã hóa khối nào khác) tạo thành một khối xây dựng cơ bản. Tuy nhiên để sử dụng chúng trong thực tế, chúng ta thường cần làm việc với các khối lượng dữ liệu không thể biết trước được, có thể chúng là một khối dữ liệu sẵn sàng ngay cho việc mã hóa(khi đó việc sử dụng mã hóa theo cơ chế khối là phù hợp), hoặc có thể chỉ được một vài bit, byte tại một thời điểm (khi đó sử dụng chế độ dòng là phù hợp). Vì thế các cơ chế sử dụng mã khối được trình bày trong phần này là riêng cho DES nhưng cũng được áp dụng tương tự cho các hệ mã khối khác.*

2.6.2. Cơ chế bảng tra mã điện tử ECB (Electronic CodeBook Book)

Thông điệp cần mã hóa được chia thành các khối độc lập để mã hóa, mỗi khối bản mã là kết quả của việc mã hóa riêng biệt khối bản rõ tương ứng với nó và độc lập với khối khác. Cách làm việc này giống như chúng ta thay thế các khối bản mã bằng các khối bản rõ tương ứng nên có tên gọi là bảng tra mã điện tử.

$P = P_1P_2 \ldots P_N$

Mã hóa: $C_i = DES_K(P_i)$, kết quả bản mã là $C = C_1C_2..C_N$. Quá trình giải mã tiến hành ngược lại: $P_i = DES^{-1}_K(C_i)$.



Hình 3.14: Cơ chế ECB

ECB là chế độ sử dụng đơn giản và dễ cài đặt nhất, được sử dụng khi chỉ một khối đơn thông tin cần được gửi đi (chẳng hạn như một khóa session được mã hóa bằng cách dùng một khóa chính).

Do trong ECB các khối bản rõ được mã hóa độc lập nên làm nảy sinh một số nhược điểm sau: các lặp lại của thông điệp có thể được thể hiện trên bản mã, nghĩa là nếu có các bản rõ giống nhau thì tương ứng các bản mã giống nhau, điều này đặc biệt thể hiện rõ với các dữ liệu lặp lại nhiều chẳng hạn như các dữ liệu hình ảnh. Việc để lộ tính lặp lại của bản rõ có thể dẫn tới các tấn công theo phương pháp phân tích thống kê. Hơn nữa các bản mã có thể bị giả mạo bằng cách thêm một số khối bản mã giả vào kết quả mã hóa, bên nhận sẽ không phát hiện ra sự giả mạo này. Bên cạnh đó việc mã hóa các khối

thông điệp là độc lập làm suy yếu DES. Trên thực tế ECB chỉ thực sự có ích khi gửi một khối dữ liệu nhỏ.

2.6.3. Cơ chế mã móc xích CBC - Cipher Block Chaining

Để vượt qua các vấn đề về sự lặp lại và yêu cầu độc lập trong ECB, chúng ta cần một vài cách để làm cho bản mã phụ thuộc vào tất cả các khối trước nó. Đó này chính là điều mà CBC cung cấp cho chúng ta bằng cách kết hợp khối bản rõ trước với khối thông điệp hiện tại trước khi mã hóa.

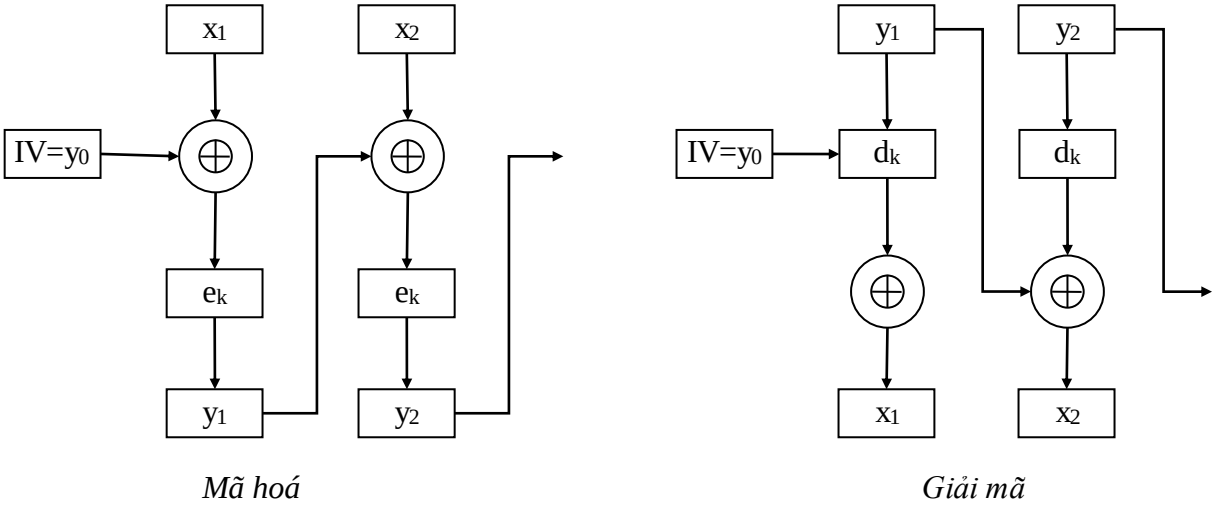
Cũng giống như cơ chế EBC trong cơ chế CBC bản rõ sẽ được chia thành các khối nhưng sẽ được liên kết với nhau trong quá trình mã hóa để tạo thành bản rõ. Chính vì các khối bản mã được móc xích với bản rõ và vì thế chế độ này có tên là CBC

CBC sử dụng một vector khởi tạo IV (Initial Vector) để bắt đầu:

$C_0 = IV,\ P = P_1P_2..P_N$

Mã hóa: $C_i = DES_K\ (P_i \oplus C_{i-1}),\ C = C_1C_2..C_N$

Giải mã: $P_i = DES^{-1}{}_K(C_i) \oplus C_{i-1},\ P = P_1P_2..P_N$.



Hình 3.15: Chế độ CBC

Chế độ CBC phù hợp với các yêu cầu cần gửi các lượng lớn dữ liệu một cách an toàn (chẳng hạn như FTP, EMAIL, WEB)

Trong CBC mỗi khối bản mã là phụ thuộc vào tất cả các khối thông điệp đứng trước đó nên việc sai lệch một khối bản rõ hoặc bản mã nào đó cũng làm sai lệch kết quả mã hóa và giải mã tương ứng. Khó khăn nhất trong việc sử dụng CBC chính là quản lý các giá trị IV sử dụng, thường thì cả hai bên nhận và gửi đều biết (chẳng hạn như bằng 0) hoặc sẽ được khởi tạo bằng các giá trị mới và gửi cho bên nhận trước khi mã hóa. Tuy nhiên nếu IV bị tiết lộ kẻ tấn công có thể làm thay đổi các bit ở khối đầu tiên, vì thế có thể IV là một giá trị cố định hoặc được gửi đi sau khi đã mã hóa bằng ECB.

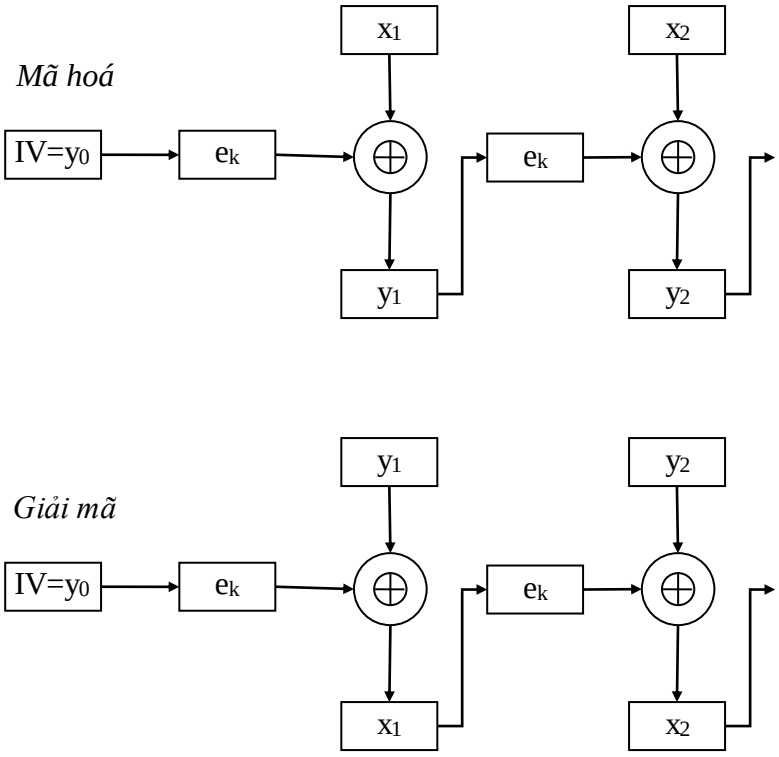
2.6.4. Chế độ mã phản hồi CFB (Cipher Feedback) và chế độ mã phản hồi đầu ra OFB (Output Feedback)

Các chế độ luồng CFB và OFB được sử dụng để mã hóa các dữ liệu được cung cấp rời rạc, thường là các tín hiệu nhận được từ vệ tinh hoặc do một bộ cảm biến nào đó

truyền về. Chính vì dữ liệu được cung cấp rời rạc nên tại một thời điểm chúng ta không thể biết trước độ lớn và vị trí dữ liệu sẽ được mã hóa. Do đó đối với các chế độ luồng input cho thuật toán mã hóa được xem là một luồng các bit của bản rõ được lần lượt theo thời gian.

Trong chế độ OFB và CFB dòng khoá được tạo ra sẽ được cộng modulo 2 với bản rõ. OFB thực sự là một hệ mã đồng bộ: dòng khoá được thành lập bởi việc tạo lập các vector khởi tạo 64 bit (vector IV). Ta xác định z_0 = IV và tính dòng khoá $z_1z_2 ... z_n$ theo quy tắc $z_i = e_k(z_{i-1})$ với i ≥ 1. Sau đó dãy bản rõ $x_1x_2 ... x_n$ sẽ được mã hoá bằng cách tính $y_i = x_i \oplus z_i$ với i ≥ 1.

Trong chế độ CFB, ta bắt đầu với y_0 = IV (vector khởi tạo 64 bit) và tạo phần tử z_i của dòng khoá bằng cách mã hoá khối bản mã trước đó. Tức là $z_i = e_k(y_{i-1})$ với i≥1 và $y_i = x_i \oplus z_i$ với i≥1. Việc sử dụng CFB được mô tả bằng sơ đồ sau (e_k trong trường hợp này được sử dụng cho cả mã hoá và giải mã):



Hình 3.16: Chế độ CFB

Cũng có một vài dạng khác của OFB và CFB được gọi là chế độ phản hồi k-bit (1< k < 64). Ở đây ta đã mô tả chế độ phản hồi 64 bit. Các chế độ phản hồi 1-bit và 8-bit thường được sử dụng cho phép mã hoá đồng thời 1 bit (hay byte) dữ liệu. Kỹ thuật cơ bản được sử dụng ở đây là một thanh ghi dịch 64 bit và mỗi bước dịch được k-bit làm đầu vào cho mã hoá. K-bit bên trái của đầu vào hàm mã hoá được XOR với đơn vị đầu của block bản rõ tiếp theo để đưa ra một đơn vị bản mã truyền đi và đơn vị này được đưa lại vào k-bit bên phải của thanh ghi dịch. Quá trình xử lý tiếp tục cho tới khi tất cả đơn vị bản rõ đều được mã hoá. Điểm khác nhau giữa CFB và OFB là k-bit hồi tiếp cho bộ ghi dịch được lấy từ trước hay sau bộ XOR (nếu lấy sau bộ XOR thì dữ liệu đã mã hoá ứng với CFB, còn lấy phía trước thì là OFB).

Nhìn chung , bốn chế độ của DES đều có những ưu nhược điểm riêng. Ở chế độ ECB và OFB, sự thay đổi của một khối bản rõ x_i 64 bit sẽ làm thay đổi khối bản mã y_i tương ứng, nhưng các khối bản khác thì không bị ảnh hưởng. Trong một số tình huống, đây là một tính chất đáng mong muốn. Ví dụ như chế độ OFB thường được dùng để mã hoá trong việc truyền tín hiệu qua vệ tinh.

Mặt khác, ở chế độ CBC và CFB, nếu một khối bản rõ x_i bị thay đổi thì y_i và các khối tiếp theo sẽ bị ảnh hưởng. Như vậy ở chế độ CBC và CFB có thể được sử dụng rất hiệu quả trong mục đích xác thực. Cũng vì lý do đó nên CFB thường được dùng để mã hóa trong các trường hợp mà đường truyền tốt, tín hiệu ít nhiễu. Đặc biệt hơn, các chế độ này dùng để tạo mã xác thực bản tin (MAC – Message Authentication Code). MAC được gắn thêm vào các khối bản rõ để thuyết phục R (receiver) rằng đây chính là dãy bản rõ được gửi từ S (sender) mà không phải một ai khác giả mạo. Như vậy MAC đảm bảo tính xác thực của bản tin.

Ta sẽ mô tả cách sử dụng chế độ CBC để tạo MAC. Ta bắt đầu bằng vector khởi tạo IV chứa toàn số 0. Sau đó dùng chế độ CBC để tạo các khối bản mã $y_1y_2...y_n$ với khoá K. Cuối cùng ta xác định MAC là y_n. Người gửi S (sender) sẽ phát đi khối bản rõ $x_1x_2 ... x_n$ cùng với MAC. Khi người nhận R (receiver) thu được $x_1x_2 ... x_n$, anh ta sẽ khôi phục lại y_1, y_2, y_n bằng khoá bí mật K và xác minh liệu y_n có giống MAC của mình thu được hay không. Nếu một người thứ ba E (enemy) thu chặn được bản rõ $x_1x_2 ... x_n$ rõ ràng E không thể tạo ra MAC hợp lệ nếu không biết khoá bí mật K mà S và R đang dùng. Hơn nữa, nếu E thay đổi ít nhiều nội dung thì chắc chắn E không thể thay đổi được MAC để được R chấp nhận.

Thông thường ta muốn kết hợp cả tính xác thực lẫn độ bảo mật. Điều đó được thực hiện như sau: trước tiên S dùng khoá K_1 để tạo MAC cho dãy bản rõ $x_1x_2... x_n$, sau đó S xác định x_{n+1} là MAC, rồi mã hoá dãy $x_1x_2 ... x_nx_{n+1}$ bằng khoá K_2 để tạo dãy bản mã $y_1y_2 ... y_ny_{n+1}$. Khi R nhận được $y_1y_2...y_ny_{n+1}$, R sẽ giải mã bằng khoá K_2 và sau đó kiểm tra xem x_{n+1} có phải là MAC (bằng khoá K_1) của dãy bản rõ $x_1x_2 ... x_n$ hay không.

3. Bài tập

Bài tập 3.1: Hãy giải mã bản mã được mã hóa bằng hệ mã Caesar sau (sử dụng bảng chữ cái tiếng Anh): WKXPEVXS.

Bài tập 3.2 (khó): Thông điệp bí mật ẩn sau đoạn văn bản tiếng Anh sau là gì:

“The supply of game for London is going steadily

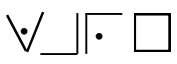
up. Head keeper Hudson, we believe, has been

now told to receive all orders for fly paper and for

preservations of your hen-pheasant's life.”

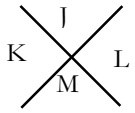
Trích trong tác phẩm ”The Gloria Scott”.

Bài tập 3.3: Sử dụng bảng sau (hệ mã Freemason) để giải mã thông điệp:

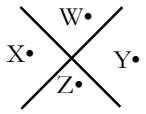


Bảng mã các ký tự:

A	B	C
D	E	F
G	H	I



N•	O•	P•
Q•	R•	S•
T•	U•	V•



Gợi ý: đây là một hệ mã thay thế tượng hình.

Bài tập 3.4: Hãy tìm thông điệp bí mật ẩn giấu trong đoạn văn bản sau:

Dear George, 3rd March
Greetings to all at Oxford. Many thanks for your letter and for the Summer examination package. All Entry Forms and Fees Forms should be ready for final dispatch to the Syndicate by Friday 20th or at the very least, I'm told, by the 21st. Admin has improved here, though there's room for improvement still; just give us all two or three more years and we'll really show you! Please don't let these wretched 16+ proposals destroy your basic O and A pattern. Certainly this sort of change, if implemented immediately, would bring chaos.

Bài tập 3.5: Cho hệ mã Affine được cài đặt trên Z_{99}. Khi đó khóa là các cặp (a, b) trong đó a, b $\in Z_{99}$ với (a, 99) = 1. Hàm mã hóa $E_K(x) = (a * x + b) \bmod 99$ và hàm giải mã $D_K(x) = a^{-1} * (x - b) \bmod 99$.

a) Hãy xác định số khóa có thể được sử dụng cho hệ mã này.
b) Nếu như khóa giải mã là $K^{-1} = (16, 7)$, hãy thực hiện mã hóa xâu m = "DANGER".

Bài tập 3.6: Cho hệ mã Affine được cài đặt trên Z_{39}. Khi đó khóa là các cặp (a, b) trong đó a, b $\in Z_{39}$ với (a, 39) = 1. Hàm mã hóa $E_K(x) = (a * x + b) \bmod 39$ và hàm giải mã $D_K(x) = a^{-1} * (x - b) \bmod 39$.

a) Hãy xác định số khóa có thể được sử dụng cho hệ mã này.
b) Nếu như khóa giải mã là $K^{-1} = (23, 7)$, hãy thực hiện mã hóa xâu m = "ATTACK".

Bài tập 3.7: Cho hệ mã Affine được cài đặt trên Z_{55}. Khi đó khóa là các cặp (a, b) trong đó a, b $\in Z_{55}$ với (a, 55) = 1. Hàm mã hóa $E_K(x) = (a * x + b) \bmod 55$ và hàm giải mã $D_K(x) = a^{-1} * (x - b) \bmod 55$.

a) Hãy xác định số khóa có thể được sử dụng cho hệ mã này.
b) Khóa giải mã là $K^{-1} = (13, 17)$, hãy xác định khóa mã hóa.

Bài tập 3.8: Giả sử hệ mã Affine được cài đặt trên Z_{99}.

a) Hãy xác định só khóa có thể có của hệ mã.
b) Giả sử khóa mã hóa là (16, 7), hãy xác định khóa giải mã.

Bài tập 3.9: Giả sử hệ mã Affine được cài đặt trên Z_{126}.

a) Hãy xác định só khóa có thể có của hệ mã.
b) Giả sử khóa mã hóa là (23, 7), hãy xác định khóa giải mã.

Bài tập 3.10: Cho hệ mã Hill có M = 2.

a) Ma trận A = $\begin{bmatrix} 5 & 3 \\ 13 & 17 \end{bmatrix}$ có thể được sử dụng làm khóa cho hệ mã trên không giải thích.

b) Cho A = $\begin{bmatrix} 12 & 5 \\ 3 & 7 \end{bmatrix}$ hãy thực hiện mã hóa và giải mã với xâu S = “HARD”.

Bài tập 3.11: Cho hệ mã Hill có M = 2.

a) Ma trận A = $\begin{bmatrix} 5 & 3 \\ 11 & a \end{bmatrix}$ được sử dụng làm khóa cho hệ mã trên. Hãy tìm tất cả các khóa có thể sử dụng của hệ mã trên.

b) Giả sử người ta sử dụng hệ mã trên để mã hóa bản rõ P = “EASY” và thu được bản mã là “UMQA”. Hãy thực hiện giải mã với bản mã là C = “MCDZUZ” và đưa ra bản rõ.

Bài tập 3.12: Cho hệ mã Hill có M = 2.

a) Ma trận A = $\begin{bmatrix} 15 & 13 \\ 7 & a \end{bmatrix}$ được sử dụng làm khóa cho hệ mã trên. Hãy tìm tất cả các khóa có thể sử dụng của hệ mã trên.

b) Giả sử người ta sử dụng hệ mã trên để mã hóa bản rõ P = “MARS” và thu được bản mã là “YARH”. Hãy thực hiện giải mã với bản mã là C = “MANNTF” và đưa ra bản rõ.

Bài tập 3.13: Cho hệ mã Vigenere có M = 6, K = “CIPHER”.

a) Hãy thực hiện mã hóa xâu P = “THIS IS MY TEST“.

b) Hãy thực hiện giải mã xâu M = “EICJIC RTPUEI GBGLEK CBDUGV”.

Bài tập 3.14: Cho hệ mã Vigenere có M = 6. Mã hóa xâu P = “THIS IS MY TEST“ người ta thu được bản mã là “LLKJML ECVVWM”.

a) Hãy tìm khóa mã hóa đã dùng của hệ mã trên.

b) Dùng khóa tìm được ở phần trên hãy giải mã bản mã C = “KLGZWT OMBRVW”.

Bài tập 3.15: Cho hệ mã Vigenere có M = 6. Mã hóa xâu P = “SPIRIT“ người ta thu được bản mã là “OXHRZW”.

a) Hãy tìm khóa mã hóa đã dùng của hệ mã trên.

b) Dùng khóa tìm được ở phần trên hãy giải mã bản mã C = “BQETYH HMBEEW”.

Bài tập 3.16: Cho hệ mã Vigenere có M = 6. Giải mã xâu C = “RANJLV” người ta thu được bản rõ là “CIPHER”.

a) Tìm khóa đã sử dụng của hệ mã trên.

b) Dùng khóa tìm được ở phần trên hãy hãy giải mã xâu M = “PLDKCI DUJQJO“.

Bài tập 3.17: Phương pháp mã hóa thay thế đơn giản

Đoạn văn bản sau được mã hóa bằng cách sử dụng một phương pháp mã hóa thay thế đơn giản. Bản rõ là một phần của một văn bản tiếng Anh viết hoa, bỏ qua các dấu câu. Hãy sử dụng bảng thống kê tần suất xuất hiện của các chữ cái trong tiếng Anh để giải mã bản mã đã cho.

ODQSOCL OW GIU BOEE QRROHOCS QV GIUR KIA QF Q DQCQSLR WIR

ICL IW CQFQF EIYQE YIDJUVLR FGFVLDF GIU SLV OCVI GIUR

IWWOYL IC VXQV DICPQG DIRCOCS VI WOCP VXL JXICLF ROCSOCS

LHLRG YQEELR OF Q POFVRQUSXV YICWUFLP CQFQ BIRMLR QCP

LHLRG YQEELR QFFURLF GIU VXQV XOF IR XLR WOEL IR

QYYIUCVOCS RLYIRP IR RLFLQRYX JRIKLYV LHLRG ICL IW BXOYX

OF DOFFOCS WRID VXL YIDJUVLR FGFVLD OF QAFIEUVLEG HOVQE

Bảng thống kê tần suất xuất hiện của các chữ cái trong tiếng Anh:

Chữ cái	Tần suất	Chữ cái	Tần suất	Chữ cái	Tần suất
A	8.2 %	J	0.2 %	S	6.3 %
B	1.5 %	K	08 %	T	9.1 %
C	2.8 %	L	4.0 %	U	2.8 %
D	4.3 %	M	2.4 %	V	1.0 %
E	12.7 %	N	6.7 %	W	2.3 %
F	2.2 %	O	7.5 %	X	0.1 %
G	2.0 %	P	1.9 %	Y	2.0 %
H	6.1 %	Q	0.1 %	Z	0.1 %
I	7.0 %	R	6.0 %		

Bài tập 3.18: Cho bản mã sau:

EYMHP GZYHH PTIAP QIHPH YIRMQ EYPXQ FIQHI AHYIW ISITK MHXQZ PNMQQ XFIKJ MKXIJ RIKIU XSSXQ ZEPGS ATIHP PSXZY H

Biết rằng bảng chữ cái sử dụng là tiếng Anh, hãy thực hiện các yêu cầu sau:

a) Hãy đưa ra bảng phân phối tần suất của các chữ cái trong bản mã trên.
b) Giả sử bản mã trên nhận được bằng cách sử dụng phương pháp mã hóa đổi chỗ hoặc thay thế đơn âm, hãy dựa vào bảng phân phối tần suất ở phần a để xác định xem khả năng nào là cao hơn (hệ mã đổi chỗ hay thay thế đơn âm)?
c) Hãy xác định bản rõ nếu như phần bắt đầu của bản rõ là “What ought …”.
d) Giải thích cách thành lập khóa của hệ mã.

Bài tập 3.19 (khó):

Hãy giải mã bản mã được mã hóa bằng hệ mã Vigenere sau, xác định độ khóa sử dụng biết rằng bản rõ gồm các chữ cái trong bảng mã tiếng Anh.

IGDLK	MJSGC	FMGEP	PLYRC	IGDLA	TYBMR	KDYVY	XJGMR	TDSVK	ZCCWG	ZRRIP
UERXY	EEYHE	UTOWS	ERYWC	QRRIP	UERXJ	QREWQ	FPSZC	ALDSD	ULSWF	FFOAM
DIGIY	DCSRR	AZSRB	GNDLC	ZYDMM	ZQGSS	ZBCXM	OYBID	APRMK	IFYWF	MJVLY
HCLSP	ZCDLC	NYDXJ	QYXHD	APRMQ	IGNSU	MLNLG	EMBTF	MLDSB	AYVPU	TGMLK
MWKGF	UCFIY	ZBMLC	DGCLY	VSCXY	ZBVEQ	FGXKN	QYMIY	YMXKM	GPCIJ	HCCEL
PUSXF	MJVRY	FGYRQ								

Sử dụng một trong các ngôn ngữ lập trình C, C++, Java hoặc C# để làm các bài tập sau:

Bài tập 3.20: Viết chương trình đếm tần số xuất hiện của các chữ cái tiếng Anh trong một văn bản tiếng Anh ở dạng file text.

Bài tập 3.21: Viết chương trình đếm tần số xuất hiện của các chữ cái tiếng Việt trong một văn bản tiếng Việt ở dạng file RTF.

Bài tập 3.22: Viết chương trình cài đặt thuật toán mã hóa và giải mã của hệ mã Ceasar.

Bài tập 3.23: Viết chương trình cài đặt thuật toán mã hóa và giải mã của hệ mã Affine.

Bài tập 3.24: Viết chương trình tính định thức của ma trận vuông cấp N (N < 20).

Bài tập 3.25: Viết chương trình cài đặt thuật toán mã hóa và giải mã của hệ mã Hill.

Bài tập 3.26: Viết chương trình cài đặt thuật toán mã hóa và giải mã của hệ mã Vigenere.

Bài tập 3.27: Viết chương trình mã hóa và giải mã file theo hệ mã DES với các cơ chế mã hóa ECB, CBC.

Bài tập 3.28: Viết chương trình mã hóa và giải mã file theo hệ mã AES với các cơ chế mã hóa ECB, CBC.

CHƯƠNG IV: CÁC HỆ MÃ MẬT KHÓA CÔNG KHAI

Trong các hệ mã mật khóa bí mật nếu chúng ta biết khóa và hàm mã hóa chúng ta có thể tìm được khóa và hàm giải mã một cách nhanh chóng (thời gian đa thức).

Một hệ mã mật khóa bí mật là một hệ mã mật mà tất cả mọi người đều biết hàm mã hóa và khóa mã hóa nhưng không tồn tại một thuật toán thời gian đa thức để có thể tính được khóa giải mã từ các thông tin đó.

1. Khái niệm hệ mã mật khóa công khai

Các hệ mã được trình bày trong các chương trước được gọi là các hệ mã khóa bí mật, khóa đối xứng, hay các hệ mã truyền thống (conventional).

Các hệ mã này có các điểm yếu sau đây:

- Nếu số lượng người sử dụng lớn thì số khóa sẽ tăng rất nhanh, chẳng hạn với n người sử dụng thì số khóa sẽ là n*(n-1)/2 do đó rất khó quản lý, phức tạp và không an toàn.
- Dựa trên các hệ mã này không thể xây dựng các khái niệm và dịch vụ như chữ ký điện tử, dịch vụ xác thực hóa người dùng cho các ứng dụng thương mại điện tử.

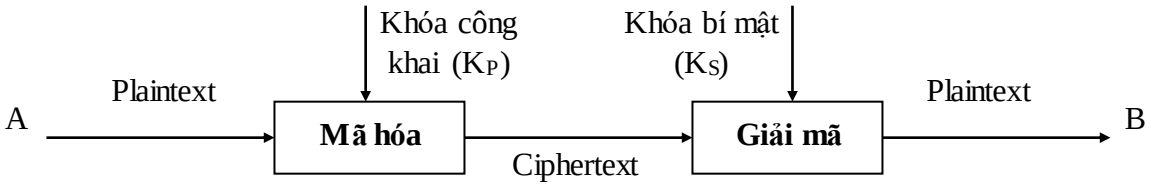
Vào năm 1975 Diffie và Hellman trong một công trình của mình (một bài báo) đã đề xuất ra các ý tưởng cho phép xây dựng lên các hệ mã hoạt động theo các nguyên tắc mới gắn liền với các bên truyền tin chứ không gắn với các cặp truyền tin.

Nguyên tắc hoạt động của các hệ mã là mỗi bên tham gia truyền tin sẽ có 2 khóa, một khóa gọi là khóa bí mật và một khóa được gọi là khóa công khai. Khóa bí mật là khóa dùng để giải mã và được giữ bí mật (K_S), khóa công khai là khóa dùng để sinh mã được công khai hóa để bất cứ ai cũng có thể sử dụng khóa này gửi tin cho người chủ của hệ mã (K_P). Ngày nay chúng ta có thể thấy rất rõ nguyên tắc này trong việc gửi email, mọi người đều có thể gửi email tới một địa chỉ email nào đó, nhưng chỉ có người chủ sở hữu của địa chỉ email đó mới có thể đọc được nội dung của bức thư, còn những người khác thì không. Với các hệ mã khóa công khai việc phân phối khóa sẽ trở nên dễ dàng hơn qua các kênh cung cấp khóa công cộng, số lượng khóa hệ thống quản lý cũng sẽ ít hơn (là n khóa cho n người dùng). Các dịch vụ mới như chữ ký điện tử, thỏa thuận khóa cũng được xây dựng dựa trên các hệ mã này.

Các yêu cầu của loại hệ mã này:

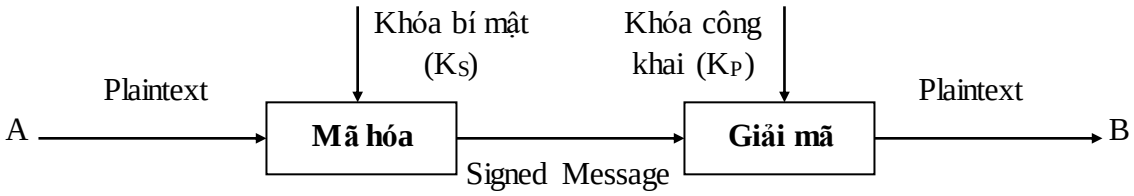
- Việc sinh K_P, K_S phải dễ dàng
- Việc tính $E(K_P, M)$ là dễ dàng
- Nếu có $C = E(K_P, M)$ và K_S thì việc tìm bản rõ cũng là dễ
- Nếu biết K_P thì việc dò tìm K_S là khó
- Việc khôi phục bản rõ từ bản mã là rất khó

Khi A muốn truyền tin cho B, A sẽ sử dụng khóa K_P của B để mã hóa tin tức và truyền bản mã tới cho B, B sẽ sử dụng khóa bí mật của mình để giải mã và đọc tin:



Hình 4.1: Mô hình sử dụng 1 của các hệ mã khóa công khai PKC

Ciphertext = $E(K_P, Plaintext)$,Plantext = $D(K_S, E(K_P, Plaintext))$ (1)



Hình 4.2: Mô hình sử dụng 2 của các hệ mã khóa công khai PKC

Ciphertext = $D(K_S, Plaintext)$, Plaintext = $E(K_P, D(K_S, Plaintext))$ (2)

Mô hình (2) được sử dụng cho các hệ chữ ký điện tử còn mô hình (1) được sử dụng cho các hệ mã mật. Các hệ mã này được gọi là các hệ mã khóa công khai PKC (Public Key Cryptosystems) hay các hệ mã bất đối xứng (Asymmetric Encryption Scheme).

2. Nguyên tắc cấu tạo của các hệ mã mật khóa công khai

Các hệ mã khóa công khai được xây dựng dựa trên các hàm được gọi là các hàm 1 phía hay hàm 1 chiều (one–way functions).

Hàm một chiều f: $X \rightarrow Y$ làm một hàm mà nếu biết $x \in X$ ta có thể dễ dàng tính được $y = f(x)$. Nhưng với y bất kỳ $\in Y$ việc tìm $x \in X$ sao cho $y = f(x)$ là khó. Có nghĩa là việc tìm hàm ngược f^{-1} là rất khó.

Ví dụ nếu chúng ta có các số nguyên tố $P_1, P_2, ..., P_n$ thì việc tính $N = P_1 * P_2 * ... * P_n$ là dễ nhưng nếu có N thì việc phân tích ngược lại là một bài toán khó với N lớn.

Để thuận tiện các hàm một phía được sử dụng trong các hệ mã PKC thường được trang bị các cửa bẫy (trapdoor) giúp cho việc tìm x thỏa mã $y = f(x)$ là dễ dàng nếu chúng ta biết được cửa bẫy này.

Hàm của bẫy (trapdoor function): là một hàm một chiều trong đó việc tính f^{-1} là rất nhanh khi chúng ta biết được cửa bẫy của hàm. Ví dụ việc tìm nghiệm của bài toán xếp balô 0/1 trong hệ mã xếp balô Knapsack mà chúng ta sẽ học trong phần tiếp theo là một hàm một phía (việc mã hóa rất nhanh và dễ dàng nhưng tìm vectơ nghiệm tương ứng là khó) nhưng nếu ta biết cửa bẫy (Vectơ xếp balô siêu tăng A') thì việc giải bài toán lại rất dễ dàng.

3. Một số hệ mã khóa công khai

3.1. Hệ mã knapsack

Bài toán xếp ba lô tổng quát:

Cho M, N và A_1, A_2, ..., A_N là các số nguyên dương tìm các số x_i không âm sao cho:

$$\mathbf{M} = \sum_{i=1}^{N} x_i * A_i$$

Vecto A = (A_1, A_2, ..., A_N) được gọi là vecto xếp balô còn vectơ X = (x_1, x_2, …, x_N) là vectơ nghiệm.

Một trường hợp riêng đáng quan tâm của bài toán xếp ba lô tổng quát là trường hợp mà $x_i \in \{0, 1\}$. Khi đó ta có bài toán xếp ba lô 0, 1.

Vecto xếp ba lô siêu tăng: Trong trường hợp vecto (A_1, A_2, ..., A_N) được sắp lại thành (A'_1, A'_2, ..., A'_N) sao cho:

$\forall$ i ta có: $\sum_{j<i} A'_j < A'_i$ thì vecto (A_1, A_2, ..., A_N) được gọi là vecto xếp balo siêu tăng.

Khi (A_1, A_2, ..., A_N) là một vecto xếp balo siêu tăng ta có ngay tính chất: M >= A'_i $\forall$ i. Do đó việc giải bài toán xếp ba lô 0/1 trở nên dễ dàng hơn rất nhiều.

Hệ mã knapsack do Merkle và Hellman đưa ra vào năm 1978.

Cách xây dựng:

1. Chọn 1 vecto siêu tăng A' = (a'_1, a'_2, ..., a'_N), chọn 1 số M > 2 * a'_N, chọn ngẫu nhiên 1 số u < M và (u, M) = 1
2. Xây dựng Vecto A = (a_1, a_2, ..., a_N) trong đó a_i = (a'_i * u) mod M
3. Khóa: K_P = (A, M), K_S = (u, u^{-1})
4. Không gian các bản rõ là không gian mọi dãy N bit

   P = (x_1, x_2, ..., x_n).

Mã hóa: C = $(\sum_{i=1}^{N} a_i * x_i)$ mod M

Giải mã: tính C' = C * u^{-1} mod M sau đó giải bài toán xếp ba lô 0/1 với A', C' từ đó tìm được P = (x_1, x_2, ..., x_n).

Ví dụ 1: Cho hệ mã Knapsack có A' = (2, 3, 6, 12, 25), N = 5, M = 53, u = 46, u^{-1} = 15.

a) Hãy tìm các khóa của hệ mã trên

b) Mã hóa và giải mã bản mã tương ứng của bản rõ M = 01001.

3.2. Hệ mã RSA

Hệ mã RSA được đặt tên dựa theo các chữ cái đầu của 3 tác giả của hệ mã là Rivest, Shamir và Adleman. Đây là thuật toán mã hóa nổi tiếng nhất và cũng là thuật toán được ứng dụng thực tế nhất.

Để cài đặt RSA ban đầu mỗi người dùng sinh khóa công khai và khóa bí mật của mình bằng cách:

- chọn hai số nguyên tố lớn ngẫu nhiên (cỡ gần 100 chữ số) khác nhau p và q
- tính N = p*q
- chọn một số e nhỏ hơn N và (e, ∅(N)) = 1, e được gọi là số mũ lập mã
- tìm phần tử ngược của e trên vành module ∅(N), d là số mũ giải mã
- khóa công khai là K_P = (e, N)
- khóa bí mật là $K_S = K^{-1}_P$ = (d, p, q)

Việc thiết lập khóa này được thực hiện 1 lần khi một người dùng thiết lập (thay thế) khóa công khai của họ. Mũ e thường là khá nhỏ (để mã hóa nhanh), và phải là nguyên tố cùng nhau với ∅(N). Các giá trị thường được chọn cho e là 3 hoặc $2^{16} – 1 = 65535$. Tuy nhiên khi e nhỏ thì d sẽ tương đối lớn. Khoá bí mật là (d, p, q). Các số p và q thường có giá trị xấp xỉ nhau nhưng không được bằng nhau. Chú ý là việc để lộ một trong các thành phần trên sẽ làm cho hệ mã hóa trở thành không an toàn.

Sử dụng RSA

- để mã hóa một thông điệp M: $C = M^e \pmod N$ (0<= M < N)
- giải mã: $M = C^d \pmod N$

Thuật toán mã hóa RSA làm việc được bởi vì nó dựa trên cơ sở toán học là sự tổng quát định lý Ferma nhỏ của Ơclit: $X^{\varnothing(N)} = 1 \pmod N$. Trong thuật toán RSA chúng ta chọn e và d là nghịch đảo của nhau trên vành $Z_{\varnothing(N)}$ với e được chọn trước.

Do đó chúng ta sẽ có $e.d \equiv 1 \bmod \varnothing(N)$, suy ra:

$M = C^d = M^{e.d} = M^{1+q.\varnothing(N)} = M . (M^{\varnothing(N)})^q = M \bmod N$

Công thức này đảm bảo việc giải mã sẽ cho kết quả đúng là bản rõ ban đầu (chú ý là điều này chỉ đúng khi p khác q).

Ví dụ 1: Cho hệ mã RSA có N = p*q = 11 * 47 = 517, e = 3.

- Hãy tìm các khóa công khai và bí mật của hệ mã trên
- Mã hóa bản rõ M = 26.

Đầu tiên ta tính được ∅(N) = 460 = 10 * 46, do (3,460) = 1 nên áp dụng thuật toán Ơclit mở rộng ta tìm được d = 307.

Vậy khóa công khai của hệ mã K_P = (e, N) = (3, 517), khóa bí mật là K_S = (d, p, q) = (307, 11, 47).

Mã hóa M = 26 ta có $C = M^e \bmod N = 26^3 \bmod 517 = 515$.

Độ an toàn của RSA

Độ an toàn của RSA phụ thuộc vào độ khó của việc tính ∅(N) và điều này đòi hỏi chúng ta cần phân tích N ra thừa số nguyên tố. Thuật toán phân tích số nguyên tố hiệu quả nhất hiện nay là Brent-Pollard, chúng ta hãy xem xét bảng thống kê sau để thấy được tốc độ hoạt động của nó:

Số chữ số trong hệ thập phân của N	**Số các thao tác Bit để phân tích N**

20	7.20e+03
40	3.11e+06
60	4.63e+08
80	3.72e+10
100	1.97e+12
120	7.69e+13
140	2.35e+15
160	5.92e+16
180	1.26e+18
200	2.36e+19

Bảng 4.1: Tốc độ của thuật toán Brent-Pollard

Các nghiên cứu về vấn đề phân tích các số nguyên lớn hiện nay tiến triển rất chậm, các tiến bộ lớn nhất cũng chỉ là các cải tiến về thuật toán và có thể nói rằng trừ khi có các đột phá trong việc phân tích các số 1024 bit, RSA là an toàn trong thời điểm hiện nay.

Các nhà mật mã học phát minh ra hệ mã RSA đã đưa ra một giải thưởng trị giá 100 $ vào năm 1977. Đó là một hệ mã với số N có 129 chữ số, thách thức này đã được phá.

Trên thực tế để cài đặt RSA cần phải thực hiện các thao tác modulo với các số 300 chữ số (hay 1024 bit) mà hiện nay các máy tính mới chỉ thao tác với các số nguyên 64 bit, điều này dẫn đến nhu cầu cần các thư viện số học nhân chính xác để làm việc với các số nguyên lớn này. Ngoài ra việc sử dụng RSA cần tới các số nguyên tố lớn nên chúng ta cũng phải có một cơ sở dữ liệu các số nguyên tố.

Để tăng tốc cho RSA chúng ta có thể sử dụng một số phương pháp khác chẳng hạn như cải tiến các phép tính toán nhân hai số lớn hoặc tăng tốc việc tìm bản mã, bản rõ.

Đối với phép nhân 2 số n bit thông thường chúng ta cần thực hiện $O(n^2)$ phép tính bit. Thuật toán nhân các số nguyên Schonhage – Strassen cho phép chúng ta thực hiện phép nhân 2 số với độ phức tạp là O(n log n) với các bước như sau:

- Chia mỗi số nguyên thành các khối, sử dụng các khối này như các hệ số của một đa thức.
- Tính các đa thức này tại một số các điểm thích hợp, và nhân các kết quả thu được.
- Nội suy các kết quả này hình thành các hệ số của đa thức tích
- Kết hợp các hệ số để hình thành nên tích của hai số ban đầu
- Biến đổi Fourier rời rạc, và lý thuyết chặp có thể được sử dụng để tăng tốc độ của quá trình nội suy.

Một cách khác nữa để tăng tốc việc nhân các số lớn trong hệ mã RSA là sử dụng các phần cứng chuyên dụng với các thuật toán song song.

Như đã trình bày ở phần trước khi mã hóa chúng ta thường chọn e nhỏ để đẩy nhanh quá trình mã hóa nhưng điều này cũng đồng nghĩa là việc giải mã sẽ chậm do số mũ lớn. Một cải tiến đáng kể trong tốc độ giải mã RSA có thể nhận được bằng cách sử dụng định lý phần dư Trung Hoa làm việc với modulo p và q tương ứng thay vì N. Vì p và q chỉ bằng một nửa của N nên tính toán sẽ nhanh hơn nhiều.

Định lý phần dư Trung Hoa được sử dụng trong RSA bằng cách tạo ra hai phương trình từ việc giải mã $M = C^d \pmod N$ như sau:

$M_1 = M \bmod p = (C \bmod p)^{d \bmod (p-1)}$

$M_2 = M \bmod q = (C \bmod q)^{d \bmod (q-1)}$

Sau đó ta giải hệ:

$M = M_1 \bmod p$

$M = M_2 \bmod q$

Hệ này có nghiệm duy nhất theo định lý phần dư Trung Hoa

$M = [(M_2 + q - M_1)u \bmod q]\, p + M_1$

Trong đó $p.u \bmod q = 1$

Việc sử dụng định lý phần dư Trung Hoa là một phương pháp được sử dụng rộng rãi và phổ biến để tăng tốc độ giải mã của RSA.

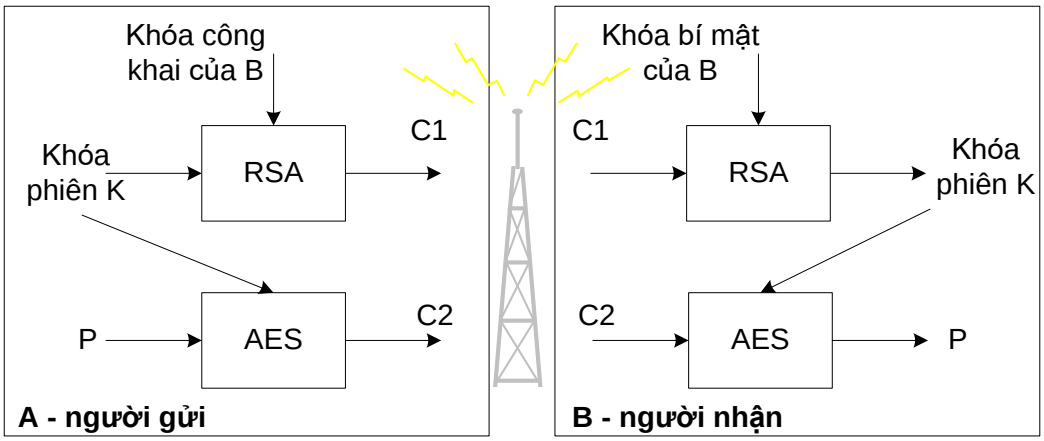
Hiện tượng lộ bản rõ

Một hiện tượng cần lưu ý khi sử dụng các hệ mã RSA là hiện tượng lộ bản rõ. Ta hãy xét hệ mã RSA có N = p*q = 5*7, e = 17, khi đó với M = 6 ta có $C = 6^{17} \bmod N = 6$.

Tương tự với hệ mã RSA có N = p*q = 109*97, e = 865, với mọi M ta đều có $M^e \bmod N = M$.

Theo tính toán thì với một hệ mã RSA có N = p*q và e bất kỳ, số lượng bản rõ sẽ bị lộ khi mã hóa sẽ là (1 + (e-1, p-1))*(1 + (e-1, q-1)).

Trong số các hệ mã khóa công khai thì có lẽ hệ mã RSA (cho tới thời điểm hiện tại) là hệ mã được sử dụng rộng rãi nhất.Tuy nhiên do khi làm việc với dữ liệu đầu vào (thông điệp mã hóa, bản rõ) lớn thì khối lượng tính toán rất lớn nên trên thực tế người ta hay dùng hệ mã này để mã hóa các dữ liệu có kích thước nhỏ, hoặc có yêu cầu bảo mật cao, chẳng hạn như các khóa phiên (session key) trong các phiên truyền tin. Khi đó hệ mã RSA sẽ được sử dụng kết hợp với một hệ mã khối khác, chẳng hạn như AES, theo mô hình lai ghép như sau:



Hình 4.3: Mô hình ứng dụng lai ghép RSA với các hệ mã khối

3.3. Hệ mã El Gamal

Hệ mã El Gamal là một biến thể của sơ đồ phân phối khoá Diffie – Hellman. Hệ mã này được El Gamal đưa ra vào năm 1985. Giống như sơ đồ phân phối khóa Diffie – Hellman tính an toàn của nó dựa trên tính khó giải của bài toán logarit rời rạc. Nhược điểm chính của nó là kích thước thông tin sau khi mã hóa gửi đi sẽ tăng gấp đôi so với thông tin gốc.

Tuy nhiên so với RSA, El Gamal không có nhiều rắc rối về vấn đề bản quyền sử dụng.

Ban đầu người ta sẽ chọn một số nguyên tố lớn p và hai số nguyên tuỳ ý nhỏ hơn p là a (a là một phần tử nguyên thủy của Z^*_P) và x (x là của người nhận, bí mật) sau đó tính:

$y = a^x \bmod p$

Để mã hóa một thông điệp M (là một số nguyên trên Z_P) thành bản mã C người gửi chọn một số ngẫu nhiên k nhỏ hơn p và tính khóa mã hóa K:

$K = y^k \bmod p$

Sau đó tính cặp bản mã:

- $C_1 = a^k \bmod p$
- $C_2 = K.M \bmod p$

Và gửi bản mã $C = (C_1, C_2)$ đi (chú ý là sau đó k sẽ bị huỷ).

Để giải mã thông điệp đầu tiên ta cần tính lại khóa mã hóa thông điệp K:

$K = C_1^x \bmod p = a^{k.x} \bmod p$

Sau đó tính M bằng cách giải phương trình sau đây:

$M = C_2 . K^{-1} \bmod p$

Việc giải mã bao gồm việc tính lại khóa tạm thời K (rất giống với mô hình của Diffie – Hellman đưa ra). Khóa công khai của hệ mã là (p, a, y), khóa bí mật là x.

Ví dụ: Cho hệ mã El Gamal có P = 97, a = 5, x = 58.

- Tìm khóa của hệ mã trên.
- Mã hóa bản rõ M = 3 với k được chọn bằng 36.

Trước hết ta tính $y = 5^{58} \bmod 97 = 44$, từ đó suy ra $K_P = (P, a, y) = (97, 5, 44)$ và $K_S = (58)$.

Để mã hóa thông điệp M = 3 ta tính khóa $K = 44^{36} \bmod 97 = 75$ sau đó tính:

- $C_1 = 5^{36} = 50 \bmod 97$
- $C_2 = 75.3 \bmod 97 = 31 \bmod 97$

Vậy bản mã thu được là C = (50, 31).

Vấn đề đối với các hệ mã khóa công khai nói chung và El Gamal nói riêng là tốc độ (do phải làm việc với các số nguyên lớn), bên cạnh đó dung lượng bộ nhớ dành cho việc lưu trữ các khóa cũng lớn. Với hệ mã El Gamal chúng ta cần gấp đôi bộ nhớ để chứa bản mã so với các hệ mã khác. Ngoài ra do việc sử dụng các số nguyên tố nên việc sinh khóa và quản lý khóa cũng khó khăn hơn với các hệ mã khối. Trên thực tế các hệ mã khóa công khai thường được sử dụng kết hợp với các hệ mã khối (mã hóa khóa của hệ mã) hoặc để mã hóa các thông tin có dung lượng nhỏ và là một phần quan trọng của một phiên truyền tin nào đó.

Thám mã đối với hệ mã El Gamal

Để thực hiện thám mã hệ mã El Gamal chúng ta cần giải bài toán Logaritm rời rạc. Ở đây chúng ta sẽ xem xét hai thuật toán có thể áp dụng để giải bài toán này, với độ phức tạp và khả năng áp dụng khác nhau.

Thuật toán Shank

Thuật toán này còn có tên khác là thuật toán cân bằng thời gian – bộ nhớ (Time-Memory Trade Off), có nghĩa là nếu chúng ta có đủ bộ nhớ thì có thể sử dụng bộ nhớ đó để làm giảm thời gian thực hiện của thuật toán xuống.

Input: số nguyên tố p, phần tử nguyên thủy a của Z_p^*, số nguyên y.

Output: cần tìm x sao cho $a^x \bmod p = y$.

Thuật toán:

Gọi $m = [(p-1)^{1/2}]$ (lấy phần nguyên).

Bước 1: Tính $a^{mj} \bmod p$ với $0 \le j \le m-1$.

Bước 2: Sắp xếp các cặp $(j, a^{mj} \bmod p)$ theo $a^{mj} \bmod p$ và lưu vào danh sách L_1.

Bước 3: Tính $ya^{-i} \bmod p$ với $0 \le i \le m-1$.

Bước 4: Sắp xếp các cặp $(i, ya^{-i} \bmod p)$ theo $a^{mj} \bmod p$ và lưu vào danh sách L_2.

Bước 5: Tìm trong hai danh sách L_1 và L_2 xem có tồn tại cặp $(j, a^{mj} \bmod p)$ và $(i, ya^{-i} \bmod p)$ nào mà $a^{mj} \bmod p = ya^{-i} \bmod p$ (tọa độ thứ hai của hai cặp bằng nhau).

Bước 6: $x = (mj + i) \bmod (p-1)$. Kết quả này có thể kiểm chứng từ công thức $a^{mj} \bmod p = ya^{-i} \bmod p \Rightarrow a^{mj+i} \bmod p = y \bmod p \Rightarrow x = (mj + i) \bmod (p-1)$.

Độ phức tạp của thuật toán phụ thuộc vào m = $[(p-1)^{1/2}]$, với giá trị của m, chúng ta cần tính các phần tử thuộc hai danh sách L_1 và L_2, đều là các phép toán lũy thừa phụ thuộc vào j và i, i và j lại phụ thuộc vào m nên có thể nhận thấy là thuật toán này chỉ có thể áp dụng trong những trường hợp mà p nhỏ.

Thuật toán Pohlig-Hellman

Có những trường hợp đặc biệt mà bài toán Logarithm rời rạc có thể giải quyết với độ phức tạp nhỏ hơn $\boldsymbol{O}(p^{1/2})$, chẳng hạn như khi p – 1 chỉ có các ước nguyên tố nhỏ. Một thuật toán làm việc với các trường hợp như vậy đã được Pohlig và Hellman đưa ra vào năm 1978.

Giả sử $p - 1 = 2^n$.

Gọi a là phần tử nguyên thủy của Z_p^*, p là một số lẻ và $a^{(p-1)/2} \bmod p = -1$. Gọi m là số nguyên thuộc khoảng [0, p-2] mà chúng ta cần tìm để $y = a^m \bmod p$. Giả sử m được biểu diễn thành dạng nhị phân $m = m_0 + 2m_1 + 4m_2 + \ldots + 2^{n-1}m_{n-1}$. Khi đó:

$$y^{\frac{p-1}{2}} = (a^m)^{\frac{p-1}{2}} = (a^{m_0+2m_1+2^2m_2+\ldots+2^{n-1}m_{n-1}})^{\frac{p-1}{2}} = a^{m_0\frac{p-1}{2}} = \begin{cases} 1 \text{ nếu } m_0 = 0 \\ -1 \text{ nếu } m_0 = 1 \end{cases}$$

Việc tính $y^{(p-1)/2}$ mất nhiều nhất $2[\log_2 p]$ bước và sẽ cho ta m_0. Khi xác định được $y_1 = ya^{-m_0}$, ta lặp lại thao tác tương tự để tính m_1:

$$c_1^{\frac{p-1}{4}} = (a^{m_1+2m_2+\ldots+2^{n-2}m_{n-1}})^{\frac{p-1}{2}} = a^{m_1\frac{p-1}{2}} = \begin{cases} 1 \text{ nếu } m_1 = 0 \\ -1 \text{ nếu } m_1 = 1 \end{cases}$$

Quá trình tính toán cứ thể tiếp diễn cho tới khi chúng ta tìm được m_i. Độ phức tạp của thuật toán là: $n(2[\log_2 p] + 2) \sim \boldsymbol{O}((\log_2 p)^2)$.

3.4. Các hệ mã mật dựa trên các đường cong Elliptic

Hầu hết các sản phẩm và các chuẩn sử dụng các hệ mã khóa công khai để mã hóa và chữ ký điện tử hiện nay đều sử dụng hệ mã RSA. Tuy nhiên với sự phát triển của ngành thám mã và năng lực ngày càng tăng nhanh chóng của các hệ thống máy tính, độ dài khóa để đảm bảo an toàn cho hệ mã RSA cũng ngày càng tăng nhanh chóng, điều này làm giảm đáng kể hiệu năng của các hệ thống sử dụng hệ mã RSA, đặc biệt là với các ứng dụng thương mại điện tử trực tuyến hay các hệ thống realtime đòi hỏi thời gian xử lý nhanh chóng. Gần đây một hệ mã mới đã xuất hiện và có khả năng thay thế cho RSA, đó là các hệ mã khóa công khai dựa trên các đường cong Elliptic – ECC (Elliptic Curve Cryptography).

Điểm hấp dẫn nhất của các hệ mã dựa trên các đường cong Elliptic là nó cho phép đạt được tính an toàn tương đương với RSA trong khi kích thước khóa sử dụng lại nhỏ hơn rất nhiều, làm giảm số phép tính sử dụng khi mã hóa, giải mã và do đó đạt được hiệu năng và tốc độ cần thiết. Trên lý thuyết tính an toàn của ECC không cao bằng so với RSA và cũng khó giải thích một cách dễ hiểu hơn so với RSA hay Diffie-Hellman. Cơ sở toán học đầy đủ của các hệ mã dựa trên đường cong Elliptic vượt ra ngoài phạm vi của tài liệu này, trong phần này chúng ta sẽ chỉ xem xét các vấn đề cơ bản của các đường cong Elliptic và các hệ mã ECC.

3.4.1. Nhóm Abel

Nhóm Abel G, thường được ký hiệu là {G, •} là một tập hợp với một phép toán hai ngôi ký hiệu là •, kết qủa thực hiện của phép toán với hai phần tử a, b ∈ G, ký hiệu là (a • b) cũng là một phần tử thuộc G, tính chất này gọi là đóng đối với tập G. Đối với phép toán • các mệnh đề sau đều thỏa mãn:

(A1): ∀ a, b ∈ G thì (a • b) ∈G, tính đóng (Closure)

(A2): ∀ a, b, c ∈ G thì a • (b • c) = (a • b) • c, tính kết hợp (Associate)

(A3): Tồn tại e ∈ G: e • a = a • e = a ∀ a ∈ G, e được gọi là phần tử đơn vị của tập G.

(A4): ∀ a ∈ G, luôn ∃ a' ∈ G: a • a' = a' • a = e, a' là phần tử nghịch đảo của a.

(A5): ∀ a, b ∈ G: a • b = b • a, tính giao hoán (Commutative).

Rất nhiều các hệ mã khóa công khai dựa trên các nhóm Abel. Chẳng hạn, giao thức trao đổi khóa Diffie-Hellman liên quan tới việc nhân các cặp số nguyên khác không theo modulo q (nguyên tố). Các khóa được sinh ra bởi phép tính lũy thừa trên nhóm.

Đối với các hệ mã ECC, phép toán cộng trên các đường cong Elliptic được sử dụng là phép toán cơ bản. Phép nhân được định nghĩa là sự lặp lại của nhiều phép cộng: a x k = (a + a + … + a). Việc thám mã liên quan tới việc xác định giá trị của k với các thông tin công khai là a và (a x k).

Một đường cong Elliptic là một phương trình với hai biến và các hệ số. Các đường cong sử dụng cho các hệ mã mật có các biến và các hệ thống là các phần tử thuộc về một trường hữu hạn, điều này tạo thành một nhóm Abel. Trước hết chúng ta sẽ xem xét các đường cong Elliptic trên trường số thực.

3.4.2. Các đường cong Elliptic trên trường số thực

Các đường cong Elliptic không phải là các đường Ellipse. Tên gọi đường cong Elliptic được đặt vì loại đường cong này được mô tả bởi các phương trình bậc ba, tương tự như các phương trình được dùng để tính chu vi của một Ellipse. Ở dạng chung nhất phương trình bậc 3 biểu diễn một đường cong Elliptic có dạng:

$y^2 + axy + by = x^3 + cx^2 + dx + e.$

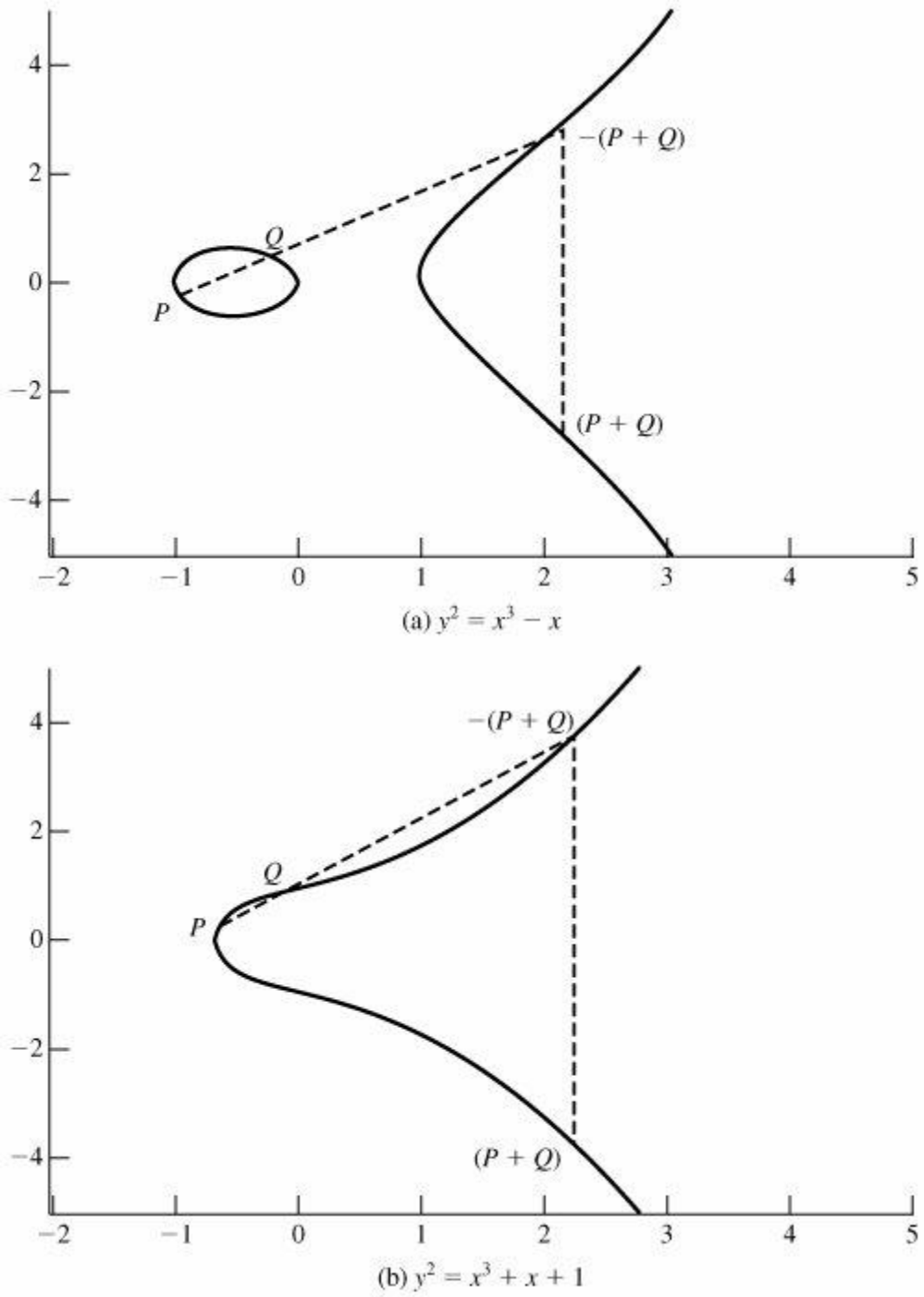
Trong đó a, b, c, d, e là các số thực, x và y là các biến thuộc trường số thực. Với mục đích để hiểu về các hệ mã ECC chúng ta chỉ xét các dạng đường cong Elliptic có dạng:

$y^2 = x^3 + ax + y$ (phương trình 1)

Các phương trình này được gọi là các phương trình bậc ba, trên các đường cong Elliptic chúng ta định nghĩa một điểm đặc biệt gọi là điểm ***O*** hay điểm tại vô cùng (point at infinity). Để vẽ đường cong Elliptic chúng ta cần tính các giá trị theo phương trình:

$$y = \sqrt{x^3 + ax + b}$$

Với mỗi giá trị cụ thể của a và b, sẽ cho chúng ta hai giá trị của y (một âm và một dương) tương ứng với một giá trị của x, các đường cong dạng này luôn đối xứng qua đường thẳng y = 0. Ví dụ về hình ảnh của một đường cong Elliptic:



Hình 4.4: Các đường cong Elliptic trên trường số thực

Chúng ta xem xét tập điểm E(a, b) chứa tất cả các điểm (x, y) thỏa mãn phương trình 1, cùng với điểm ***O***. Sử dụng các cặp (a, b) khác nhau chúng ta có các tập E(a, b) khác nhau. Sử dụng ký hiệu này ta có hình vẽ minh họa trên là biểu diễn của hai tập hợp E(1, 0) và E(1, 1) tương ứng.

3.4.3. Mô tả hình học của phép cộng trên các đường cong Elliptic

Với mỗi cặp (a, b) cụ thể chúng ta có thể thành lập một nhóm trên tập E(a, b) với các điều kiện sau:

$4a^3 + 27b^2 \neq 0$ (điều kiện 1).

Với điều kiện bổ sung này ta định nghĩa phép cộng trên đường cong Elliptic, mô tả về mặt hình học như sau: nếu ba điểm trên một đường cong Elliptic tạo thành một đường thẳng thì tổng của chúng bằng ***O***. Với định nghĩa này các luật của phép cộng trên đường cong Elliptic như sau:

1. ***O*** là phần tử trung hòa của phép cộng. $\forall$ P $\in$ E(a, b): P + ***O***= P. Trong các mệnh đề sau chúng ta giả sử P, Q ≠ ***O***.
2. P = (x, y) thì phần tử đối của P, ký hiệu là *P*, sẽ là (x, -y) và P + (*P*) = P P = *O*. P và P nằm trên một đường thẳng đứng
3. Để cộng hai điểm P và Q không có cùng hoàng độ x, vẽ một đường thẳng nối chúng và tìm giao điểm R. Dễ dàng nhận thấy chỉ có một điểm R như vậy, tổng của P và Q là điểm đối xứng với R qua đường thẳng y = 0.
4. Giao điểm của đường thẳng nối P với đối của P, tức *P*, được xem như cắt đường cong tại điểm vô cực và đó chính là ***O***.
5. Để nhân đôi một điểm Q, ta vẽ một tiếp tuyến tại Q với đường cong và tìm giao điểm S: Q + Q = 2Q = S.

Với 5 điều kiện này E(a, b) là một nhóm Abel.

3.4.4. Mô tả đại số về phép cộng

Trong phần này chúng ta sẽ trình bày một số kết quả cho phép tính toán trên các đường cong Elliptic. Với hai điểm phân biệt P = (x_P, y_P) và Q = (x_Q, y_Q) không phải là đối của nhau, độ dốc của đường nối l giữa chúng là Ä = (y_Q, y_P). Có chính xác một điểm khác mà l giao với đường cong, và đó chính là đối của tổng giữa P và Q. Sau một số phép toán đại số chúng ta có thể tính ra R = P + Q như sau:

$$x_R = \Delta^2 - y_P - x_Q$$

$$y_R = -y_P + \Delta(x_P - y_R)$$

Phép toán nhân đôi đối với P được tính như sau:

$$x_R = (\frac{3x_P^2 + a}{2y_P})^2 - 2x_P$$

$$y_R = (\frac{3x_P^2 + a}{2y_P})(x_P - x_R) - y_P$$

3.4.5. Các đường cong Elliptic trên Z_P

Các hệ mã ECC sử dụng các đường cong Elliptic với các biến và các hệ số giới hạn thuộc về một trường hữu hạn. Có hai họ các đường cong Elliptic có thể sử dụng với các hệ mã ECC: các đường cong nguyên tố trên Z_P và các đường cong nhị phân trên GF(2^m). Một đường cong nguyên tố trên Z_P, chúng ta sử dụng phương trình bậc ba mà các biến và các hệ số của nó đều là các giá trị nguyên nằm từ 0 tới p-1 và các phép tính được thực hiện theo modulo P. Trên đường cong nhị phân, các biến và các hệ số là các giá trị trên GF(2^n). và các tính toán được thực hiện trên GF(2^n). Các nghiên cứu về lý thuyết đã cho thấy các đường cong nguyên tố là phù hợp nhất cho các ứng dụng phần mềm vì những phức tạp trong tính toán đối với các đường cong nhị phân, nhưng đối với các ứng dụng phần cứng thì việc sử dụng các đường cong nhị phân lại tốt hơn vì cơ chế làm việc của các mạch, các con chíp rất phù hợp với các tính toán trên trường nhị phân.

Với các đường cong Elliptic trên Z_P chúng ta định nghĩa lại phương trình biểu diễn như sau:

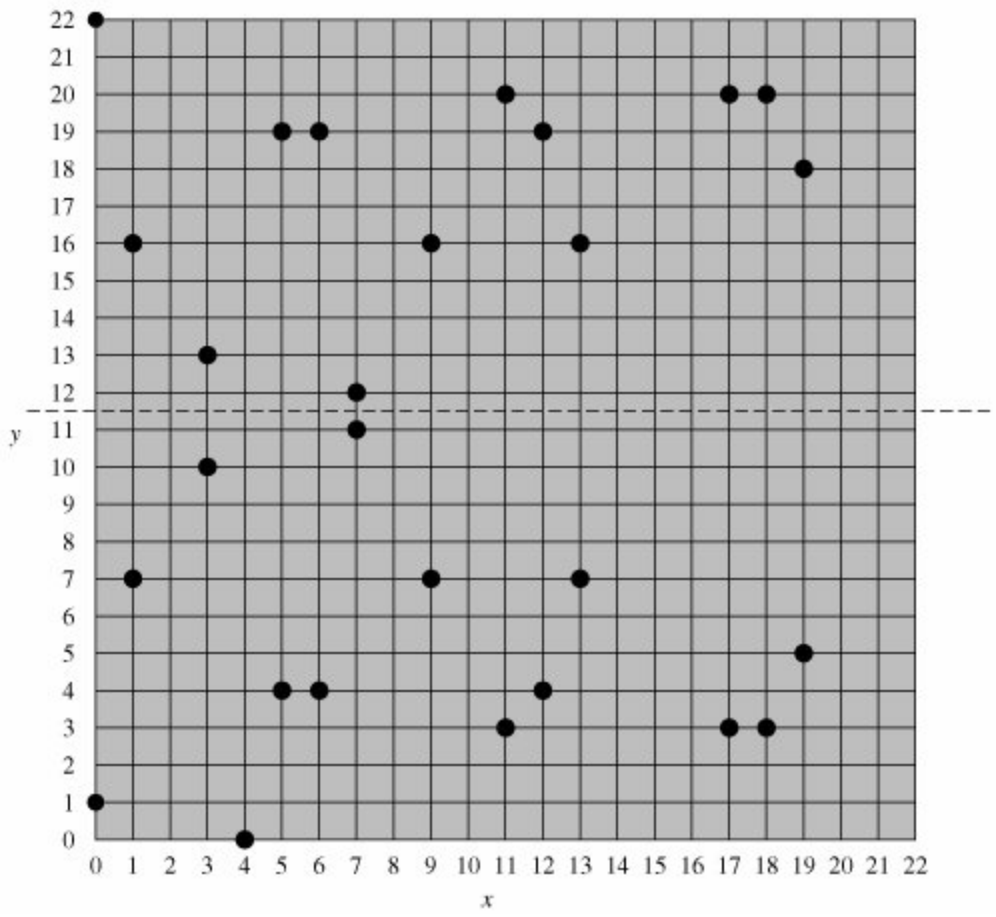
$y^2 \bmod p = (x^3 + ax + y) \bmod p$. (phương trình 2)

Chẳng hạn các giá trị a = 1, b = 1, x = 9, y = 9, y = 7, p = 23 thỏa mãn phương trình trên.

Các giá trị hệ số a, b và các biến số x, y đều thuộc Z_P. Tập $E_P(a, b)$ gồm tất cả các cặp (x, y) thỏa mãn phương trình phương trình 2.

Ví dụ với p = 23, a = b = 1, ta có tập $E_{23}(1, 1)$:

(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)



Bảng 4.2: Biểu diễn của tập E23(1, 1)

Các qui tắc về phép cộng cũng được định nghĩa tương tự đối với các đường cong Elliptic nguyên tố:

Điều kiện: $(4a^3 + 27b^2) \bmod p \neq 0$.

1. $P + \boldsymbol{O} = P$

2. Nếu $P = (x_P, y_P)$ thì $P + (x_P, y_P) = O$, điểm (x_P, y_P) được gọi là đối của P, ký hiệu là *P*. Chẳng hạn trên $E_{23}(1, 1)$, P = (13, 7) ta có *P* = (13, 7) nhưng 7 mod 23 = 16 nên *P* = (13, 16), cũng thuộc $E_{23}(1, 1)$.

3. Với hai điểm phân biệt $P = (x_P, y_P)$ và $Q = (x_Q, y_Q)$, $R = P + Q = (x_R, y_R)$ được định nghĩa như sau:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$

$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

Trong đó:

$$\lambda = \begin{cases} (\frac{y_Q - y_P}{x_Q - x_P}) \bmod p, (P \neq Q) \\ (\frac{3x_P^2 + a}{2y_P}) \bmod p, () p = Q) \end{cases}$$

4. Phép nhân được định nghĩa là tổng của các phép cộng, chẳng hạn 4P = P + P + P + P. Ví dụ với P = (3, 10) và Q = (9, 7) trên $E_{23}(1, 1)$ ta có:

$$\lambda = (\frac{7-10}{9-3}) \bmod 23 = (\frac{-3}{6}) \bmod 23 = (\frac{-1}{2}) \bmod 23 = 11 \text{ nên}$$

$x_R = (11^2 - 3 - 9) \bmod 23 = 17$

$y_R = (11(3 - 17) - 10) \bmod 23 = 20$. Nên P + Q = (17, 20).

Để tìm 2P ta tính:

$$\lambda = (\frac{3(3^2)+1}{2 \times 10}) \bmod 23 = (\frac{5}{20}) \bmod 23 = (\frac{1}{4}) \bmod 23 = 6$$

Chú ý là để thực hiện phép tính cuối cùng ta lấy phần tử nghịch đảo của 4 trên Z_{23} sau đó nhân với tử số là 1.

$x_R = (6^2(3 - 7) - 10) \bmod 23 = 30 \bmod 23 = 7$

$y_R = (6(3 - 7) - 10) \bmod 23 = 34 \bmod 23 = 12$

Kết luận: 2P = (7, 12).

Để xác định độ an toàn của các hệ mã mật dựa trên các đường cong Elliptic, người ta thường dựa trên một con số là số phần điểm trên một nhóm Abel hữu hạn, gọi là N, được định nghĩa trên một đường cong Elliptic. Trong trường hợp nhóm hữu hạn $E_P(a, b)$, ta có các cận của N là:

$p+1-2\sqrt{p} \leq N \leq p+1+2\sqrt{p}$, con số này xấp xỉ bằng số phần tử của Z_P (bằng p).

3.4.6. Các đường cong Elliptic dựa trên các trường hữu hạn GF(2^m)

Số phần tử của trường hữu hạn GF(2^m) là 2^m, các phép toán được trang bị trên GF(2^m) là phép toán cộng và phép toán nhân được thực hiện với các đa thức. Đối với các đường cong Elliptic dựa trên GF(2^m), chúng ta sử dụng một phương trình bậc ba với các biến và các tham số có giá trị thuộc GF(2^m), các phép tính được thực hiện tuân theo các phép toán trên GF(2^m).

1. Phương trình biểu diễn

So với các hệ mã mật dựa trên các đường cong trên Z_P, dạng biểu diễn của các hệ mã dựa trên GF(2^m) tương đối khác:

$y^2 + xy = x^3 + ax^2 + b$ (phương trình 3)

Trong đó các biến x, y và các hệ số a, b là các phần tử của GF(2^m) và các phép tính toán được thực hiện tuân theo các qui tắc trên GF(2^m).

Chúng ta ký hiệu $E_2^m(a, b)$ là tất cả các cặp số nguyên (x, y) thỏa mãn phương trình phương trình 3 và điểm vô cùng ***O***.

Ví dụ: chúng ta có thể sử dụng GF(2^4) với đa thức bất khả qui $f(x) = x^4 + x + 1$. Phần tử sinh của GF(2^4) là g thỏa mãn $f(g) = 0$, $g^4 = g + 1$, hay ở dạng nhị phân là 0010. Chúng ta có bảng lũy thừa của g như sau:

g0 = 0001	g4 = 0011	g8 = 0101	g12 = 1111
g1 = 0010	g5 = 0110	g9 = 1010	g13 = 1101
g2 = 0100	g6 = 1100	g10 = 0111	g14 = 1001
g3 = 1000	g7 = 1011	g11 = 1110	g15 = 0001

Chẳng hạn $g^5 = g^4 g = (g+1)g = g^2 + g = 0110$.

Xét đường cong Elliptic $y^2 + xy = x^3 + g^4x^2 + 1$, trong trường hợp này $a = g^4$ và $b = g^0 = 1$. Một điểm nằm trên đường cong là (g^5, g^3):

$(g^3)^2 + (g^5)(g^3) = (g^5)^3 + (g^4)(g^5)^2 + 1$

$\Leftrightarrow g^6 + g^8 = g^{15} + g^{14} + 1$

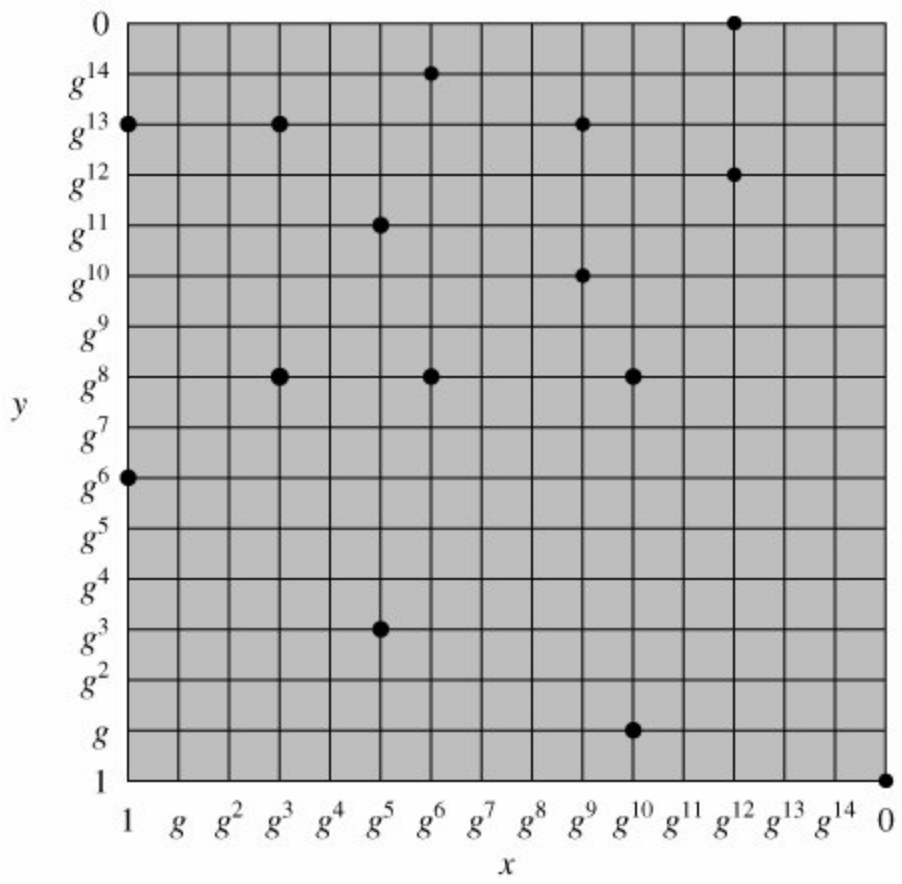
$\Leftrightarrow 1100 + 0101 = 0001 + 1001 + 0001$

$\Leftrightarrow 1001 = 1001$

Bảng sau là các điểm trên $E_2^4(g^4, 1)$:

(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})

Hình biểu diễn tương đương:



Hình 4.5: Hình biểu diễn $E_2^4(g^4, 1)$

Một nhóm Abel có thể định nghĩa dựa trên $E_2^m(a, b)$ với điều kiện b≠0. Các luật thực hiện với phép cộng, ∀ a, b ∈$E_2^m(a, b)$:

1. P + ***O*** = P
2. Nếu P = (x_P, y_P) thì P + (x_P, x_P + y_P) = ***O***. Điểm (x_P, x_P + y_P) là điểm đối của P, ký hiệu là *P*.
3. Nếu P = (x_P, y_P) và Q = (x_Q, y_Q) và P≠Q, *P*≠Q thì R = P + Q = (x_R, y_R) được xác định bằng các công thức sau:

$$x_R = \lambda^2 + \lambda + x_P + x_Q + a$$

$$y_R = \lambda(x_P + x_R) + x_R + y_P + a$$

Trong đó:

$$\lambda = \frac{y_Q + y_P}{x_Q + x_P}$$

4. Nếu P = (x_P, y_P) thì R = 2P = (x_R, y_R) được xác định bằng các công thức sau:

$$x_R = \lambda^2 + \lambda + a$$

$$y_R = x_P^2 + (\lambda + 1)x_R$$

Trong đó:

$$\lambda = x_P + \frac{y_P}{x_P}$$

3.4.7. Hệ mã mật dựa trên các đường cong Elliptic

Phép toán cộng trên đường cong Elliptic tương ứng với phép nhân theo modulo trong hệ mã RSA, còn phép toán nhân (cộng nhiều lần) trên đường cong Elliptic tương ứng với phép lũy thừa theo modulo trong hệ mã RSA. Tương tự như bài toán cơ sở của hệ mã RSA là bài toán phân tích ra dạng thừa số nguyên tố của một số nguyên lớn, các hệ mã dựa trên các đường cong Elliptic cũng có các bài toán cơ sở là một bài toán khó giải, gọi là bài toán Logarithm trên đường cong Elliptic:

Xét phương trình Q = kP trong đó P, Q $\in E_P(a, b)$ và k < p. Việc tính Q nếu biết P và k là một bài toán dễ (thực hiện theo các công thức). Nhưng việc xác định k với giá trị P, Q cho trước lại là bài toán khó.

Chúng ta xem xét ví dụ (Certicom Website www.certicom.com): $E_{23}(9, 17)$ được xác định bởi phương trình $y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23$.

Với Q = (4, 5) và P = (16, 5) thì k thỏa mãn Q = kP sẽ bằng bao nhiêu? Phương pháp đơn giản nhất là nhân P lên nhiều lần cho tới khi bằng Q:

P = (16, 5), 2P = (20, 20), 3P = 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).

Như vậy k = 9. Trên thực tế các hệ mã sẽ đảm bảo giá trị k là đủ lớn để phương pháp vét cạn như trên là không thể thực hiện được.

3.4.8. Phương pháp trao đổi khóa Diffie-Hellman dựa trên các đường cong Elliptic

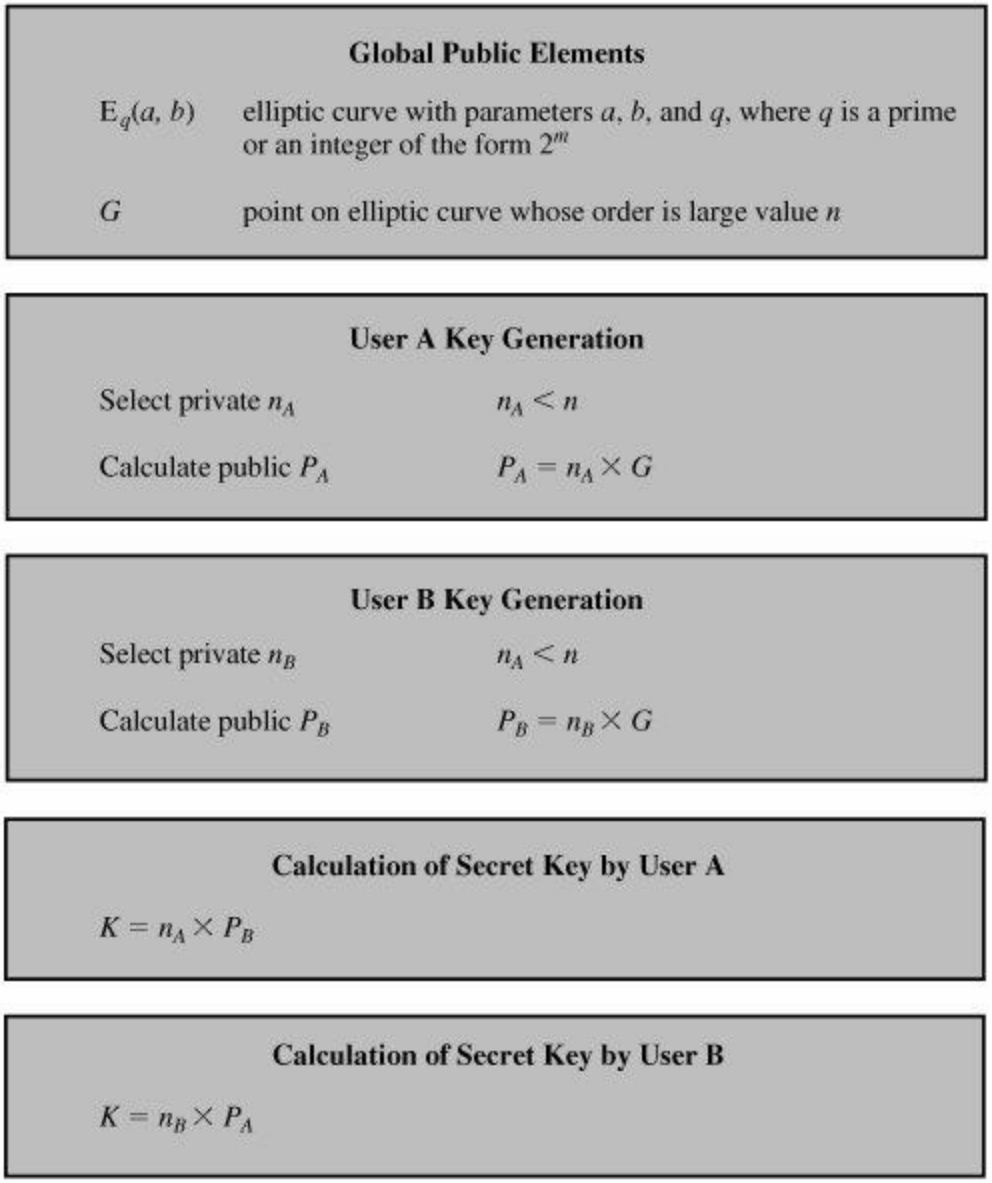
Ban đầu người ta chọn một số nguyên lớn q, có thể là một số nguyên tố p hay có dạng 2^m tương ứng với các phương trình biểu diễn và các tham số a, b. Việc lựa chọn này cho chúng ta tập hợp $E_q(a, b)$. Tiếp theo chọn một điểm G = $(x_1, y_1) \in E_P(a, b)$ có bậc n rất lớn, bậc n của điểm G là số nguyên nhỏ nhất thỏa mãn nG = ***O***. $E_q(a, b)$ và G là các tham số công khai cho hệ mã mật dựa trên đường cong Elliptic tương ứng với các tham số p, a, b.

Phương pháp trao đổi khóa giữa hai người dùng A và B có thể thực hiện như sau:

1. A chọn một số nguyên n_A nhỏ hơn n. Đó chính là khóa riêng của A. Sau đó sinh khóa công khai $P_A = n_A \times G$, khóa này là một điểm trên $E_q(a, b)$.
2. Tương tự B cũng chọn một khóa riêng n_B và tính khóa công khai P_B.
3. A sinh một khóa bí mật $K = n_A \times P_B$. B sinh khóa bí mật $K = n_B \times P_A$.

Dễ dàng kiểm chứng các khóa bí mật của A và B tính được đều bằng nhau: $n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$.

Hình minh họa các bước:



Hình 4.6: Phương pháp trao đổi khóa Diffie-Hellman dựa trên ECC

Để tấn công phương pháp trao đổi khóa trên, kẻ tấn công cần phải tính được giá trị k với các giá trị công khai là G và kG, và đây chính là bài toán Logarithm trên đường cong Elliptic, một bài toán khó.

Ví dụ: p = 211, $E_{211}(0, 4)$ tương ứng với phương trình biểu diễn $y^2 = x^3 + 4$, ta chọn G = (2, 2). Do 240G = ***O*** nên n = 240. A chọn khóa riêng là n_A = 121, khóa công khai tương ứng của A sẽ là P_A = 121(2, 2) = (115, 48). Khóa riêng của B là n_B = 203 nên khóa công khai cùa B là P_B = 203(2, 2) = (130, 203). Khóa bí mật (chia sẻ) giữa A và B là 121(130, 203) = 203(115, 48) = (161, 69).

3.4.9. Thuật toán mã hóa và giải mã

Có nhiều cách mã hóa/giải mã đã được nghiên cứu với các hệ mã trên các đường cong Elliptic, ở đây chúng ta sẽ xem xét cách đơn giản nhất. Thuật toán mã hóa ban đầu sẽ thực hiện phép biến đổi tiền xử lý từ input là một bản rõ m thành dạng một điểm P_m. Điểm P_m sẽ được mã hóa thành bản mã và sau đó giải mã. Thực chất việc tiền xử lý này không đơn giản vì không phải tất cả các tọa độ có dạng (x, y) đều thuộc $E_P(a, b)$. Có

nhiều cách khác nhau cho việc tiền xử lý này, chúng ta không bàn kỹ tới chúng ở đây nhưng thực tế là có một vài cách dễ hiểu để thực hiện việc đó.

Giống như đối với hệ trao đổi khóa, chúng ta cần một điểm G và một nhóm Elliptic $E_q(a, b)$ làm tham số. Mỗi người dùng A lựa chọn một khóa riêng n_A và sinh một khóa công khai $P_A = n_A \times G$.

Để mã hóa một thông điệp P_m để gửi tới cho B, A sẽ chọn một số nguyên dương ngẫu nhiên k và sinh bản mã C_m gồm một cặp điểm:

$C_m = \{kG, P_m + kP_B\}$.

Chú ý là ở đây A sử dụng khóa công khai của B. Để giải mã bản mã, B sẽ nhân điểm thứ nhất với khóa bí mật của B và lấy kết quả nhận được trừ đi điểm thứ hai:

$P_m + kP_B\, n_B(kG) = P_m + k(n_BG)\, n_B(kG) = P_m$.

A đã che đi giá trị của P_m bằng cách cộng kP_B vào P_m. Chỉ có duy nhất A biết giá trị k, nên thậm chí biết khóa công khai P_B, không ai có thể loại bỏ mặt nạ kP_B để tìm ra P_m. Tuy nhiên giá trị của C_m cũng gồm một đầu mối để B (người duy nhất giữ khóa riêng n_B) có thể dựa vào đầu mối đó mà tìm ra P_m.

Ví dụ: p = 751, $E_P(1, 188)$ tương ứng với phương trình $y^2 = x^3 + x + 188$, G = (0, 376). Giả sử A muốn gửi một thông điệp tương ứng với P_m = (562, 201) và A lựa chọn k = 386, khóa công khai của B là P_B = (201, 5). Chúng ta có 386(0, 376) = (676, 558) và (562, 201) + 386(201, 5) = (385, 328). Bản mã sẽ là C_m = {(676, 558), (385, 328)}.

3.4.10. Độ an toàn của các hệ mã mật dựa trên các đường cong Elliptic

Độ an toàn của các hệ mã ECC phụ thuộc vào việc xác định được giá trị của k dựa trên các giá trị kP và P. Bài toán này được gọi là bài toán Logarithm trên các đường cong Elliptic. Thuật toán nhanh nhất để giải bài toán này là thuật toán của Pollard. Bảng sau cho chúng ta sự so sánh tương quan giữa các hệ mã:

Symmetric Scheme (key size in bits)	**ECC-Based Scheme (size of n in bits)**	**RSA/DSA (modulus size in bits)**
56	112	512
80	160	1024
112	224	2048
128	256	3072
92	384	7680
256	512	15360
Nguồn: Certicom		

Bảng 4.3: Bảng so sánh các hệ mã ECC với hệ mã RSA

Có thể thấy là so với RSA, các hệ mã ECC có ưu thế hơn về độ dài khóa sử dụng, đặc biệt là khi chúng ta sử dụng các khóa có độ dài nhỏ thì ECC còn có ưu thế về tốc độ (số phép tính) xử lý trong mã hóa và giải mã.

4. Bài tập

Bài tập 4.1: Cho N = 1517. Hãy tính 13^{1435} mod N.

Bài tập 4.2: Trong hệ mã RSA có N = p * q = 103 * (2^{19} – 1) thì có thể sử dụng tối đa là bao nhiêu gía trị của e để làm khóa mã hóa, giải thích.

Bài tập 4.3: Trong hệ mã RSA có N = p*q = 103 * 113 sẽ có bao nhiêu trường hợp lộ bản rõ.

Bài tập 4.4: Trong hệ chữ ký điện tử ElGamma có p = 2^{31} – 1 khi ký lên một văn bản có thể sử dụng tối đa bao nhiêu gía trị k, giải thích.

Bài tập 4.5: Cho hệ mã ElGamma có p = 31, a = 11 và x = 6. Để mã hóa M = 18 người ta chọn k = 7. Hãy thực hiện tính toán và đưa ra bản mã kết quả.

Bài tập 4.6: Cho hệ RSA có n = 1363, biết phi(n) = 1288 hãy mã hóa bản rõ M = 2007.

Bài tập 4.7: Tương tự Câu 1 với n = 215629 và phi(n) = 214684 hãy giải mã bản mã M = 2007.

Bài tập 4.8: Giả sử có 4 tổ chức sử dụng 4 hệ mã RSA để truyền thông với nhau. Gọi N_1, N_2, N_3, N_4 lần lượt là các tham số tương ứng mà họ sử dụng và (N_i, N_j) = 1 $\forall$ i $\neq$ j và i, j $\in$ Z_5/{0}. Cả bốn hệ RSA này đều có số mũ lập mã là e = 3. Một thông điệp m sau khi mã hóa bằng 4 hệ mã trên nhận được 4 bản mã tương ứng là C_1, C_2, C_3, C_4. Hãy tìm m.

Bài tập 4.9: Cho hệ mã Knapsack có A = {11, 15, 30, 60}, M = 150 và u = 77.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Để mã hóa các thông điệp viết bằng tiếng Anh người ta dùng một hàm chuyển đổi từ các ký tự thành các xâu nhị phân như sau:

Ký tự	Xâu bít	Ký tự	Xâu bít	Ký tự	Xâu bít	Ký tự	Xâu bít
A	00000	H	00111	O	01110	V	10101
B	00001	I	01000	P	01111	W	10110
C	00010	J	01001	Q	10000	X	10111
D	00011	K	01010	R	10001	Y	11000
E	00100	L	01011	S	10010	Z	11001
F	00101	M	01100	T	10011		
G	00110	N	01101	U	10100		

Khi đó ví dụ xâu ABCD sẽ được chuyển thành 00000 00001 00010 00011 và cắt thành các xâu có độ dài 4 để thực hiện mã hóa. Kết quả thu được bản mã là một dãy các số $\in$ Z_M. Hãy thực hiện mã hóa xâu P = “ANTI”.

c) Giả sử bản mã thu được là C = <120, 105, 105, 0, 60, 75, 30, 22, 22, 30>. Hãy thực hiện giải mã bản mã trên để thu được thông điệp ban đầu.

Bài tập 4.10: Cho hệ mã Knapsack có A = {7, 13, 31, 53}, M = 173 và u = 97.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.

b) Để mã hóa các thông điệp viết bằng tiếng Anh người ta dùng một hàm chuyển đổi từ các ký tự thành các xâu nhị phân như sau:

Ký tự	Xâu bít	Ký tự	Xâu bít	Ký tự	Xâu bít	Ký tự	Xâu bít
A	00000	H	00111	O	01110	V	10101
B	00001	I	01000	P	01111	W	10110
C	00010	J	01001	Q	10000	X	10111
D	00011	K	01010	R	10001	Y	11000
E	00100	L	01011	S	10010	Z	11001
F	00101	M	01100	T	10011		
G	00110	N	01101	U	10100		

Khi đó ví dụ xâu ABCD sẽ được chuyển thành 00000 00001 00010 00011 và cắt thành các xâu có độ dài 4 để thực hiện mã hóa. Kết quả thu được bản mã là một dãy các số $\in Z_M$. Hãy thực hiện mã hóa xâu P = "AUNT".

c) Giả sử bản mã thu được là C = < 67,160, 66, 66, 0, 116, 4, 111, 0, 17>. Hãy thực hiện giải mã bản mã trên để thu được thông điệp ban đầu.

Bài tập 4.11: Cho hệ mã Knapsack có A = {2, 3, 7, 13, 29, 57}, M = 151 và u = 71.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Để mã hóa các thông điệp viết bằng tiếng Anh người ta dùng một hàm chuyển đổi từ các ký tự thành các xâu nhị phân như sau:

Ký tự	Xâu bít	Ký tự	Xâu bít	Ký tự	Xâu bít	Ký tự	Xâu bít
A	00000	H	00111	O	01110	V	10101
B	00001	I	01000	P	01111	W	10110
C	00010	J	01001	Q	10000	X	10111
D	00011	K	01010	R	10001	Y	11000
E	00100	L	01011	S	10010	Z	11001
F	00101	M	01100	T	10011		
G	00110	N	01101	U	10100		

Khi đó ví dụ xâu ABCDEF sẽ được chuyển thành 00000 00001 00010 00011 00100 00101 và cắt thành các xâu có độ dài 6 để thực hiện mã hóa. Kết quả thu được bản mã là một dãy các số $\in Z_M$. Hãy thực hiện mã hóa xâu P = "ANSWER".

c) Giả sử bản mã thu được là C = <44, 40, 121, 104, 0>. Hãy thực hiện giải mã bản mã trên để thu được thông điệp ban đầu.

Bài tập 4.12: Cho hệ mã RSA có p = 31, q = 41, e = 271.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Để mã hóa các thông điệp được viết bằng tiếng Anh người ta dùng một hàm chuyển đổi các ký tự thành các số thập phân có hai chữ số như sau:

Ký tự	A	B	C	D	E	F	G	H	I	J	K	L	M
Mã hóa	00	01	02	03	04	05	06	07	08	09	10	11	12
Ký tự	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Mã hóa	13	14	15	16	17	18	19	20	21	22	23	24	25

Khi đó ví dụ xâu ABC sẽ được chuyển thành 00 01 02 và sau đó cắt thành các số có 3 chữ số 000 (bằng 0) và 102 để mã hóa. Bản mã thu được là một tập các số $\in Z_N$. Hãy thực hiện mã hóa xâu P = ”SERIUS”.

c) Giả sử bản mã thu được là C = <201, 793, 442, 18> hãy thực hiện giải mã để tìm ra thông điệp bản rõ ban đầu.

Bài tập 4.13: Cho hệ mã RSA có p = 29, q = 43, e = 11.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Để mã hóa các thông điệp được viết bằng tiếng Anh người ta dùng một hàm chuyển đổi các ký tự thành các số thập phân có hai chữ số như sau:

Ký tự	A	B	C	D	E	F	G	H	I	J	K	L	M
Mã hóa	00	01	02	03	04	05	06	07	08	09	10	11	12
Ký tự	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Mã hóa	13	14	15	16	17	18	19	20	21	22	23	24	25

Khi đó ví dụ xâu ABC sẽ được chuyển thành 00 01 02 và sau đó cắt thành các số có 3 chữ số 000 (bằng 0) và 102 để mã hóa. Bản mã thu được là một tập các số $\in Z_N$. Hãy thực hiện mã hóa xâu P = ”TAURUS”.

c) Giả sử bản mã thu được là C = <1, 169, 1206, 433> hãy thực hiện giải mã để tìm ra thông điệp bản rõ ban đầu.

Bài tập 4.14: Cho hệ mã RSA có n = 1363, e = 57.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Giả sử bản rõ P = 102 hãy mã hóa và đưa ra bản mã C.

c) Giả sử hệ mã trên được dùng làm hệ chữ ký điện tử, hãy tính chữ ký với thông điệp M = 201.

Bài tập 4.15: Cho hệ mã ElGamma có p = 83, a = 5 là một phần tử nguyên thuỷ của Z_P^*, x = 37.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Để mã hóa bản rõ P = 72 người ta chọn k = 23, hãy mã hóa và đưa ra bản mã.
c) Hãy tìm tất cả các phần tử nguyên thuỷ của Z_P^*.

Bài tập 4.16: Cho hệ mã mật ElGamma có p = 1187, a = 79 là một phần tử nguyên thuỷ của Z_P^*, x = 113.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Để mã hóa các thông điệp được viết bằng tiếng Anh người ta dùng một hàm chuyển đổi các ký tự thành các số thập phân có hai chữ số như sau:

Ký tự	A	B	C	D	E	F	G	H	I	J	K	L	M
Mã hóa	00	01	02	03	04	05	06	07	08	09	10	11	12
Ký tự	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Mã hóa	13	14	15	16	17	18	19	20	21	22	23	24	25

Khi đó ví dụ xâu ABC sẽ được chuyển thành 00 01 02 và sau đó cắt thành các số có 3 chữ số 000 (bằng 0) và 102 để mã hóa. Bản mã thu được là một tập các cặp số (C1, C2) $\in Z_P$. Hãy thực hiện mã hóa xâu m = "TAURUS" với các giá trị 13 < k < 19.

c) Giả sử thu được bản mã là một tập các cặp (C1, C2) là <(358, 305), (1079, 283), (608, 925),(786, 391)>. Hãy giải mã và đưa ra thông điệp ban đầu.

Bài tập 4.17: Cho bản mã nhận được bằng cách sử dụng một hệ mã RSA như sau:

11437 6198 16611 2405 18636 2679 12205 24142 6375 2134

16611 2405 9529 7260 7834 15094 4667 24027 762 5878

5206 16683 5359 10888 4168 3536 23229 20351 15580 6704

7977 374 6525 4287 14402 527 12887 21628 11884 9402

15470 1339 10420 18051 23125 7747 135 22007 20049 9984

13199 15176 1379 8313 19574 7989 22869 406 10057 21758

3918 23991 14237 7989 3947 19529 15728 5601 3527 7200

7601 13282 21160 6291 15994 7785 8982 3045 6596 16796

4663 2405 20302 11929 17125 14533 21001 8351 11571 22082

11040 8687 6704 3330 5630 19650 13024

Khóa công khai có n = 24637 và e = 3.

a) Hãy xác định p, q và d.
b) Giải mã bản mã để nhận được bản rõ (là các số trên Z24637).
c) Chuyển bản rõ nhận được thành dạng văn bản tiếng Anh, biết rằng mỗi số nguyên trên Z24637 biểu diễn một bộ 3 chữ cái theo qui tắc sau:

$$DOG \rightarrow 3 \times 26^2 + 14 \times 26 + 6 = 2398$$
$$CAT \rightarrow 2 \times 26^2 + 0 \times 26 + 19 = 1371$$
$$ZZZ \rightarrow 25 \times 26^2 + 25 \times 26 + 25 = 17575$$

Bài tập 3.18: Cho hệ mã ElGamal có p = 71 và a = 7.

a) Giả sử khóa công khai của B là $Y_B = 3$ và A chọn số ngẫu nhiên k = 2, hãy xác định bản mã tương ứng với bản mã M = 30.
b) Giả sử A chọn một giá trị ngẫu nhiên k khác và bản mã tương ứng với M = 30 bây giờ là C = (59, C_2). Hãy xác định C_2?

Bài tập 3.19: Cho hệ mã dựa trên đường cong Elliptic có các tham số là $E_{11}(1, 6)$ và G = (2, 7). Khóa bí mật của B là $n_B = 7$.

a) Hãy xác định khóa công khai của B?
b) Giả sử cần mã hóa bản rõ P_m = (10, 9) và số ngẫu nhiên k = 3. Hãy xác định bản mã C_m.
c) Minh họa quá trình giải mã với C_m nhận được ở phần b.

Sử dụng một trong các ngôn ngữ lập trình C, C++, Java hoặc C# để làm các bài tập sau:

Bài tập 3.20: Viết chương trình cài đặt thuật toán mã hóa và giải mã của hệ mã Knapsack.

Bài tập 3.21: Viết chương trình cài đặt thuật toán mã hóa và giải mã của hệ mã RSA.

Bài tập 3.22: Viết chương trình cài đặt thuật toán mã hóa và giải mã của hệ mã El Gammal.

Bài tập 3.23: Viết chương trình mã hóa và giải mã File với thuật toán mã hóa và giải mã RSA.

Bài tập 3.24: Viết chương trình truyền file qua hệ thống mạng sử dụng thuật toán mã hóa RSA.

Bài tập 3.25: Viết chương trình chia sẻ file trên mạng cục bộ sử dụng hệ mã RSA.

Bài tập 3.26: Viết chương trình phân phối khóa dựa trên hệ mã RSA.

CHƯƠNG V: CHỮ KÝ ĐIỆN TỬ VÀ HÀM BĂM

1. Chữ ký điện tử

1.1. Khái niệm về chữ ký điện tử

Kể từ khi con người phát minh ra chữ viết, các chữ ký thường luôn được sử dụng hàng ngày, chẳng hạn như ký một biên nhận trên một bức thư nhận tiền từ ngân hàng, ký hợp đồng hay một văn bản bất kỳ nào đó. Chữ ký viết tay thông thường trên tài liệu thường được dùng để xác định người ký nó.

Sơ đồ chữ ký điện tử là một phương pháp ký một văn bản hay lưu bức điện dưới dạng điện tử. Chẳng hạn một bức điện có chữ ký được lưu hành trên mạng máy tính. Chữ ký điện tử từ khi ra đời đã có nhiều ứng dụng rộng rãi trong các giao dịch thương mại, từ việc xác minh chữ ký cho đến các thẻ tín dụng, các sơ đồ định danh và các sơ đồ chia sẻ bí mật ... Sau đây, chúng ta sẽ tìm hiểu một số sơ đồ chữ ký quan trọng. Song trước hết, chúng ta sẽ thảo luận một vài điểm khác biệt cơ bản giữa chữ ký thông thường và chữ ký điện tử.

Đầu tiên là vấn đề ký một tài liệu. Với chữ ký thông thường nó là một phần vật lý của tài liệu. Tuy nhiên, một chữ ký điện tử không gắn theo kiểu vật lý vào bức điện nên thuật toán được dùng phải là “không nhìn thấy” theo cách nào đó trên bức điện.

Thứ hai là vấn đề kiểm tra. Chữ ký thông thường được kiểm tra bằng cách so sánh nó với các chữ ký xác thực khác. Ví dụ, ai đó ký một tấm séc để mua hàng, người bán sẽ so sánh chữ ký trên mảnh giấy đó với chữ ký nằm ở mặt sau thẻ tín dụng để kiểm tra. Mặt khác, chữ ký số có thể kiểm tra bằng một thuật toán kiểm tra một cách công khai. Như vậy, bất kỳ ai cũng có thể kiểm tra được chữ ký điện tử. Việc sử dụng một sơ đồ ký an toàn có thể ngăn chặn được khả năng giả mạo.

Sự khác biệt cơ bản giữa chữ ký điện tử và chữ ký thông thường là ở chỗ: một bản copy tài liệu có chữ ký được đồng nhất với bản gốc. Nói cách khác, tài liệu có chữ ký trên giấy thường có thể khác biệt với bản gốc điều này để ngăn chặn một bức điện được ký khỏi bị dùng lại. Ví dụ, nếu B ký một bức điện xác minh cho A rút 100$ từ tài khoản của mình, anh ta chỉ muốn A có khả năng làm điều đó một lần. Vì thế, bản thân bức điện phải chứa thông tin để khỏi bị dùng lại, chẳng hạn như dùng dịch vụ gán nhãn thời gian (Time Stamping Service).

Một sơ đồ chữ ký điện tử thường chứa hai thành phần: thuật toán ký sig() và thuật toán xác minh ver(). B có thể ký một bức điện x dùng thuật toán ký an toàn (bí mật). Kết quả chữ ký y = sig(x) nhận được có thể được kiểm tra bằng thuật toán xác minh công khai ver(y). Khi cho trước cặp (x, y), thuật toán xác minh cho giá tri TRUE hay FALSE tuỳ thuộc vào việc chữ ký được xác thực như thế nào.

Vậy thế nào là chữ ký điện tử? Chúng ta có một số định nghĩa như sau:

- Là một định danh điện tử được tạo ra bởi máy tính được các tổ chức sử dụng nhằm đạt được tính hiệu quả và có hiệu lực như là các chữ ký tay.
- Là một cơ chế xác thực hóa cho phép người tạo ra thông điệp đính kèm một mã số vào thông điệp giống như là việc ký một chữ ký lên một văn bản bình thường.

Các chữ ký điện tử được sinh và sử dụng bởi các hệ chữ ký (sơ đồ) điện tử, dưới đây là định nghĩa một hệ chữ ký điện tử.

Định nghĩa:

Một sơ đồ chữ ký điện tử là bộ 5 $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ thoả mãn các điều kiện dưới đây:

1) *$\mathcal{P}$ là tập hữu hạn các bức điện (thông điệp, bản rõ) có thể.*

2) *$\mathcal{A}$ là tập hữu hạn các chữ ký có thể.*

3) *$\mathcal{K}$ là tập không gian khoá (tập hữu hạn các khoá có thể).*

4) *Với mỗi khoá $K \in \mathcal{K}$ tồn tại một thuật toán ký $sig_K \in \mathcal{S}$ và một thuật toán xác minh $ver_K \in \mathcal{V}$. Mỗi sig_K: $\mathcal{P} \rightarrow \mathcal{A}$ và ver_K: $\mathcal{P} x \mathcal{A} \rightarrow$ {TRUE, FALSE} là những hàm sao cho mỗi bức điện $x \in \mathcal{P}$ và mỗi chữ ký $y \in \mathcal{A}$ thoả mãn phương trình dưới đây:*

$Ver(x, y) = \begin{cases} \\ \\ \end{cases}$	TRUE	nếu	$y = sig(x)$	
	FALSE	nếu	$y \neq sig(x)$.	[5]

Với mỗi $K \in \mathcal{K}$, hàm sig_K và ver_K là các hàm đa thức thời gian. Hàm ver_K sẽ là hàm công khai còn hàm sig_K là bí mật. Không thể dễ dàng tính toán để giả mạo chữ ký của B trên bức điện x, nghĩa là với x cho trước chỉ có B mới có thể tính được y để ver(x, y) = TRUE. Một sơ đồ chữ ký không thể an toàn vô điều kiện vì một người C nào đó có thể kiểm tra tất cả chữ số y trên bức điện x nhờ dùng thuật toán ver() công khai cho tới khi anh ta tìm thấy chữ ký đúng. Vì thế, nếu có đủ thời gian, C luôn có thể giả mạo chữ ký của B. Như vậy mục đích của chúng ta là tìm các sơ đồ chữ ký điện tử an toàn về mặt tính toán.

Chú ý rằng ai đó có thể giả mạo chữ ký của B trên một bức điện "ngẫu nhiên" x bằng cách tính $x = e_K(y)$ với y nào đó; khi đó $y = sig_K(x)$. Một biện pháp xung quanh vấn đề khó khăn này là yêu cầu các bức điện chứa đủ phần dư để chữ ký giả mạo kiểu này không phù hợp với toàn bộ nội dung của bức điện x trừ một xác suất rất nhỏ. Có thể dùng các hàm Băm (hash function) như MD4, MD5 trong việc tính kết nối các sơ đồ chữ ký điện tử sẽ loại trừ phương pháp giả mạo này (sẽ trình bày trong các phần sau của tài liệu).

1.2. Hệ chữ ký RSA

Dựa vào ưu điểm của hệ mã RSA, nếu thiết lập được sơ đồ chữ ký dựa trên bài toán phân tích ra thừa số nguyên tố thì độ an toàn của chữ ký sẽ rất cao. Việc thiết lập sơ đồ xác thực chữ ký RSA rất đơn giản, ta chỉ cần đảo ngược hàm mã hoá và giải mã. Sau đây là sơ đồ chữ ký RSA.

*Cho n = p*q, trong đó p, q là các số nguyên tố. Đặt $\mathcal{P} = \mathcal{A} = Z_n$ và định nghĩa:*

*$\mathcal{K}$ = {(n, p, q, a, b): n=p*q, p và q là các số nguyên tố, $ab \equiv 1 \pmod{\phi(n)}$}.*

Các giá trị n và b là công khai; còn p, q, a là bí mật.

Với K = (n, p, q, a, b), ta xác định:

$sig_K(x) = x^a \bmod n$

và

$ver_K(x,y) = TRUE \Leftrightarrow x \equiv y^b \pmod n$ *với* $x, y \in Z_n$. [5]

Thông thường, chữ ký được kết hợp với hàm mã hoá công khai. Giả sử A muốn gửi một bức điện đã được mã hoá và đã được ký đến cho B. Với bản rõ x cho trước, A sẽ tính toán chữ ký của mình $y = sig_A(x)$ và sau đó mã hoá cả x và y sử dụng khoá công khai e_B của B, kết quả nhận được là $z = e_B(x, y)$. Bản mã z sẽ được gửi tới B, khi B nhận được z, đầu tiên anh ta giải mã với hàm giải mã d_B của mình để nhận được (x, y). Sau đó anh ta dùng hàm xác minh công khai của A để kiểm tra xem $ver_A(x,y) = TRUE$ hay không.

Song nếu đầu tiên A mã hoá x , rồi sau đó mới ký lên bản mã nhận được thì sao? Khi đó, A sẽ tính:

$y = sig_A(e_B(x))$

A sẽ truyền cặp (z, y) tới B, B sẽ giải mã z và nhận được x, sau đó xác minh chữ ký y trên x nhờ dùng ver_A. Một vấn đề nảy sinh nếu A truyền (x, y) kiểu này thì một người thứ ba C có thể thay chữ ký y của A bằng chữ ký của chính mình:

$y' = sig_C(e_B(x))$

Chú ý rằng, C có thể ký lên bản mã $e_B(x)$ ngay cả khi anh ta không biết bản rõ x. Khi đó nếu C truyền (z, y') đến B, chữ ký của C được B xác minh bằng ver_C và do đó, B cho rằng bản rõ x xuất phát từ C. Do khó khăn này, hầu hết người sử dụng được khuyến nghị "ký trước khi mã".

1.3. Hệ chữ ký ElGammal

Hệ chữ ký ElGammal được đưa ra vào 1985. Một phiên bản sửa đổi hệ này được Học viện Quốc gia tiêu chuẩn và kỹ thuật (NIST) đưa ra như một chuẩn của chữ ký điện tử. Hệ chữ ký ElGammal được thiết kế riêng biệt cho mục đích chữ ký, trái ngược với RSA thường được sử dụng cho cả mục đích mã hoá công khai và chữ ký. Hệ chữ ký ElGammal là không xác định, nghĩa là có rất nhiều giá trị chữ ký cho cùng một bức điện cho trước. Thuật toán xác minh phải có khả năng nhận bất kỳ giá trị chữ ký nào như là việc xác thực. Sơ đồ chữ ký ElGammal được miêu tả như sau:

Cho p là một số nguyên tố như là bài toán logarit rời rạc trong Z_p, $\alpha \in Z_p^*$ *là một phần tử nguyên tử và* $P = Z_p^*$, $A = (Z_p^*)^* Z_{p-1}$, *và định nghĩa:*

$K = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod p\}$

trong đó giá trị p, α *và* β *là công khai, còn a là bí mật.*

Với $K = (p, \alpha, a, \beta)$ *và chọn một số ngẫu nhiên* $k \in Z_{p-1}^*$, *định nghĩa:*

$sig_K(x, k) = (\gamma, \delta)$

trong đó: $\gamma = \alpha^k \bmod p$

$\delta = (x - a*\gamma)k^{-1} \bmod (p - 1)$.

Với $x, \gamma \in Z_p^*$ *và* $\delta \in Z_{p-1}$, *định nghĩa:*

$ver(x, \gamma, \delta) = TRUE \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod p$. [5]

Nếu chữ ký là đúng thì việc xác nhận thành công khi:

$\beta^{\gamma}\gamma^{\delta} \equiv \alpha^{a\gamma}\alpha^{k\delta} \pmod p$

$\equiv \alpha^{x} \pmod p$.

trong đó: $a\gamma + k\delta \equiv x \pmod{p-1}$.

B sẽ tính toán chữ ký bằng việc sử dụng cả giá trị bí mật a (một phần của khoá) và số bí mật ngẫu nhiên k (giá trị để ký bức điện). Việc xác minh có thể thực hiện được chỉ với các thông tin được công khai:

Ví dụ:

Chúng ta chọn p = 467, α = 2, a = 127. Ta tính: $\beta = \alpha^{a} \bmod p = 2^{127} \bmod 467 = 132$.

Bây giờ B muốn ký lên bức điện x = 100 và anh ta chọn một giá trị ngẫu nhiên k = 213 (chú ý là UCLN(213, 466) = 1 và $213^{-1} \bmod 466 = 431$). Sau đó tính:

$\gamma = 2^{213} \bmod 467 = 29$

$\delta = (100 - 127*29)431 \bmod 466 = 51$.

Bất cứ ai cũng có thể kiểm tra chữ ký này bằng cách tính:

$132^{29}29^{51} \equiv 189 \pmod{467}$

$2^{100} \equiv 189 \pmod{467}$.

Giả sử kẻ thứ ba C muốn giả mạo chữ ký của B trên bức điện x mà không biết số bí mật a. Nếu C chọn một giá trị γ và cố gắng tìm δ, anh ta phải tính một hàm logarit rời rạc $\log_{\gamma}\alpha^{x}\beta^{-\gamma}$. Mặt khác, nếu đầu tiên anh ta chọn δ để cố gắng tìm γ thì anh ta phải tính $\beta^{\gamma}\gamma^{\delta} = \alpha^{x} \pmod p$. Cả hai việc này đều không thể thực hiện được.

Tuy nhiên có một lý thuyết mà C có thể ký lên một bức điện ngẫu nhiên bằng cách chọn đồng thời γ, δ và x. Cho i, j là số nguyên với $0 \le i, j \le p - 2$, và UCLN(j, p - 1) = 1. Sau đó tính:

$\gamma = \alpha^{i}\beta^{j} \bmod p$

$\delta = -\gamma j^{-1} \pmod{p-1}$

$x = -\gamma ij^{-1} \pmod{p-1}$.

Như vậy, ta xem (γ, δ) là giá trị chữ ký cho bức điện x. Việc xác minh sẽ thực hiện như sau:

$\beta^{\gamma}\gamma^{\delta} \equiv \beta^{\alpha^{i}\beta^{j}}(\alpha^{i}\beta^{j})^{-\alpha^{i}\beta^{j}j^{-1}} \pmod p$

$\equiv \beta^{\alpha^{i}\beta^{j}}\alpha^{-ij^{-1}\alpha^{i}\beta^{j}}\beta^{-\alpha^{i}\beta^{j}} \pmod p$

$\equiv \alpha^{-ij^{-1}\alpha^{i}\beta^{j}} \pmod p$

$\equiv \alpha^{-\gamma ij^{-1}} \pmod p$

$\equiv \alpha^{x} \pmod p$.

Ví dụ:

Như ví dụ trên, ta chọn p = 467, α = 2, β = 132. Kể thứ ba C sẽ chọn i = 99 và j = 179. Anh ta sẽ tính:

$\gamma =$	$2^{99}132^{179} \bmod 467 = 117$
$\delta =$	$-117*151 \bmod 466 = 41$
x =	$99*44 \bmod 466 = 331$

Cặp giá trị (117, 41) là giá trị chữ ký cho bức điện 331. Việc xác minh được thực hiện như sau:

$132^{117}117^{41} \equiv 303 \pmod{467}$

$2^{331} \equiv 303 \pmod{467}$.

Một phương pháp thứ hai có thể giả mạo chữ ký là sử dụng lại chữ ký của bức điện trước đó, nghĩa là với cặp (γ, δ) là giá trị chữ ký của bức điện x, nó sẽ được C ký cho nhiều bức điện khác. Cho h, i và j là các số nguyên, trong đó $0 \le i, j, h \le p-2$ và UCLN($h\gamma - j\delta$, p-1) = 1.

$\lambda = \gamma^h \alpha^i \beta^j \bmod p$

$\mu = \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1)$

$x' = \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1)$.

Ta có thể kiểm tra: $\beta^\lambda \lambda^\mu = \alpha^{x'} \bmod p$. Và do đó, (λ, μ) là cặp giá trị chữ ký của bức điện x'.

Điều thứ ba là vấn đề sai lầm của người ký khi sử dụng cùng một giá trị k trong việc ký hai bức điện khác nhau. Cho (γ, δ_1) là chữ ký trên bức điện x_1 và (γ, δ_2) là chữ ký trên bức điện x_2. Việc kiểm tra sẽ thực hiện:

$\beta^\gamma \gamma^{\delta_1} \equiv \alpha^{x_1} \pmod p$

$\beta^\gamma \gamma^{\delta 2} \equiv \alpha^{x2} \pmod p$.

Do đó: $\alpha^{x_1 - x_2} \equiv \gamma^{\delta_1 - \delta_2} \pmod p$.

Đặt $\gamma = \alpha^k$, khi đó: $x_1 - x_2 = k(\delta_1 - \delta_2) \pmod{p-1}$.

Bây giờ đặt d = UCLN($\delta_1 - \delta_2$, p - 1). Vì d | ($\delta_1 - \delta_2$) và d | (p - 1) nên nó cũng chia hết cho ($x_1 - x_2$). Ta đặt tiếp:

$$x' = \frac{x_1 - x_2}{d}$$

$$\delta' = \frac{\delta_1 - \delta_2}{d}$$

$$p' = \frac{p-1}{d}$$

Cuối cùng, ta được: $x' \equiv k\delta' \pmod{p'}$. Vì UCLN($\delta'$, p') = 1 nên ta có:

$\varepsilon = (\delta')^{-1} \bmod p'$

Như vậy, giá trị k sẽ được xác định như sau:

$k = x'\varepsilon \pmod{p'} = x'\varepsilon + ip' \pmod{p}$

Với $0 \le i \le d-1$, ta có thể tìm được giá trị k duy nhất bằng hàm kiểm tra:

$\gamma \equiv \alpha^k \bmod p$.

1.4. Chuẩn chữ ký điện tử (Digital Signature Standard)

1.4.1. Thuật toán chữ ký điện tử (Digital Signature Algorithm)

Tháng 8/1991, NIST đã đưa ra thuật toán chữ ký điện tử (DSA) là cơ sở cho chuẩn chữ ký điện tử. Đây là một biến thể của thuật toán ElGammal.

1) Chọn một số nguyên tố q với $2^{159} < q < 2^{160}$.

2) Chọn t sao cho $0 \le t \le 8$ và chọn một số nguyên tố p, trong đó $2^{511+64t} < p < 2^{512+64t}$ và q phải chia hết (p-1) (hay q là một ước nguyên tố của p-1).

3) Bây giờ, tạo ra một số α duy nhất cho q trong trường Z_p^*.

- *Chọn một giá trị $g \in Z_p^*$ và tính $\alpha = g^{(p-1)/q} \bmod p$.*
- *Nếu α = 1 thì quay lại bước trên. (chọn lại giá trị g cho phù hợp)*

4) Chọn một số nguyên ngẫu nhiên a để $1 \le a \le q-1$.

5) Tính $y = \alpha^a \bmod p$.

6) Như vậy , khoá để ký là (p, q, α, y) được công khai và a là khoá bí mật.

1.4.2. Chuẩn chữ ký điện tử

Chuẩn chữ ký điện tử (DSS) được sửa đổi từ hệ chữ ký ElGammal. Nó được công bố tại hội nghị Tiêu chuẩn xử lý thông tin Liên Bang (FIPS) vào 19/05/1994 và trở thành chuẩn vào 01/12/1994. DSS sử dụng một khoá công khai để kiểm tra tính toàn vẹn của dữ liệu nhận được và đồng nhất với dữ liệu của người gửi. DSS cũng có thể sử dụng bởi người thứ ba để xác định tính xác thực của chữ ký và dữ liệu trong nó. Đầu tiên chúng ta hãy tìm hiểu động cơ của sự thay đổi này, sau đó sẽ tìm hiểu thuật toán của DSS.

Trong rất nhiều trường hợp, một bức điện có thể được mã hoá và giải mã một lần, vì vậy nó đáp ứng cho việc sử dụng của bất kỳ hệ thống bảo mật nào được biết là an toàn lúc bức điện được mã hoá. Nói cách khác, một bức điện được ký đảm nhiệm chức năng như một văn bản hợp pháp, chẳng hạn như các bản hợp đồng, vì vậy nó cũng giống như việc cần thiết để xác minh chữ ký sau rất nhiều năm bức điện được ký. Điều này rất quan trọng cho việc phòng ngừa về độ an toàn của chữ ký được đưa ra bởi một hệ thống bảo mật. Vì hệ chữ ký ElGammal không đảm nhận được điều này, việc thực hiện này cần một giá trị lớn modulo p. Tất nhiên p nên có ít nhất 512-bit, và nhiều người cho rằng độ dài của p nên là 1024-bit nhằm chống lại việc giả mạo trong tương lai.

Tuy nhiên, ngay cả một thuật toán modulo 512-bit dùng để ký cũng phải thực hiện việc tính toán đến 1024-bit. Cho ứng dụng tiềm năng này, có rất nhiều card thông minh được đưa ra, nhằm thực hiện một chữ ký ngắn hơn như mong muốn. DSS đã sửa đổi hệ chữ ký ElGammal cho phù hợp theo cách này một cách khéo léo, để mỗi 160-bit bức điện được ký sử dụng một chữ ký 320-bit, nhưng việc tính toán được thực hiện với 512-bit

modulo p. Cách này được thực hiện nhờ việc chia nhỏ Z_p^* thành các trường có kích thước 2^{160}. Việc thay đổi này sẽ làm thay đổi giá trị δ:

$\delta = (x + \alpha\gamma)k^{-1} \bmod (p - 1)$.

Điều này cũng làm cho giá trị kiểm tra cũng thay đổi:

$\alpha^x\beta^\gamma \equiv \gamma^\delta \pmod p$. (1.4.2.1)

Nếu UCLN(x + αγ, p - 1) = 1 thì sẽ tồn tại $\delta^{-1} \bmod (p - 1)$, do đó (6.1) sẽ biến đổi thành:

$\alpha^{x\delta^{-1}}\beta^{\gamma\delta^{-1}} \equiv \gamma \pmod p$. (1.4.2.2)

Đây chính là sự đổi mới của DSS. Chúng ta cho q là một số nguyên tố 160-bit sao cho q | (p-1), và α là một số thứ q của 1 mod p, thì β và γ cũng là số thứ q của 1 mod p. Do đó α, β và γ có thể được tối giản trong modulo p mà không ảnh hưởng gì đến việc xác minh chữ ký. Sơ đồ thuật toán như sau:

Cho p là một số nguyên tố 512-bit trong trường logarit rời rạc Z_p; q là một số nguyên tố 160-bit và q chia hết (p-1). Cho $\alpha \in Z_p^*$; $P = Z_p^*$, $A = Z_q^*Z_q$, và định nghĩa:

$K = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod p\}$

trong đó giá trị p, q, α và β là công khai, còn a là bí mật.

Với K = (p, α, a, β) và chọn một số ngẫu nhiên k (1 ≤ k ≤ q-1), định nghĩa:

$sig_K(x, k) = (\gamma, \delta)$

trong đó: $\gamma = (\alpha^k \bmod p) \bmod q$

$\delta = (x + a^*\gamma)k^{-1} \bmod q$.

Với $x \in Z_p^*$ *và* $\gamma, \delta \in Z_q$, *việc xác minh được thực hiên bằng cách tính:*

$e_1 = x\delta^{-1} \bmod q$

$e_2 = \gamma\delta^{-1} \bmod q$

$ver(x, \gamma, \delta) = TRUE \Leftrightarrow (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma$. [5]

Chú ý rằng, với DSS thì $\delta \neq 0 \pmod q$ vì giá trị: $\delta^{-1} \bmod q$ cần cho việc xác minh chữ ký (điều này cũng tương tự như việc yêu cầu UCLN(δ, p-1) = 1 để (1.4.2.1) → (1.4.2.2)). Khi B tính một giá trị $\delta \equiv 0 \pmod q$ trong thuật toán ký, anh ta nên bỏ nó đi và chọn một số ngẫu nhiên k mới.

Ví dụ:

Chúng ta chọn q = 101 và p = 78*q + 1 = 7879 và g = 3 là một nguyên tố trong Z_{7879}. Vì vậy , ta có thể tính:

$\alpha = 3^{78} \bmod 7879 = 170$.

Chọn a = 75, do đó: $\beta = \alpha^a \bmod 7879 = 4567$.

Bây giờ, B muốn ký một bức điện x = 1234, anh ta chọn một số ngẫu nhiên k = 50. Vì vậy :

k^{-1} mod 101 = 99.

Tiếp đó: $\gamma = (170^{50} \bmod 7879) \bmod 101 = 2518 \bmod 101 = 94$

$\delta = (1234 + 75*94)99 \bmod 101 = 97$.

Cặp chữ ký (94, 97) cho bức điện 1234 được xác thực như sau:

$\delta^{-1} = 97^{-1} \bmod 101 = 25$

$e_1 = 1234*25 \bmod 101 = 45$

$e_2 = 94*25 \bmod 101 = 27$

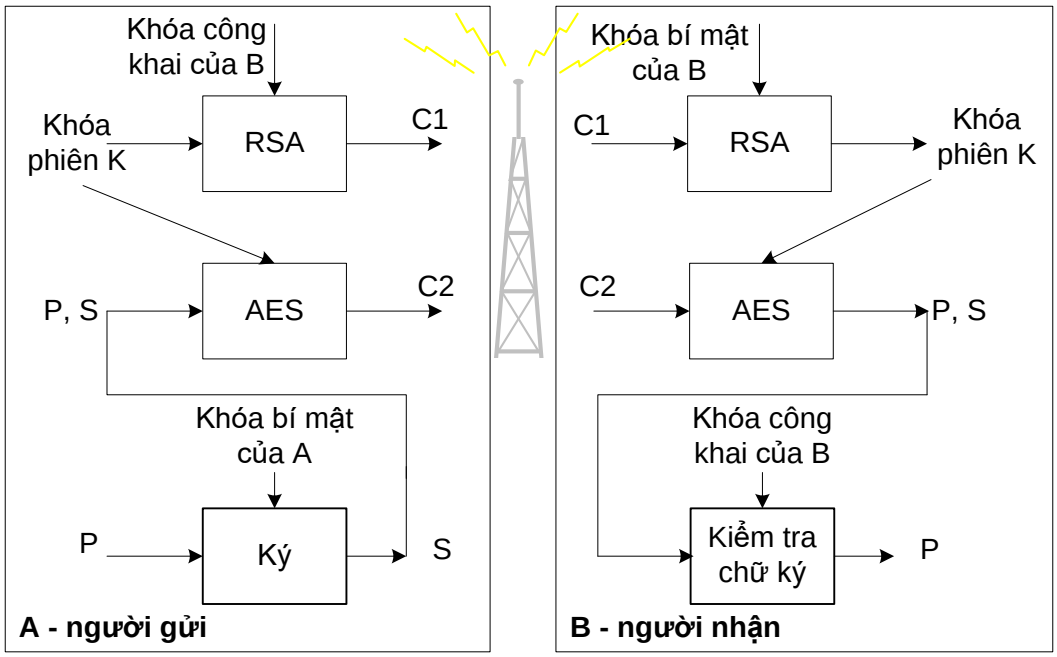
$(170^{45}4567^{27} \bmod 7879) \bmod 101 = 2518 \bmod 101 = 94$.

Kể từ khi DSS được đề xuất vào năm 1991, đã có nhiều phê bình đưa ra. Chẳng hạn như kích cỡ của moduloe p bị cố định 512-bit, điều mà nhiều người không muốn. Vì vậy, NIST đã thay đổi chuẩn này để có thể thay đổi kích thước moduloe (chia bởi 64) thành một dãy từ 512 đến 1024-bit.

Ngoài ra, một sự phê bình khác về DSS là chữ ký được tạo ra nhanh hơn so với việc xác minh nó. Trái ngược với hệ chữ ký RSA thì việc xác minh công khai là rất nhanh chóng (mà ta biết trong thương mại điện tử việc xác minh là rất quan trọng và đòi hỏi thời gian thực hiện phải nhanh chóng).

1.5. Mô hình ứng dụng của chữ ký điện tử

Khác với chữ ký thông thường trên thực tế, các chữ ký điện tử là một thông tin ở dạng số hóa được tạo ra từ văn bản sử dụng hệ chữ ký điện tử và không phải là một phần của văn bản. Do đó sau khi được tạo ra, chữ ký điện tử sẽ được gửi đi cùng với thông điệp, người nhận nhận được thông điệp và chữ ký tương ứng sẽ thực hiện thuật toán kiểm tra xem chữ ký có đúng là chữ ký của người gửi lên văn bản nhận được hay không. Mô hình ứng dụng này có thể được minh họa qua hình vẽ sau:



Hình 5.1: Mô hình ứng dụng của chữ ký điện tử

2. Hàm Băm (Hash Function)

2.1. Khái niệm

Ta thấy rằng các hệ chữ ký được miêu tả ở trên chỉ cho phép ký các bức điện ngắn. Ví dụ như trong DSS, 160-bit bức điện được ký với 320-bit. Như vậy với những bức điện hàng Megabyte thì chúng ta phải làm thế nào!

Một cách đơn giản để giải quyết vấn đề này là chia bức điện lớn thành những đoạn nhỏ 160-bit, và sau đó ký lên mỗi đoạn nhỏ đó, điều này cũng tương tự như mã hoá một chuỗi dài bản rõ bằng việc mã hoá từng ký tự bản rõ sử dụng cùng một khoá.

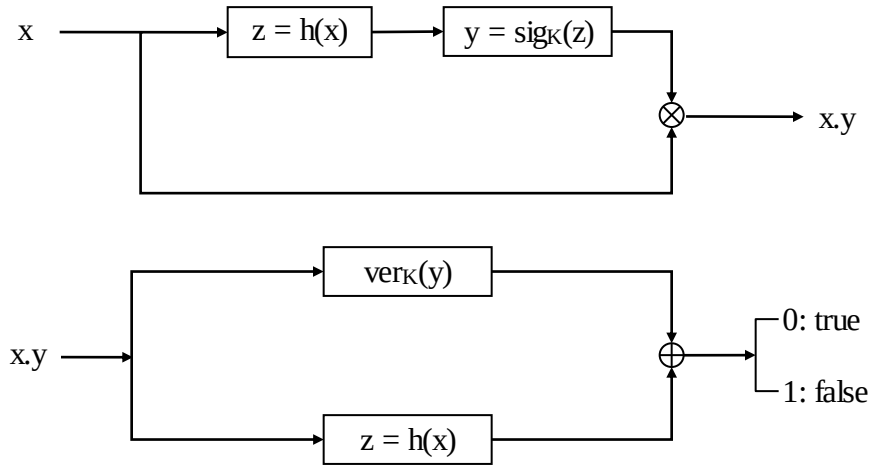
Nhưng có một vài vấn đề trong việc tạo chữ ký điện tử. Đầu tiên là với một bức điện dài, chúng ta sẽ kết thúc với một lượng chữ ký khổng lồ. Ngoài ra, điều bất tiện là hầu hết các hệ chữ ký đều rất chậm. Nghiêm trọng hơn là với rất nhiều đoạn được ký như vậy sẽ dẫn đến khi sắp xếp lại và có thể một vài đoạn bị bỏ đi (mất đi tính toàn vẹn).

Để giải quyết tất cả các rắc rối này, người ta sử dụng hàm Băm (hash function).

Định nghĩa:

Một hàm Băm H sẽ lấy ở đầu vào một thông tin X có kích thước biến thiên và sinh kết quả là một chuỗi có độ dài cố định, được gọi là cốt của bức điện (message digest).

Ví dụ như khi B muốn ký một bức điện x (độ dài bất kỳ), đầu tiên anh ta tính cốt của bức điện z = h(x) (độ dài cố định) và sau đó ký y = $sig_K(z)$. Anh ta phát cặp (x,y) lên kênh truyền, bây giờ việc kiểm tra có thể thực hiện bằng việc tính lại cốt của bức điện z = h(x), sau đó kiểm tra $ver_K(z,y)$ có bằng TRUE hay không.



Hình 5.2: Sơ đồ chữ ký sử dụng hàm Băm

2.2. Đặc tính của hàm Băm

Một vấn đề cần bàn ở đây là tính đụng độ của hàm Băm. Theo nguyên lý Diricle: *nếu có n+1 con thỏ được bỏ vào n cái chuồng thì phải tồn tại ít nhất một cái chuồng mà trong đó có ít nhất là hai con thỏ ở chung* [9]. Rõ ràng với không gian giá trị Băm nhỏ hơn rất nhiều so với không gian tin về mặt kích thước thì chắc chắn sẽ tồn tại đụng độ, nghĩa là có hai tin $x \neq x'$ mà giá trị Băm của chúng là giống nhau, tức h(x) = h(x').

Sau đây chúng ta sẽ xét các dạng tấn công có thể có, từ đó rút ra các tính chất của hàm Băm:

Dạng tấn công thứ nhất là người C bắt đầu với một bức điện được ký có giá trị (x, y), trong đó $y = sig_K(h(x))$ (cặp (x, y) có thể là bất kỳ bức điện trước đó mà B đã ký). Sau đó, C tính z = h(x) và cố gắng tìm $x' \neq x$ để h(x') = h(x). Nếu C làm được điều này thì cặp (x', y) sẽ là một bức điện được ký có giá trị (một bức điện giả mạo có giá trị). Để ngăn cản việc này, hàm Băm h phải thoả mãn tính chất sau:

Tính chất 1:

Một hàm Băm h có tính phi đụng độ cao khi với một bức điện x cho trước , không tìm ra một bức điện $x' \neq x$ sao cho h(x') = h(x). [5]

Một dạng tấn công khác mà người C có thể làm là: đầu tiên anh ta tìm 2 bức điện $x \neq x'$ sao cho h(x) = h(x'). Sau đó C đưa bức điện x cho B và thuyết phục B ký vào cốt bức điện h(x); và vì vậy, anh ta tìm được y. Như vậy, cặp (x', y) là một cặp chữ ký giả có giá trị. Điều này là nguyên nhân mà việc thiết kế hàm Băm phải thoả mãn tính chất 2 như sau:

Tính chất 2:

Một hàm Băm h có tính đụng độ cao khi không thể tìm ra những bức điện x và x' sao cho $x' \neq x$ và h(x') = h(x). [5]

Dạng tấn công thứ 3 là chọn một giá trị cốt z ngẫu nhiên. Người C sẽ tính một chữ ký với một giá trị ngẫu nhiên z, sau đó anh ta tìm một bức điện x sao cho z = h(x). Nếu anh ta làm được điều này thì cặp (x, y) là cặp chữ ký giả có giá trị. Như vậy một tính chất nữa mà h cần thoả mãn là tính một chiều:

Tính chất 3:

Một hàm Băm h có tính một chiều khi với cốt của một bức điện z cho trước không thể tìm được một bức điện x sao cho h(x) = z. [5]

2.3. Birthday attack

Như đã biết, một dạng tấn công có khả năng đối với các hệ chữ ký điện tử có dùng hàm Băm là tìm cách tạo ra những văn bản x và x' có nội dung khác nhau (một có lợi và một là bất lợi cho bên ký) mà giá trị Băm giống nhau. Kẻ địch có thể tìm cách tạo ra một số lượng rất lớn các văn bản có nội dung không thay đổi nhưng khác nhau về biểu diễn nhị phân (đơn giản là việc thêm bớt khoảng trắng hay dùng nhiều từ đồng nghĩa để thay thế ...), sau đó sử dụng một chương trình máy tính để tính giá trị Băm của các văn bản đó và đem so sánh với nhau để hi vọng tìm ra một cặp văn bản đụng độ (sử dụng phương pháp thống kê).

Nhưng việc này đòi hỏi số văn bản cần được tính giá trị Băm phải lớn hơn kích thước không gian Băm rất nhiều. Chẳng hạn như nếu hàm Băm có không gian Băm 64-bit thì số lượng văn bản cần được đem ra nạp vào chương trình phải ít nhất 2^{64} (với một máy tính có thể thực hiện việc Băm 1 triệu bức điện trong 1 giây, thì phải mất 6000.000 năm tính toán [6])

Tuy nhiên nếu kẻ địch thử với lượng văn bản ít hơn nhiều, trong phạm vi có thể tính được thì xác suất để tìm được đụng độ sẽ như thế nào? Câu trả lời là "có thể thực hiện

được". Bản chất của hiện tượng này được minh hoạ rõ thông qua phát biểu sau, thường được gọi là nghịch lý ngày sinh (birthday paradox):

Trong một nhóm có 23 người bất kỳ , xác suất để có hai người có cùng ngày sinh nhật ít nhất là ½. [5]

Một cách tổng quát, giả sử một hàm Băm có n giá trị Băm khác nhau, nếu chúng ta có k giá trị Băm từ k thông tin khác nhau được chọn ngẫu nhiên, thì xác suất để không xảy ra đụng độ là:

(1- $\frac{1}{n}$)(1- $\frac{2}{n}$) ... (1- $\frac{k-1}{n}$) = $\prod_{i=1}^{k-1}(1-\frac{i}{n})$.

Với $\frac{i}{n} << 1$, thì $\prod_{i=1}^{k-1}(1-\frac{i}{n}) \approx \prod_{i=1}^{k-1} e^{\frac{-i}{n}} = e^{\frac{-k(k-1)}{2n}}$. Do đó, xác suất để xảy ra đụng độ ít nhất là $1-e^{\frac{-k(k-1)}{2n}}$. Giả sử gọi xác suất trên là ε ta có:

$1-e^{\frac{-k(k-1)}{2n}} \approx \varepsilon$ (*)

Suy ra : $k^2-k \approx 2n\log\frac{1}{1-\varepsilon}$, suy ra: $k \approx \sqrt{2n\log\frac{1}{1-\varepsilon}}$ (**)

Theo công thức (**) này khi giá trị e rất gần với 1 thì $\log\frac{1}{1-\varepsilon}$ vẫn khá nhỏ nên k là tỉ lệ với $\sqrt{n}$. Với ε = 0.5 ta có k≈1.1774$\sqrt{n}$ (***).

Ví dụ:

Với k = 23 là số người, n = 365 là số ngày trong năm thì xác xuất tồn tại hai người có cùng sinh nhật sẽ là ε = 1 – $2{,}7^{-0{,}7}$ ≈0,5075. Và đây chính là nghịch lý ngày sinh đã phát biểu ở trên. Hoặc chúng ta có thể thay n = 365 vào công thức (***) sẽ nhận được k = 22.49 ≈23.

Nghịch lý ngày sinh hay công thức (*) cho phép chúng ta dự đoán được chặn dưới của số lượng phép thử cần thực hiện để tìm ra đụng độ của một hàm băm. Một hàm băm 40-bit sẽ là không an toàn vì chỉ cần thử 2^{20} (khoảng 1 tỉ) phép thử chúng ta đã có xác suất đụng độ là 50%.

Tương tự, với một hàm Băm có không gian Băm 64-bit nêu trên thì số phép thử để có xác suất đụng độ là 50% sẽ là 2^{32}, điều này là có khả năng thức hiện được. Ví dụ với loại máy tính nêu trên chỉ mất khoảng 1 giờ tính toán.

Hàm băm được coi là an toàn là các hàm băm 128 bit (như MD5 ..) vì khi đó số lượng phép thử sẽ là 2^{64}. Tuy nhiên hiện nay với sự phát triển của các thuật toán thám mã hàm băm mới được phát hiện các hàm băm 128 cũng được khuyến nghị là không nên sử dụng trong các hệ thống bảo mật mới. Các hàm băm được khuyến nghị thay thế cho MD5 là các hàm băm 164 bit như DSS, SHA2.

2.4. Một số hàm Băm nổi tiếng

2.4.1. MD5 (Message Digest)

Ronald Rivest là người đã phát minh ra các hàm Băm MD2, MD4 (1990) và MD5 (1991). Do tính chất tương tự của các hàm Băm này, sau đây chúng ta sẽ xem xét hàm Băm MD5, đây là một cải tiến của MD4 và là hàm Băm được sử dung rộng rãi nhất, nguyên tắc thiết kế của hàm băm này cũng là nguyên tắc chung cho rất nhiều các hàm băm khác.

a. Miêu tả MD5:

Đầu vào là những khối 512-bit, được chia cho 16 khối con 32-bit. Đầu ra của thuật toán là một thiết lập của 4 khối 32-bit để tạo thành một hàm Băm 128-bit duy nhất.

Đầu tiên, ta chia bức điện thành các khối 512-bit, với khối cuối cùng (đặt là x và x < 512-bit) của bức điện, chúng ta cộng thêm một bit 1 vào cuối của x, theo sau đó là các bit 0 để được độ dài cần thiết (512 bit). Kết quả là bức điện vào là một chuỗi M có độ dài chia hết cho 512; vì vậy ta có thể chia M ra thành các N word 32-bit (N word này sẽ chia hết cho 16).

Bây giờ, ta bắt đầu tìm cốt của bức điện với 4 khối 32-bit A, B, C và D (được xem như thanh ghi) :

A = 0x01234567

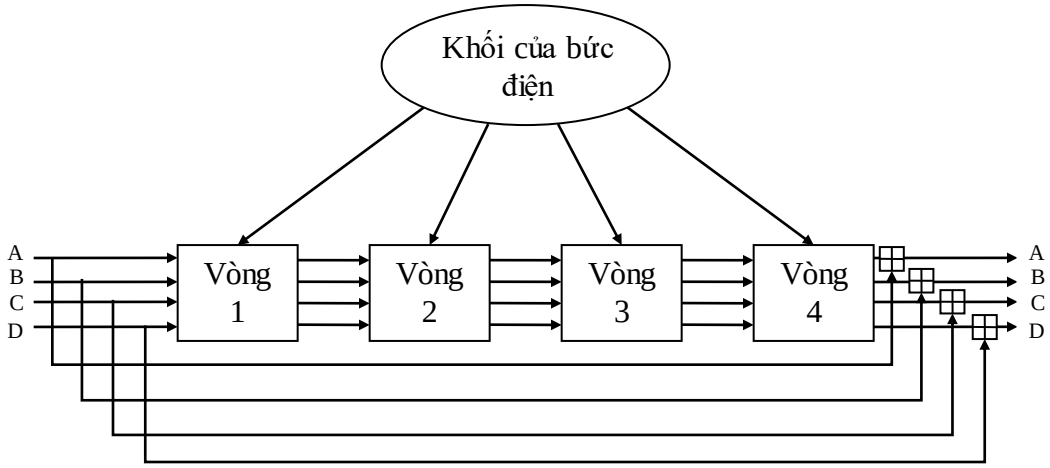
B = 0x89abcdef

C = 0xfedcba98

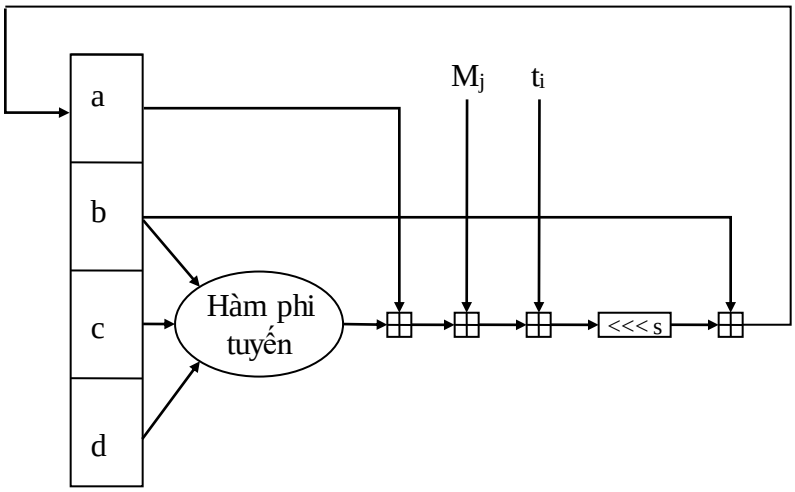
D = 0x76543210.

người ta thường gọi A, B, C, D là các chuỗi biến số (chaining variables).

Bức điện được chia ra thành nhiều khối 512-bit, mỗi khối 512-bit lại được chia ra 16 khối 32-bit đi vào bốn vòng lặp của MD5. Giả sử ta đặt a, b, c và d thay cho A, B, C và D đối với khối 512-bit đầu tiên của bức điện. Bốn vòng lặp trong MD5 đều có cấu trúc giống nhau. Mỗi vòng thực hiện 16 lần biến đổi: thực hiện với một hàm phi tuyến của 3 trong 4 giá trị a, b, c và d; sau đó nó cộng kết quả đến giá trị thứ 4, tiếp đó cộng với một khối con 32-bit và một hằng số. Sau đó, nó dịch trái một lượng bit thay đổi và cộng kết quả vào một trong 4 giá trị a, b, c hay d. Kết quả cuối cùng là một giá trị mới được thay thế một trong 4 giá trị a, b, c hay d.



Hình 5.3: Sơ đồ vòng lặp chính của MD5



Hình 5.4: Sơ đồ một vòng lặp MD5

Có bốn hàm phi tuyến, mỗi hàm này được sử dụng cho mỗi vòng:

F(X,Y,Z) = (X ⊥ Y) □ ((¬X) ⊥ Z)

G(X,Y,Z) = ((X ⊥ Z) □ (Y ⊥ (¬Z)))

H(X,Y,Z) = X ⊕ Y ⊕ Z

I(X,Y,Z) = Y ⊕ (X □ (¬Z)).

trong đó: ⊕ là XOR, ⊥ là AND, □ là OR, và ¬ là NOT.

Những hàm này được thiết kế sao cho các bit tương ứng của X, Y và Z là độc lập và không ưu tiên, và mỗi bit của kết quả cũng độc lập và ngang bằng nhau.

Nếu M_j là một biểu diễn của khối con thứ j (j = 16) và <<<s là phép dịch trái của s bit, thì các vòng lặp có thể biểu diễn như sau:

FF(a,b,c,d,M_j,s,t_i) được biểu diễn a = b + ((a + F(b,c,d) + M_j + t_i) <<< s)

GG(a,b,c,d,M_j,s,t_i) được biểu diễn a = b + ((a + G(b,c,d) + M_j + t_i) <<< s)

HH(a,b,c,d,M_j,s,t_i) được biểu diễn a = b + ((a + H(b,c,d) + M_j + t_i) <<< s)

II(a,b,c,d,M_j,s,t_i) được biểu diễn a = b + ((a + I(b,c,d) + M_j + t_i) <<< s).

Bốn vòng (64 bước) sẽ thực hiện như sau:

Vòng 1:

FF (a, b, c, d, M_0, 7, 0x76aa478)
FF (d, a, b, c, M_1, 12, 0xe8c7b756)
FF (c, d, a, b, M_2, 17, 0x242070db)
FF (b, c, d, a, M_3, 22, 0xc1bdceee)
FF (a, b, c, d, M_4, 7, 0xf57c0faf)
FF (d, a, b, c, M_5, 12, 0x4787c62a)
FF (c, d, a, b, M_6, 17, 0xa8304613)
FF (b, c, d, a, M_7, 22, 0xfd469501)
FF (a, b, c, d, M_8, 7, 0x698098d8)
FF (d, a, b, c, M_9, 12, 0x8b44f7af)
FF (c, d, a, b, M_{10}, 17, 0xffff5bb1)
FF (b, c, d, a, M_{11}, 22, 0x895cd7be)
FF (a, b, c, d, M_{12}, 7, 0x6b901122)
FF (d, a, b, c, M_{13}, 12, 0xfd987193)
FF (c, d, a, b, M_{14}, 17, 0xa679438e)
FF (b, c, d, a, M_{15}, 22, 0x49b40821).

Vòng 2:

GG (a, b, c, d, M_1, 5, 0x61e2562)
GG (d, a, b, c, M_6, 9, 0xc040b340)
GG (c, d, a, b, M_{11}, 14, 0x265e5a51)
GG (b, c, d, a, M_0, 20, 0xe9b6c7aa)
GG (a, b, c, d, M_5, 5, 0xd62f105d)
GG (d, a, b, c, M_{10}, 9, 0x02441453)
GG (c, d, a, b, M_{15}, 14, 0xd8a1e681)
GG (b, c, d, a, M_4, 20, 0xe7d3fbc8)
GG (a, b, c, d, M_9, 5, 0x21e1cde6)
GG (d, a, b, c, M_{14}, 9, 0xc33707d6)
GG (c, d, a, b, M_3, 14, 0xf4d50d87)
GG (b, c, d, a, M_8, 20, 0x455a14ed)
GG (a, b, c, d, M_{13}, 5, 0xa9e3e905)
GG (d, a, b, c, M_2, 9, 0xfcefa3f8)
GG (c, d, a, b, M_7, 14, 0x676f02d9)
GG (b, c, d, a, M_{12}, 20, 0x8d2a4c8a).

Vòng 3:

HH (a, b, c, d, M_5, 4, 0xfffa3942)
HH (d, a, b, c, M_8, 11, 0x8771f681)
HH (c, d, a, b, M_{11}, 16, 0x6d9d6122)
HH (b, c, d, a, M_{14}, 23, 0xfde5380c)
HH (a, b, c, d, M_1, 4, 0xa4beea44)
HH (d, a, b, c, M_4, 11, 0x4bdecfa9)

HH (c, d, a, b, M_7, 16, 0xf6bb4b60)
HH (b, c, d, a, M_{10}, 23, 0xbebfbc70)
HH (a, b, c, d, M_{13}, 4, 0x289b7ec6)
HH (d, a, b, c, M_0, 11, 0xeaa127fa)
HH (c, d, a, b, M_3, 16, 0xd4ef3085)
HH (b, c, d, a, M_6, 23, 0x04881d05)
HH (a, b, c, d, M_9, 4, 0xd9d4d039)
HH (d, a, b, c, M_{12}, 11, 0xe6db99e5)
HH (c, d, a, b, M_{15}, 16, 0x1fa27cf8)
HH (b, c, d, a, M_2, 23, 0xc4ac5665).

Vòng 4:

II (a, b, c, d, M_0, 6, 0xf4292244)
II (d, a, b, c, M_7, 10, 0x432aff97)
II (c, d, a, b, M_{14}, 15, 0xab9423a7)
II (b, c, d, a, M_5, 21, 0xfc93a039)
II (a, b, c, d, M_{12}, 6, 0x655b59c3)
II (d, a, b, c, M_3, 10, 0x8f0ccc92)
II (c, d, a, b, M_{10}, 15, 0xffeff47d)
II (b, c, d, a, M_1, 21, 0x85845dd1)
II (a, b, c, d, M_8, 6, 0x6fa87e4f)
II (d, a, b, c, M_{15}, 10, 0xfe2ce6e0)
II (c, d, a, b, M_6, 15, 0xa3013414)
II (b, c, d, a, M_{13}, 21, 0x4e0811a1)
II (a, b, c, d, M_4, 6, 0xf7537e82)
II (d, a, b, c, M_{11}, 10, 0xbd3af235)
II (c, d, a, b, M_2, 15, 0x2ad7d2bb)
II (b, c, d, a, M_9, 21, 0xeb86d391).

Những hằng số t_i được chọn theo quy luật sau: ở bước thứ i giá trị t_i là phần nguyên của 2^{32}*abs(sin(i)), trong đó i = [0..63] được tính theo radian.

Sau tất cả những bước này a, b, c và d lần lượt được cộng với A, B, C và D để cho kết quả đầu ra; và thuật toán tiếp tục với khối dữ liệu 512-bit tiếp theo cho đến hết bức điện. Đầu ra cuối cùng là một khối 128-bit của A, B, C và D, đây chính là hàm Băm nhận được.

b. Tính bảo mật trong MD5:

Ron Rivest đã phác hoạ những cải tiến của MD5 so với MD4 như sau:

– Vòng thứ 4 được thêm vào (còn MD4 chỉ có 3 vòng).

– Mỗi bước được cộng thêm một hằng số duy nhất.

– Hàm G ở vòng 2 thay đổi từ ((X ⊥ Y) □ (X ⊥ Z) □ (Y ⊥ Z)) thành ((X ⊥ Z) □ (Y ⊥ (¬Z))) nhằm giảm tính đối xứng của G (giảm tính tuyến tính).

– Mỗi bước được cộng kết quả của bước trước nó, làm các quá trình có tính liên kết, phụ thuộc lẫn nhau.

– Việc các khối con bị thay đổi khi vào vòng 2 và vòng 3 làm cho khuôn dạng cấu trúc vòng lặp thay đổi theo.

– Số lượng lượng bit dịch trái của mỗi vòng được tối ưu và các bước dịch ở mỗi vòng là khác nhau.

Năm 1993, den Boer và Bosselaers đã tìm ra đụng độ trong việc sử dụng hàm nén (vòng 2 và 3) của MD5. Điều này phá vỡ quy luật thiết kế MD5 là chống lại sự đụng độ, nhưng MD5 vẫn là hàm Băm được sử dụng rộng rãi hiện nay.

2.4.2. SHA (Secure Hash Algorithm)

Năm 1995, tổ chức NIST cùng NSA đã thiết kế ra thuật toán hàm Băm an toàn (SHA) sử dụng cho chuẩn chữ ký điện tử DSS. SHA được thiết kế dựa trên những nguyên tắc của MD4/MD5, tạo ra 160-bit giá trị Băm.

a. Miêu tả SHA:

Cũng giống với MD5, bức điện được cộng thêm một bit 1và các bit 0 ở cuối bức điện để bức điện có thể chia hết cho 512. SHA sử dụng 5 thanh ghi dịch:

A = 0x67452301

B = 0xefcdab89

C = 0x98badcfe

D = 0x10325476

E = 0xc3d2e1f0

Bức điện được chia ra thành nhiều khối 512-bit. Ta cũng đặt là a, b, c, d và e thay cho A, B, C, D và E đối với khối 512-bit đầu tiên của bức điện. SHA có bốn vòng lặp chính với mỗi vòng thực hiện 20 lần biến đổi: bao gồm thực hiện với một hàm phi tuyến của 3 trong 5 giá trị a, b, c, d và e; sau đó cũng được cộng và dịch như trong MD5.

SHA xác lập bốn hàm phi tuyến như sau:

$f_t(X,Y,Z) = (X \perp Y) \square ((\neg X) \perp Z)$ với $0 \le t \le 19$

$f_t(X,Y,Z) = X \oplus Y \oplus Z$ với $20 \le t \le 39$

$f_t(X,Y,Z) = (X \perp Y) \square (X \perp Z) \square (Y \perp Z)$ với $40 \le t \le 59$

$f_t(X,Y,Z) = X \oplus Y \oplus Z$ với $60 \le t \le 79$.

Bốn hằng số sử dụng trong thuật toán là:

$K_t = 2^{1/2} /4 =$ 0x5a827999 với $0 \le t \le 19$

$K_t = 3^{1/2} /4 =$ 0x6ed9eba1 với $20 \le t \le 39$

$K_t = 5^{1/2} /4 =$ 0x8f1bbcdc với $40 \le t \le 59$

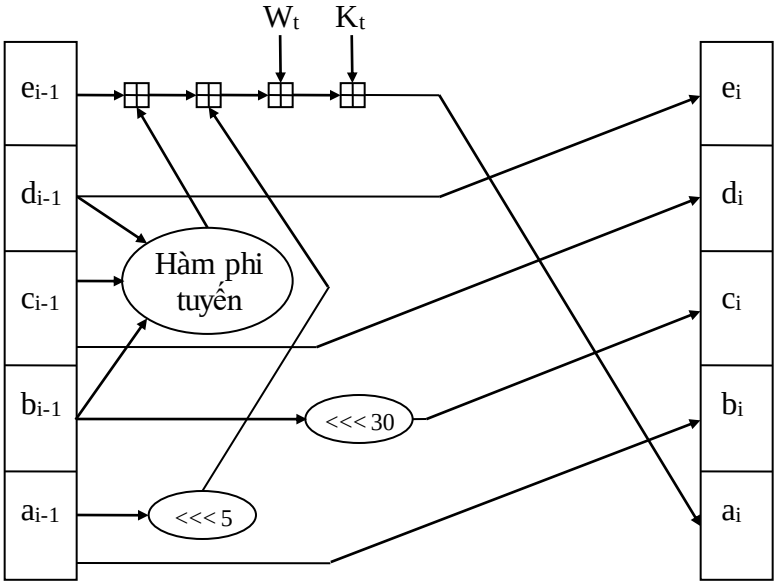
$K_t = 10^{1/2} /4 =$ 0xca62c1d6 với $60 \le t \le 79$.

Các khối bức điện được mở rộng từ 16 word 32-bit (M_0 đến M_{15}) thành 80 word 32-bit (W_0 đến W_{79}) bằng việc sử dụng thuật toán mở rộng:

$W_t = M_t$ với $0 \le t \le 15$

$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$ *với* $16 \le t \le 79$.

Ta có thể miêu tả một vòng lặp của SHA như sau:



Hình 5.5: Sơ đồ một vòng lặp của SHA

Nếu gọi W_t là biểu diễn của khối con thứ t của bức điện được mở rộng, và <<<s là biểu diễn dịch trái s bit, thì vòng lặp chính của SHA như sau:

a = A, b = B, c = C, D = D, e = E,

for t = 0 to 79

{

TEMP $= (a <<< 5) + f_t(b, c, d) + e + W_t + K_t$,

e = d,

d = c,

c = b <<< 30,

b = a,

a = TEMP,

}

A = A + a, B = B + b, C = C + c, D = D + d, E = E + e,

Thuật toán tiếp tục với khối 512-bit tiếp theo cho tới khi hết bức điện, và kết quả sau cùng trong 4 thanh ghi A, B, C, D và E chính là hàm Băm SHA 160-bit.

b. Tính bảo mật trong SHA:

Để hiểu rõ hơn về tính bảo mật của SHA, ta hãy so sánh SHA với MD5 để có thể tìm ra những điểm khác nhau của hai hàm Băm này:

- MD5 và SHA đều cộng thêm các bit "giả" để tạo thành những khối chia hết cho 512-bit, nhưng SHA sử dụng cùng một hàm phi tuyến f cho cả bốn vòng.

- MD5 sử dụng mỗi hằng số duy nhất cho mỗi bước biến đổi, SHA sử dụng mỗi hằng số cho mỗi vòng biến đổi, hằng số dịch này là một số nguyên tố đối với độ lớn của word (giống với MD4).
- Trong hàm phi tuyến thứ 2 của MD5 có sự cải tiến so với MD4, SHA thì sử dụng lại hàm phi tuyến của MD4, tức (X ⊥ Y) □ (X ⊥ Z) □ (Y ⊥ Z).
- Trong MD5 với mỗi bước được cộng kết quả của bước trước đó. Sự khác biệt đối với SHA là cột thứ 5 được cộng (không phải b, c hay d như trong MD5), điều này làm cho phương pháp tấn công của Boer-Bosselaers đối với SHA bị thất bại (den Boer và Bosselaers là hai người đã phá thành công 2 vòng cuối trong MD4).

Cho đến nay, chưa có một công bố nào được đưa ra trong việc tấn công SHA, bởi vì độ dài của hàm Băm SHA là 160-bit, nó có thể chống lại phương pháp tấn công bằng vét cạn (kể cả birthday attack) tốt hơn so với hàm Băm MD5 128-bit.

2.5. Một số ứng dụng của hàm Băm

Như đã trình bày ở phần đầu chương, ứng dụng chính của các hàm băm là sử dụng với các hệ chữ ký điện tử, trong đó thay vì ký trực tiếp lên các văn bản, thông điệp (mà trong đa số trường hợp là rất lớn, tốc độ chậm) người ta sẽ ký lên giá trị băm đại diện cho toàn bộ văn bản đó. Điều này đặc biệt quan trọng và hiệu quả bởi vì chúng ta biết rằng các hệ chữ ký điện tử đều làm việc với các phép tính số học số lớn nên bản thân chúng đã tương đối chậm, việc sử dụng giá trị băm thay cho toàn bộ văn bản là giải pháp toàn diện khắc phục được yếu điểm này của các hệ chữ ký điện tử. Ngoài việc xử dụng với các hệ chữ ký điện tử hàm băm còn được sử dụng vào các mục đích khác như: xác thực hóa thông điệp, xác thực hóa người dùng.

Đối với các ứng dụng không cần giữ bí mật thông điệp mà chỉ cần đảm bảo thông điệp không bị thay đổi trên đường truyền người ta sẽ sử dụng hàm băm cho mục đích xác thực tính nguyên vẹn của thông điệp đó. Chẳng hạn chúng ta có một phần mềm mã nguồn mở ở dạng setup muốn phân phối cho người dùng, rõ ràng việc gửi phần mềm đó tới máy tính của người dùng là không cần phải mã hóa, tuy nhiên nếu như phần mềm đó bị thay đổi trên đường truyền (chẳng hạn như bị gắn thêm các spyware, virus …) thì sẽ rất nguy hiểm. Để đảm bảo chúng ta sẽ cung cấp giá trị băm của phần mềm đó (khi đó phần mềm chính là thông điệp). Người dùng sẽ download cả phần mềm và giá trị băm nhận được, sau đó tiến hành băm lại, đối sánh giá trị băm nhận được với giá trị băm được cung cấp cùng với phần mềm, nếu hai giá trị này khớp nhau thì có thể đảm bảo phần mềm không bị sửa đổi trên đường truyền. Hiện nay đa số các phần mềm mã nguồn mở đều được phân phối theo cách này.

Trong các hệ thống yêu cầu có xác thực người dùng như các hệ quản trị cơ sở dữ liệu, hệ điều hành, các ứng dụng web, ứng dụng dạng desktop application, để lưu mật khẩu người dùng người ta cũng sử dụng các hàm băm hoặc các hệ mã trong các vai trò của hàm băm (không sử dụng khóa). Khi đó mỗi tài khoản của người dùng thay vì lưu dưới dạng tên truy cập (username) và mật khẩu (password) sẽ được lưu dưới dạng: tên người dùng, giá trị băm của mật khẩu. Khi một người dùng đăng nhập vào hệ thống, hệ thống sẽ lấy tên truy cập, mật khẩu họ nhập vào, kiểm tra xem có tên truy cập nào như vậy hay không. Nếu có sẽ tiến hành băm giá trị mật khẩu do người dùng nhập vào, đối

sánh với giá trị băm tương ứng lưu trong cơ sở dữ liệu (có thể ở dạng file text, xml, hay file cơ sở dữ liệu của một hệ quản trị cơ sở dữ liệu nào đó). Nếu kết quả đối sánh là khớp thì người dùng đó là hợp lệ, ngược lại nếu không khớp có nghĩa là sai mật khẩu. Hiện nay tất cả các hệ quản trị cơ sở dữ liệu đều được trang bị các hàm băm để cho phép người dùng tạo ra các giá trị băm của mật khẩu người dùng và lưu lại các giá trị băm này. Việc lưu các giá trị băm đảm bảo chúng ta không bị lộ mật khẩu do mật khẩu được lưu ở dạng nguyên bản trên máy tính hoặc khi truyền qua hệ thống mạng. Hệ điều hành Unix sử dụng nguyên tắc lưu mật khẩu như trên với hàm băm là hệ mã DES được lặp lại 25 lần, mật khẩu của người dùng được sử dụng như khóa của hệ mã, bản rõ đem mã hóa là xâu 64 bit 0.

Ngày nay với sự phát triển mạnh mẽ của thương mại điện tử, các giao dịch đều được thực hiện từ xa, trên các hệ thống mạng nên việc ứng dụng của các hệ chữ ký điện tử và đi kèm với đó là các hàm băm ngày càng trở nên quan trọng. Mọi thông tin trong các giao dịch thương mại điện tử đều cần được bảo vệ bằng các chữ ký, hàm băm. Vì thế có thể nói rằng đôi khi các hàm băm còn quan trọng hơn cả các hệ mã mật.

3. Bài tập

Bài tập 5.1: Cho hệ chữ ký điện tử ElGamma có p = 1019, a = 191 là một phần tử nguyên thuỷ của Z_P^*, x = 37.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ chữ ký trên.
b) Để ký lên bản rõ M = 102 người ta chọn k = 143, hãy thực hiện ký đưa ra chữ ký tương ứng.
c) Kiểm tra xem cặp (K, S) = (251, 507) có là chữ ký lên văn bản M = 127 hay không.

Bài tập 5.2: Cho hệ chữ ký điện tử RSA có p = 31, q = 41, e = 271.

a) Hãy tìm khóa công khai K_P, và khóa bí mật K_S của hệ mã trên.
b) Hãy tính chữ ký cho thông điệp M = 100.

Bài tập 5.3: Cho thuật toán chữ ký điện tử DSA có q = 11, p = 67, α = 9, β = 62, khóa bí mật a = 4, để ký lên văn bản M = 8, người ta chọn k = 2. Hãy xác định chữ ký lên văn bản M.

Bài tập 5.4: Cho hệ chữ ký điện tử RSA có p = 47, q = 71, e= 79. Hãy xác định chữ ký của hệ mã lên thông điệp M = 688.

Sử dụng một trong các ngôn ngữ lập trình C, C++, Java hoặc C# để làm các bài tập sau:

Bài tập 5.5: Cài đặt hệ chữ ký điện tử RSA.

Bài tập 5.6: Cài đặt hệ chữ ký điện tử El Gammal.

Bài tập 5.7: Cài đặt hàm băm MD5.

Bài tập 5.8: Cài đặt hàm băm SHA.

Gợi ý: Có thể sử dụng các thư viện số lớn như MIRACL hoặc các thư viện mã nguồn mở như Crypto++ (chi tiết tại địa chỉ website: http://www.cryptopp.com/), Cryptolib (chi tiết tại địa chỉ website http://www.cs.auckland.ac.nz/~pgut001/cryptlib).

CHƯƠNG VI: QUẢN LÝ KHÓA

1. Quản lý khoá trong các mạng truyền tin

Trong các chương trước, ta đã làm quen với các phương pháp lập mã và các bài toán quan trọng khác liên quan đến việc truyền tin bảo mật trên các mạng truyền tin công cộng nói chung. Ta cũng đã thấy rằng các hệ mật mã khoá công khai công khai có nhiều ưu việt hơn các hệ mật mã đối xứng trong việc làm nền tảng cho các giải pháp an toàn thông tin, và đặc biệt đối với các hệ mã khoá đối xứng thì việc thực hiện đòi hỏi những kênh bí mật để chuyển khoá hoặc trao đổi khoá giữa các đối tác, thì về nguyên tắc, đối với các hệ mã hoá với khoá công khai không cần có những kênh bí mật như vậy, vì các khoá công khai có thể được truyền hay trao đổi cho nhau một cách công khai qua các kênh truyền tin công cộng. Tuy nhiên, trên thực tế, để bảo đảm cho các hoạt động thông tin được thật sự an toàn, không phải bất cứ thông tin nào về các khoá công khai của một hệ mã, của một thuật toán kiểm tra chữ ký, của một giao thức xác nhận thông báo hay xác nhận danh tính … cũng phát công khai một cách tràn lan trên mạng công cộng, mặc dù là công khai nhưng người ta cũng muốn là những ai cần biết thì mới nên biết mà thôi. Do đó, mặc dù sử dụng các hệ có khoá công khai, người ta cũng muốn có những giao thức thực hiện việc trao đổi khoá giữa các đối tác thực sự có nhu cầu giao lưu thông tin với nhau, kể cả trao đổi khoá công khai. Việc trao đổi khoá giữa các chủ thể trong một cộng đồng nào đó có thể được thiết lập một cách tự do giữa bất cứ hai người nào khi có nhu cầu trao đổi thông tin, hoặc có thể được thiết lập một cách tương đối lâu dài trong thời gian nào đó trong cả cộng đồng với sự điều phối của một cơ quan được uỷ thác TA. Việc trao đổi khoá trong trường hợp thứ nhất ta gọi đơn giản là thoả thuận khoá, còn trong trường hợp thứ hai ta gọi là phân phối khoá; TA là nơi thực hiện việc phân phối, cũng là nơi quản lý khoá. Việc thoả thuận khoá nói chung không cần có sự tham gia của một TA nào và chỉ có thể xảy ra khi các hệ bảo mật mà ta sử dụng là hệ có khoá công khai, còn việc phân phối khoá thì có thể xảy ra đối với các trường hợp sử dụng các hệ khoá đối xứng cũng như các hệ có khoá công khai. Việc phân phối khoá với vai trò quản trị khoá của một TA là một việc bình thường, đã tồn tại rất lâu trước khi có các hệ mật mã khoá công khai. Ta sẽ bắt đầu với một vài hệ phân phối khoá như vậy, sau đó sẽ giới thiệu một số hệ phân phối hoặc trao đổi khoá khi dùng các sơ đồ an toàn và bảo mật với khoá công khai.

2. Một số hệ phân phối khoá

2.1. Sơ đồ phân phối khoá Blom

Giả sử ta có một mạng gồm có n người dùng và mỗi người dùng đó đều có nhu cầu trao đổi thông tin bí mật với mọi người trong mạng. Giả sử sơ đồ mật mã được sử dụng là một sơ đồ mật mã khoá đối xứng (chẳng hạn như DES). Toàn bộ mạng cần có $\frac{n(n-1)}{2}$ khoá khác nhau cho chừng ấy cặp người dùng khác nhau trong mạng. Một cơ quan uỷ thác TA quản lý chừng ấy khoá và phải chuyển cho mỗi người dùng (n-1) khoá chung với (n-1) người còn lại trong mạng; như vậy TA phải truyền bằng những kênh bí mật tất cả là n(n-1) lượt khoá đến tất cả n người dùng.

Năm 1985, Blom đề nghị một sơ đồ phân phối khoá, mà sau đây ta gọi là sơ đồ Blom, trong trường hợp đơn giản nhất được mô tả như sau:

- TA chọn một số nguyên tố p ≥ n, và chọn cho mỗi người dùng A một số $r_A \in Z_p$. Số p và các số r_A được công bố công khai.
- Sau đó, TA chọn ba số ngẫu nhiên a, b, c $\in Z_p$ và lập đa thức:

  $$f(x, y) = a + b(x + y) + cxy \bmod p$$

- Với mỗi người dùng A, TA tính $g_A(x) = f(x, r_A) = a_A + b_A x \bmod p$, trong đó $a_A = a + br_A \bmod p$, $b_A = b + cr_A \bmod p$. TA chuyển bí mật cặp số (a_A, b_A) cho A. Như vậy, A biết $g_A(x) = a_A + b_A x$.

So với việc TA phải truyền bí mật n(n-1) lượt khoá trên thì với sơ đồ Blom, TA chỉ phải truyền n lượt các cặp số (a_A, b_A) mà thôi.

Sau khi đã thực hiện xong các công việc chuẩn bị đó, bây giờ nếu hai người dùng A và B muốn tạo khoá chung để truyền tin bằng mật mã cho nhau thì khoá chung $K_{A,B}$ đó sẽ là:

$$K_{A,B} = g_A(r_B) = g_B(r_A) = f(r_A, r_B),$$

mà mỗi người A và B tính được bằng những thông tin mình đã có.

Như vậy, theo sơ đồ phân phối này, TA phân phối cho mọi người dùng một phần bí mật của khoá, hai người dùng bất kỳ phối hợp phần bí mật của riêng mình với phần công khai của người kia để cùng tạo nên khoá bí mật chung cho hai người. Sơ đồ này là an toàn theo nghĩa sau đây: bất kỳ một người thức ba C nào (kể cả C là một người tham gia trong mạng) có thể được phát hiện được khoá bí mật riêng của hai người A và B. Thực vậy, dù C có là người tham gia trong mạng đi nữa, thì cái mà C biết nhiều lắm là hai số a_C, b_C do TA cấp cho. Ta chứng minh rằng với những gì mà C biết thì bất kỳ giá trị $\ell \in Z_p$ nào cũng có thể được chấp nhận là $K_{A,B}$. Những gì mà C biết , kể cả chấp nhận $\ell = K_{A,B}$, được thể hiện thành:

$$\begin{array}{ccccccc} a & + & b(r_A + r_B) & + & cr_A r_B & = & \ell \\ a & + & br_C & & & = & a_C \\ & & b & + & cr_C & = & b_C \end{array}$$

Nếu xem a, b, c là ẩn số, ta có định thức các hệ số ở vế phải là:

$$\begin{vmatrix} 1 & r_A + r_B & r_A r_B \\ 1 & r_C & 0 \\ 0 & 1 & r_C \end{vmatrix} = (r_C - r_A)(r_C - r_B),$$

Theo giả thiết chọn các số r, định thức đó khác 0, do đó hệ phương trình luôn có nghiệm (a, b, c), tức việc chấp nhận ℓ là giá trị của $K_{A,B}$ là hoàn toàn có thể. Bất kỳ giá trị

$\ell \in Z_p$ nào cũng có thể được C chấp nhận là $K_{A,B}$, điều đó đồng nghĩa với việc C không biết $K_{A,B}$ là số nào.

Tuy nhiên, nếu có hai người tham gia C và D (khác A, B) liên minh với nhau để phát hiện $K_{A,B}$ thì lại rất dễ dàng, vì cả C và D biết:

$$\begin{aligned} a + br_C &= a_C \\ b + cr_C &= b_C \\ a + br_D &= a_D \\ b + cr_D &= b_D \end{aligned}$$

bốn phương trình đó đủ để xác định (a, b, c) từ đó tìm được $K_{A,B}$.

Ta có thể mở rộng sơ đồ Blom nói trên để được một sơ đồ Blom tổng quát, trong đó mọi khoá chung $K_{A,B}$ của hai người dùng A và B là bí mật hoàn toàn đối với bất kỳ liên minh nào gồm k người ngoài A và B, nhưng không còn là bí mật đối với mọi liên minh gồm k+1 người tham gia trong mạng. Muốn vậy, ta chỉ cần thay đa thức f(x, y) nói trên bằng một đa thức đối xứng bậc 2k sau đây:

$$f(x,y) = \sum_{i=0}^{k}\sum_{j=0}^{k} a_{ij}x^i y^j \bmod p,$$

trong đó $a_{ij} \in Z_p, 0 \le i, j \le k, a_{ij} = a_{ji}$ với mọi i, j.

2.2. Hệ phân phối khoá Kerberos

Kerberos là tên của một hệ dịch vụ phân phối (hay cấp phát) khoá phiên (sesion key) cho từng phiên truyền tin bảo mật theo yêu cầu của người dùng trong một mạng truyền tin. Hệ mật mã được sử dụng thường là hệ có khoá đối xứng chẳng hạn như DES.

Để thực hiện hệ này, trước hết cơ quan được uỷ thác (hay trung tâm điều phối) TA cần chia sẻ một khoá DES bí mật K_A với mỗi thành viên A trong mạng. Sau đó, mỗi lần A có nhu cầu truyền tin bảo mật với một thành viên khác B thì yêu cầu TA cấp một khoá phiên cho cả A và B. Việc cấp phát đó sẽ được thực hiện bằng một giao thức phân phối khoá như sau:

1) TA chọn ngẫu nhiên một khoá phiên K, xác định một tem thời gian T và thời gian sống L (như thế có nghĩa là khoá phiên K có giá trị sử dụng trong khoảng thời gian từ T đến T+L).

2) TA tính $m_1 = e_{K_A}(K, ID(B), T, L)$, $m_2 = e_{K_B}(K, ID(A), T, L)$ và gửi (m_1, m_2) đến A.

3) A dùng hàm giải mã d_{K_A} cho m_1 để thu được K, T, L, ID(B). Sau đó tính $m_3 = e_K(ID(A), T)$, và gửi (m_3, m_2) cho B.

4) B dùng các hàm giải mã d_{K_B} cho m_2 và d_K cho m_3 để thu được K, T, L, ID(A) và ID(A), T. Nếu thấy hai giá trị của ID(A) và của T trùng nhau thì B tính tiếp $m_4 = e_K(T + 1)$ và gửi m_4 cho A.

5) A dùng hàm giải mã d_K cho m_4 và thử xem kết quả thu được có đúng là T+1 hay không.

Trong giao thức nói trên, các ký hiệu ID(A) và ID(B) là chỉ danh tính của A và của B, các thông tin đó là công khai.

Hoàn thành giao thức gồm 5 bước nói trên, TA (cùng với A và B) đã thực hiện xong việc cấp phát một khoá phiên K cho hai người dùng A và B để truyền tin mật mã cho nhau. Tất cả các việc trao đổi các thông tin trong giao thức đó đều được thực hiện trên các kênh công cộng, dù khoá K vẫn là bí mật (chỉ A, B và TA là được biết mà thôi). Ngoài việc cấp phát khoá, giao thức đó còn thực hiện được việc xác nhận khoá: B và A đều tin chắc được rằng đối tác của mình đã thực sự có khoá K do kết quả của việc thực hiện các phép thử ở bước 4 và 5. Thêm nữa, cả A và B còn biết được thời hạn có hiệu lực của khoá.

Phân phối khoá bí mật theo giao thức Kerberos có độ tin cậy cao, tuy nhiên trong thực tế, việc sử dụng nó cũng đòi hỏi tốn nhiều thời gian nên ngày nay cũng chỉ được dùng trong những trường hợp hạn chế.

2.3. Hệ phân phối khóa Diffe-Hellman

Hệ phân phối khoá Diffe-Hellman không đòi hỏi TA phải biết và chuyển bất kỳ thông tin mật nào về khoá của các người tham gia trong mạng để họ thiết lập được khoá chung bí mật cho việc truyền tin với nhau.

Trong một hệ phân phối khoá Diffe-Hellman, TA chỉ việc chọn một số nguyên tố lớn p và một phần tử nguyên thuỷ α theo mod p sao cho bài toán tính $\log_a$ trong Z_p^* là rất khó. Các số p và α được công bố công khai cho mọi người tham gia trong mạng. Ngoài ra, TA có một sơ đồ chữ ký với thuật toán ký bí mật sig_{TA} và thuật toán kiểm tra công khai ver_{TA}.

Một thành viên bất kỳ A với danh tính ID(A) tuỳ ý chọn một số a_A ($0 \le a_A \le p-2$) và tính $b_A = \alpha^{a_A} \bmod p$. A giữ bí mật a_A và đăng ký các thông tin (ID(A), b_A) với TA. TA cấp cho A chứng chỉ:

$C(A) = (ID(A), b_A, sig_{TA}(ID(A), b_A))$.

Các chứng chỉ của các thành viên trong mạng có thể được lưu giữ trong một cơ sở dữ liệu công khai hoặc uỷ thác cho TA lưu giữ và cung cấp công khai cho các thành viên mỗi khi cần đến.

Khi hai thành viên A và B trong mạng cần có một khoá bí mật chung để truyền tin bảo mật cho nhau thì A dùng thông tin công khai b_B có trong C(B) kết hợp với số bí mật của mình là a_A để tạo nên khoá.

$$K_{A,B} = b_B^{a_A} \bmod p = \alpha^{a_B a_A} \bmod p.$$

Khoá chung đó B cũng tạo ra được từ các thông tin công khai b_A của A và số bí mật a_B của mình:

$$K_{A,B} = b_B^{a_B} \bmod p = \alpha^{a_A a_B} \bmod p.$$

Để bảo đảm được các thông tin về b_B và b_A là chính xác, A và B có thể dùng thuật toán ver_{TA} để kiểm tra chữ ký xác nhận của TA trong các chứng chỉ C(B) và C(A) tương ứng.

Cơ sở lý thuyết đảm bảo cho sự an toàn của các phương pháp trao đổi khóa dựa trên hệ phân phối khóa Diffie-Hellman là bài toán Logarithm rời rạc, có thể tham khảo thêm trong phần 3.3 chương IV để biết thêm.

3. Trao đổi khoá và thoả thuận khoá

3.1. Giao thức trao đổi khoá Diffie-Hellman

Hệ phân phối khoá Diffie-Hellman nói trong mục trước có thể dễ dàng biến đổi thành một giao thức trao đổi (hay thoả thuận) khoá trực tiếp giữa các người sử dụng mà không cần có sự can thiệp của một TA làm nhiêm vụ điều hành hoặc phân phối khoá. Một nhóm bất kỳ người sử dụng có thể thoả thuận cùng dùng chung một số nguyên tố lớn p và một phần tử nguyên thuỷ α theo mod p, hai người bất kỳ trong nhóm A và B mỗi khi muốn truyền tin bảo mật cho nhau có thể cùng thực hiện giao thức sau đây để trao đổi khoá:

1) A chọn ngẫu nhiên số a_A ($0 \le a_A \le p-2$) bí mật, tính $b_A = \alpha^{a_A} \bmod p$ và gửi b_A cho B .

2) Tương tự, B chọn ngẫu nhiên số a_B ($0 \le a_B \le p-2$) bí mật, tính $b_B = \alpha^{a_B} \bmod p$ và gửi b_B cho A.

3) A và B cùng tính được khoá chung:

$$K_{A,B} = b_B^{a_A} \bmod p = b_A^{a_B} \bmod p \ (= \alpha^{a_A a_B} \bmod p).$$

Giao thức trao đổi khoá Diffie-Hellman có các tính chất sau:

- Giao thức là an toàn đối với việc tấn công thụ động, nghĩa là một người thứ ba dù biết b_A và b_B sẽ khó mà biết được $K_{A,B}$.

Chúng ta biết rằng bài toán "biết b_A và b_B tìm $K_{A,B}$" chính là bài toán Diffie-Hellman, bài toán này tương đương với bài toán phá mã ElGammal. Bây giờ ta sẽ chứng minh điều này.

Phép mật mã ElGammal với khoá K = (p, α, a, β), trong đó $\beta = \alpha^a \bmod p$ cho ta từ một bản rõ x và một số ngẫu nhiên $k \in Z_{p-1}$ lập được mật mã $e_K(x, k) = (y_1, y_2)$ với $y_1 = \alpha^k \bmod p$, $y_2 = x\beta^k \bmod p$. Và phép giải mã được cho bởi $y_1 = \alpha^k \bmod p$.

Giả sử ta có thuật toán A giải bài toán Diffie-Hellman. Ta sẽ dùng A để phá mã ElGammal như sau:

Cho mật mã (y_1, y_2). Trước tiên, dung A cho $y_1 = \alpha^k \bmod p$ và $\beta = \alpha^a \bmod p$, ta được $A(y_1, B) = \alpha^{ka} = \beta^k \bmod p$. Sau đó, ta thu được bản rõ x từ β^k và y_2 như sau:

$$x = y_2 (\beta^k)^{-1} \bmod p.$$

Ngược lại, giả sử có một thuật toán khác là B dùng để phá mã ElGammal, tức $B(p,\alpha,\beta,y_1,y_2) = x = y_2(y_1^a)^{-1} \bmod p$. Áp dụng B cho $\beta = b_A$, $y_1 = b_B$, $y_2 = 1$, ta được $B(p,\alpha,b_A,b_B,1)^{-1} = (1.(b_B^{a_A})^{-1})^{-1} = \alpha^{a_A a_B} \bmod p$, tức giải được bài toán Diffie-Hellman.

- Giao thức là không an toàn đối với việc tấn công chủ động bằng cách đánh tráo giữa đường.

Nghĩa là một người thứ ba C có thể đánh tráo các thông tin trao đổi giữa A và B. Chẳng hạn, C thay α^{a_A} mà A định gửi cho B bởi $\alpha^{a'_A}$ và thay α^{a_B} mà B định gửi cho A bởi $\alpha^{a'_B}$. Như vậy, sau khi thực hiện giao thức trao đổi khoá, A đã lập một khoá chung $\alpha^{a_A a'_B}$ với C mà vẫn tưởng là với B; đồng thời B cũng lập một khoá chung $\alpha^{a'_A a_B}$ với C mà vẫn tưởng là với A. C có thể giả mã mọi thông báo mà A tưởng nhầm là mình gửi đến B cũng như mọi thông báo mà B tưởng nhầm là mình gửi đến A.

Một cách khắc phục kiểu tấn công này là làm sao để A và B có kiểm thử để xác nhận tính đúng đắn của các khoá công khai b_A và b_B. Người ta đưa vào giao thức trao đổi khoá Diffie-Hellman thêm vai trò điều phối của một TA để được một hệ phân phối khoá Diffie-Hellman như một cách khắc phục nhược điểm này. Trong hệ phân phối khoá Diffie-Hellman, sự can thiệp của TA là rất yếu, thực ra TA chỉ làm mỗi việc là cấp chứng chỉ xác nhận khoá công khai cho từng người dùng chứ không đòi hỏi biết thêm bất cứ một bí mật nào của người dùng. Tuy nhiên, nếu chưa thoả mãn với vai trò hạn chế đó của TA thì có thể cho TA một vai trò xác nhận yếu hơn, không liên quan gì đến khoá, chẳng hạn như xác nhận thuật toán kiểm thử chữ ký của người dùng, còn bản thân các thông tin về khoá (cả bí mật lẫn công khai) thì do các người dùng trao đổi trực tiếp với nhau. Với cách khắc phục có vai trò hết sức hạn chế đó của TA, ta được giao thức sau đây:

3.2. Giao thức trao đổi khoá Diffie-Hellman có chứng chỉ xác nhận

Mỗi người dùng A có một danh tính ID(A) và một sơ đồ chữ ký với thuật toán ký sig_A và thuật toán kiểm thử ver_A. TA cũng có một vai trò xác nhận, nhưng không phải xác nhận bất kỳ thông tin nào liên quan đến việc tạo khoá mật mã của người dùng (dù là khoá bí mật hay khoá công khai), mà chỉ là xác nhận một thông tin ít quan hệ khác như thuật toán kiểm thử chữ ký của người dùng. Còn bản thân các thông tin liên quan đến việc tạo khoá mật mã thì các người dùng sẽ trao đổi trực tiếp với nhau. TA cũng có một sơ đồ chữ ký của mình, gồm một thuật toán ký sig_{TA} và một thuật toán kiểm thử công khai ver_{TA}. Chứng chỉ mà TA cấp cho mỗi người A sẽ là:

$C(A) = (ID(A), ver_A, sig_{TA}(ID(A), ver_A))$.

Rõ ràng trong chứng chỉ đó TA không xác nhận bất kỳ điều gì liên quan đến việc tạo khoá của A cả. Việc trao đổi khoá giữa hai người dùng A và B được thực hiện theo giao thức sau đây:

1) A chọn ngẫu nhiên số a_A ($0 \le a_A \le p-2$), tính $b_A = \alpha^{a_A} \bmod p$ và gửi b_A cho B.

2) B chọn ngẫu nhiên số a_B ($0 \le a_B \le p-2$), tính $b_B = \alpha^{a_B} \bmod p$ tính tiếp $K = b_A^{a_B} \bmod p$, $y_B = sig_B(b_B, b_A)$, và gửi $(C(A), b_B, y_B)$ cho A.

3) A tính $K = b_B^{a_A} \bmod p$, dùng ver_B để kiểm thử y_B, dùng ver_{TA} để kiểm thử C(B), sau đó tính $y_A = sig_A(b_A, b_B)$ và gửi (C(A), y_A) cho B.

4) B dùng ver_A để kiểm thử y_A và dùng ver_{TA} để kiểm thử C(A).

Nếu tất cả các bước đó được thực hiện và các phép kiểm thử đều cho kết quả đúng đắn thì giao thức được kết thúc, và cả A và B đều có được khoá chung K. Do việc dùng các thuật toán kiểm thử nên A biết chắc giá trị b_B là của B và B biết chắc giá trị b_A của A, loại trừ khả năng một người C nào khác đánh tráo các giá trị đó giữa đường.

3.3. Giao thức trao đổi khoá Matsumoto-Takashima-Imai

Giao thức trình bày trong mục trên dùng ba lần chuyển tin qua lại để thiết lập một khoá chung. Các tác giả Nhật Matsumoto, Takashima và Imai đề nghị một cải tiến để chỉ dùng một giao thức gồm hai lần chuyển tin (một từ A đến B và một từ B đến A) để thoả thuận khoá như sau:

Ta giả sử rằng trước khi thực hiện giao thức, TA đã ký cấp chứng chỉ cho mỗi người dùng A theo cách trong giao thức trao đổi DH:

C(A) = (ID(A), b_A, sig_{TA}(ID(A), b_A)).

và thuật toán kiểm thử chữ ký ver_{TA} là công khai. Trong giao thức này, các b_A không trực tiếp tạo nên các khoá mật mã cho truyền tin, mà với mỗi phiên truyền tin bảo mật, khoá phiên (sesion key) sẽ được tạo ra cho từng phiên theo giao thức.

Giao thức trao đổi khoá phiên MTI gồm ba bước (trong đó có hai lần chuyển tin) như sau:

1) A chọn ngẫu nhiên số r_A ($0 \le r_A \le p-2$), tính $s_A = \alpha^{r_A} \bmod p$, và gửi (C(A), s_A) cho B.

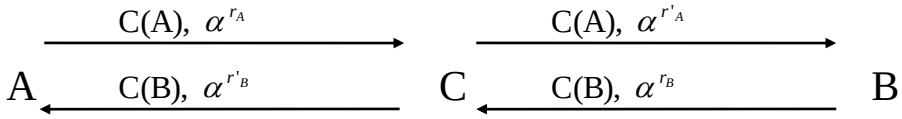
2) B chọn ngẫu nhiên số r_B ($0 \le r_B \le p-2$), tính $s_B = \alpha^{r_B} \bmod p$, và gửi (C(B), s_B) cho A.

3) A tính $K = s_B^{a_A}.b_B^{r_A} \bmod p$, với giá trị b_B thu được từ C(B)

B tính $K = s_A^{a_B}.b_B^{r_B} \bmod p$, với giá trị b_B thu được từ C(A).

Hai cách tính đó cho cùng một giá trị $K = \alpha^{r_A a_B + r_B a_A} \bmod p$.

Giao thức này cũng có khả năng giữ bí mật khoá K như đối với giao thức Diffie-Hellman trước sự tấn công thụ động. Tuy nhiên, vì không có chứng chỉ đối với các giá tri s_A, s_B nên vẫn có nguy cơ của sự tấn công tích cực bằng việc đánh tráo giữa đường bởi một người C nào đó theo kiểu sau đây:



Lẽ ra A gửi đến B cặp (C(A), s_A) thì C đánh tráo bằng cách (C(A), s_A) và gửi đến B giá trị (C(A), s'_A) với $s'_A = \alpha^{r'_A} \bmod p$. Và ngược lại, đáng lẽ B gửi đến A giá trị (C(B), s_B)

thì C đánh trao bằng cách nhận (C(B), s_B) và gửi đến A giá trị (C(B), s'_B) với $s'_B = \alpha^{r'_B} \bmod p$. Khi đó A tính được khoá:

$$K_1 = \alpha^{r_A a_B + r'_B a_A} \bmod p,$$

và B tính được khoá:

$$K_2 = \alpha^{r'_A a_B + r_B a_A} \bmod p.$$

Hai giá trị K_1 và K_2 này khác nhau nên không giúp A và B truyền tin được cho nhau, nhưng C không có khả năng tính được giá trị nào trong hai giá trị đó (vì không biết a_A và a_B) nên khác với giao thức Diffie-Hellman, ở đây C chỉ có thể phá rối, chứ không thể đánh cắp thông tin được.

3.4. Giao thức Girault trao đổi khoá không chứng chỉ

Giao thức Girault được đề xuất năm 1991. Trong giao thức này, người sử dụng A không cần dùng chứng chỉ C(A) mà thay bằng một khoá công khai tự chứng thực được cấp trước bởi một TA. Phương pháp này sử dụng kết hợp các đặc tính của bài toán RSA và logarit rời rạc.

Giả sử n là tích của hai số nguyên tố lớn p và q, n = p*q, p và q có dạng $p = 2p_1+1$, $q = 2q_1+1$, trong đó p_1 và q_1 cũng là các số nguyên tố. Nhóm nhân Z_n^* đẳng cấu với tích $Z_p^* x Z_q^*$. Cấp cao nhất của một phần tử trong Z_n^* là bội chung bé nhất của p-1 và q-1, tức là bằng $2p_1q_1$. Giả sử α là một phần tử cấp $2p_1q_1$ của Z_n^*. Nhóm tuần hoàn sinh bởi α được ký hiệu là G, bài toán tính logarit rời rạc theo cơ số α trong G được giả thiết là rất khó.

Các số n và α là công khai. Chỉ TA biết p, q. TA chọn số mũ công khai e với UCLN(e, $\phi(n)$) = 1, và giữ bí mật $d = e^{-1} \bmod \phi(n)$.

Mỗi người dùng A có một danh tính ID(A), chọn ngẫu nhiên một số $a_A \in G$, giữ bí mật a_A và tính $b_A = \alpha^{a_A} \bmod n$, rồi gửi a_A, b_A cho TA. TA thử lại điều kiện $b_A = \alpha^{a_A} \bmod n$, rồi cấp cho A một khoá công khai tự chứng thực $p_A = (b_A\text{-}ID(A))^d \bmod n$. Trong khoá công khai p_A không có thông tin về a_A nhưng TA cần biết a_A để thử điều kiện $b_A = \alpha^{a_A} \bmod n$.

Giao thức Girault trao đổi khoá giữa hai người dùng A và B được thực hiện bởi các bước sau đây:

1) A chọn ngẫu nhiên $r_A \in G$, tính $s_A = \alpha^{r_A} \bmod n$ và gửi cho B các giá trị (ID(A), p_A, s_A).

2) B chọn ngẫu nhiên $r_B \in G$, tính $s_B = \alpha^{r_B} \bmod n$ và gửi cho B các giá trị (ID(B), p_B, s_B).

3) A tính khoá $K = s_B^{a_A} (p_B^e + ID(V))^{r_A} \bmod n,$

B tính khoá $K = s_A^{a_B} (p_A^e + ID(A))^{r_B} \bmod n.$

Cả hai giá trị đó của K đều bằng nhau và bằng $K = \alpha^{r_A a_B + r_B a_A} \bmod n.$.

Bằng các lập luận tương tự như ở mục trước, ta dễ thấy rằng một người thứ ba C khó mà tạo ra các thông tin giả mạo để gửi đến A hoặc B, nếu tấn công bằng cách đánh tráo giữa đường thì có thể phá rối để ngăn cản A và B tạo lập khoá chung nhưng không thể đánh cắp thông tin trao đổi giữa A và B.

Còn lại vấn đề: tại sao TA cần biết a_A và thử điều kiện $b_A = \alpha^{a_A} \bmod n$ trước khi cấp p_A cho A! Ta giả sử rằng TA không biết a_A và cấp $p_A = (b_A\text{-ID}(A))^d \bmod n$ cho A , và thử xem có thể xảy ra chuyện gì?

Một người thứ ba C có thể chọn một giá trị a'_A và tính $b'_A = \alpha^{a'_A} \bmod n$, rồi tính $b'_C = b'_A - ID(A) - ID(C)$ và đưa (ID(C), b'_C) cho TA. TA sẽ cấp cho C một "khoá công khai tự chứng thực":

$p'_C = (b'_C - ID(C))^d \bmod n$.

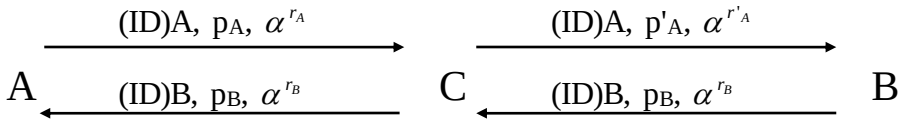
Vì $b'_C - ID(C) = b'_A - ID(A)$ nên thực tế C đã được cấp:

$p'_C = p'_A = (b'_A - ID(A))^d \bmod n$.

Bây giờ giả sử A và B thực hiện giao thức trao đổi khoá và C xen vào ở giữa. Như vậy, A gửi cho B $(ID(A), p_A, \alpha^{r_A} \bmod n)$, nhưng do C đánh tráo nên B sẽ nhận được $(ID(A), p'_A, \alpha^{r'_A} \bmod n)$. Do đó, B và C tính được cùng một khoá:

$$K' = \alpha^{r'_A a_B + r_B a'_A} \bmod n = s_B^{a'_A}(p_B^e + ID(B))^{r'_A} \bmod n,$$

còn A tính được khoá $K = \alpha^{r_A a_B + r_B a_A} \bmod n.$



B và C có cùng một khoá khác với khoá của A nhưng B vẫn nghĩ rằng mình có chung khoá với A. Vì thế, C có thể giải mã mọi thông báo mà B gửi cho A, tức đánh cắp thông tin từ B đến A. Việc TA biết a_A và thử điều kiện $b_A = \alpha^{a_A} \bmod n$ trước khi cấp p_A cho A là để loại trừ khả năng đánh tráo như vậy của một kẻ tấn công C.

4.Bài tập

Bài tập 6.1: Giả sử A và B sử dụng kỹ thuật phân phối khóa Diffie-Hellman để truyền tin cho nhau với số nguyên tố được chọn là p = 71 và phần tử nguyên thủy α = 7.

a) Nếu khóa bí mật của A là X_A = 5 thì khóa công khai của A là gì?
b) Nếu khóa bí mật của B là X_B = 12 thì khóa công khai của B là gì?
c) Cho biết khóa bí mật dùng để truyền tin?

Bài tập 6.2: A và B sử dụng kỹ thuật phân phối khóa Diffie-Hellman để truyền tin cho nhau với p = 11 và phần tử nguyên thủy α = 2.

a) Hãy chứng minh rằng α = 2 đúng là phần tử nguyên thủy của Z^*_{11}.
b) Nếu khóa công khai của A là Y_A = 9 thì khóa bí mật của A là bao nhiêu?

c) Giả sử B có khóa công khai là $Y_B = 3$, hãy tìm khóa bí mật dùng để truyền tin giữa A và B.

CHƯƠNG VII: GIAO THỨC MẬT MÃ

1. Giao thức

Định nghĩa:

Một giao thức (protocol) chỉ đơn giản là một chuỗi các bước thực hiện trong đó có ít nhất 2 bên tham dự, được thiết kế để thực hiện một nhiệm vụ nào đó.[2]

Định nghĩa này đơn giản nhưng chặt chẽ: "một chuỗi các bước" nghĩa là một dãy các bước có thứ tự, có đầu có cuối, bước trước phải được kết thúc trước khi thực hiện bước sau. "Có ít nhất hai bên tham gia" nghĩa là có thể có nhiều người cùng tham gia thực hiện chuỗi bước này, do đó nếu một người thực hiện một chuỗi các bước thì không thể gọi là một giao thức được. Và cuối cùng một giao thức phải được thiết kế nhằm đạt được tới một kết quả nào đó.

Một giao thức có những đặc tính như sau:

– Các bên tham gia phải hiểu cách thức và các bước thực hiện một giao thức khi tham gia thực hiện.

– Các bên phải đồng ý tuyệt đối tuân thủ các bước.

– Giao thức phải rõ ràng, tất cả các bước phải được viết tường minh, không có chỗ nào gây nên khả năng hiểu nhầm.

– Giao thức phải đầy đủ, tất cả các tình huống biến đổi đều phải được đưa ra.

Giao thức mật mã là một giao thức có vận dụng các kiến thức của lý thuyết mật mã để đạt được các mục tiêu về mặt an toàn và bảo mật cho hệ thống. Các thành phần tham gia có thể là bạn bè tin tưởng lẫn nhau, nhưng cũng có thể là những kẻ địch của nhau. Một giao thức mật mã có liên quan đến các thuật toán của mật mã nhưng thông thường mục đích của nó đi xa hơn là tính bảo mật thuần tuý. Các bên có thể tham dự vào việc chia sẻ các phần của một bí mật được dùng để chiết xuất ra một thông tin nào đó, có thể cùng kết hợp phát ra một chuỗi số ngẫu nhiên, có thể chứng minh danh tính của mình cho bên kia hay đồng thời ký vào một văn bản hợp đồng. Toàn bộ vấn đề của lý thuyết mật mã ở đây là làm sao dò ra và chống lại các khả năng nghe trộm hay lừa dối.

Nguyên tắc để thiết kế giao thức: phải làm sao để không ai, không bên nào có thể thu được nhiều hơn, biết được nhiều hơn những gì mà thiết kế ban đầu giả định.

2. Mục đích của các giao thức

Ngày nay, với sự phát triển vũ bão của hệ thống máy tính toàn cầu đi đến từng hộ gia đình, việc đưa các nghi thức thủ tục làm ăn bình thường của người ta thực hiện qua mạng cũng là không bao xa. Như vậy cần phải thiết kế những thủ tục làm việc tương ứng cho máy tính để có thể thay thế cho các thủ tục trong đời thường. Điểm khác biệt đặc trưng ở đây là bây giờ người làm việc với nhau thông qua các máy tính mà không cần thấy mặt nhau nữa. Hơn nữa máy tính không phải là người, nó không thể dễ dàng thích nghi với thay đổi như chúng ta đây. Vì vậy cần tính đến mọi tình huống, mọi khả năng có thể của giao thức.

Rất nhiều các thủ tục làm ăn hàng ngày của chúng ta được tin tưởng dựa trên sự có mặt cùng nhau của các bên đối tác, chính vì thế nên việc xây dựng những giao thức trên máy tính là không còn đơn giản như các thủ tục đời thường mà nó thay thế. Bạn cứ tự hỏi xem người ta có thể trao một chồng tiền mặt cho một người lạ để nhờ mua hàng có được không? Hay thử hỏi xem bạn có dám gửi thư cho chính phủ với phiếu bầu của bạn mà không có các thủ tục đảm bảo về việc giấu tên. Thật là ngây thơ nếu tin rằng mọi người làm việc trên mạng máy tính đều trung thực. Và cũng thật là cả tin nếu cho rằng các nhà quản trị mạng, hay thậm chí ngay cả các nhà thiết kế ra các mạng này là trung thực đến cùng. Dù hầu hết là như thế nhưng chỉ cần một thiểu số những người không trung thực cũng đủ ngây ra thiệt hại nếu chúng ta không có các biện pháp đảm bảo.

Với phương pháp hình thức hoá, chúng ta có thể thử thiết kế các giao thức rồi tìm hiểu, kiểm tra khả năng của nó có vững hay không trước mọi kiểu xâm phạm của các kẻ không trung thực; từ đó mà cải tiến, phát triển lên để chống lại các kiểu tấn công đó. Bằng cách đó mà người ta đã xây dựng các giao thức cho các máy tính giải quyết được các nhiệm vụ, các bài toán đời sống hàng ngày.

Hơn nữa giao thức máy tính là một hình thức trừu tượng hoá và không quan tâm đến việc cài đặt cụ thể. Một giao thức là giống nhau dù nó được cài đặt trên bất cứ hệ điều hành nào. Vì thế một khi chúng đã có thể khẳng định được độ tin cậy của giao thức ta có thể áp dụng nó ở bất cứ đâu, dù là cho máy tính, cho điện thoại hay cho một lò vi sóng thông minh ...

3. Các bên tham gia vào giao thức (the players in protocol)

Để có thể tiếp cận thống nhất với tất cả các giao thức thì một điều cần thiết là có một qui định thống nhất cách gọi tên tất cả các bên tham gia và dính líu có thể có trong giao thức: [6]

Alice	bên thứ nhất trong các giao thức.
Bob	bên thứ hai trong các giao thức.
Carol	bên tham gia thứ ba trong các giao thức.
Dave	bên tham gia thứ tư trong các giao thức.
Eve	kẻ nghe trộm (eavesdropper).
Mallory	kẻ tấn công chủ động có nhiều quyền lực trên mạng và rất nguy hiểm (malicious active attacker).
Trent	trọng tài (trusted arbitrator).
Walter	người canh gác (warden), có thể đứng canh gác Alice và Bob trong một số giao thức .
Peggy	người chứng minh (prover).
Victor	người thẩm tra (verifier), Peggy cần phải chứng minh với Victor về một quyền sở hữu nào đó chẳng hạn như danh tính của anh ta khai là đúng hay anh ta đúng là kẻ có thẩm quyền để được truy nhập vào một nơi quan trọng ...

4. Các dạng giao thức

4.1. Giao thức có trọng tài

Người trọng tài là người thoả mãn các điều kiện sau:

- Không có quyền lợi riêng trong giao thức và không thiên vị cho một bên nào.
- Các bên tham gia có quyền lợi trong giao thức đều tin tưởng vào trọng tài rằng bất kỳ cái gì mà anh ta nói và làm đều là đúng và chính xác, đồng thời tin tưởng anh ta sẽ hoàn thành trách nhiệm của mình trong giao thức.

Như vậy trọng tài có thể đứng ra để giúp hoàn thành các giao thức giữa những bên tham gia không tin tưởng lẫn nhau.

Ví dụ 1:

Alice muốn bán một chiếc xe cho một người lạ là Bob. Bob muốn trả bằng séc, tuy nhiên Alice lại không có cách nào để biết được séc đó có giá trị thật sự hay không. Do vậy, cô ta chỉ muốn được chuyển séc trước khi giao xe cho Bob và đấy chính là mâu thuẩn bế tắc vì Bob cũng chẳng tin gì Alice nên anh ta sẽ không đưa séc trước khi nhận được chiếc xe.

Cách giải quyết sẽ thông qua Trent (người mà cả Bob và Alice đều tin tưởng) và một giao thức sẽ diễn ra như sau để đảm bảo tính trung thực:

- Alice chuyển vật cần bán cho Trent
- Bob đưa tờ séc cho Alice.
- Alice chuyển séc vào tài khoản của cô ta ở ngân hàng.
- Đợi một khoảng thời gian nhất định đến khi séc đã chuyển xong, Trent sẽ giao hàng cho Bob. Nếu tờ séc không hợp lệ thì Alice sẽ báo cho Trent biết với bằng chứng cụ thể và Trent sẽ giao trả lại hàng cho cô ta.

Trong giao thức này:

- Alice tin tưởng rằng Trent sẽ không trao hàng cho Bob trừ khi séc được chuyển xong và sẽ chuyển lại hàng cho cô ta nếu séc không có giá trị.
- Bob tin tưởng Trent sẽ giữ hàng trong thời gian séc được chuyển và sẽ giao nó cho anh ta một khi được chuyển xong.
- Trent không quan tâm đến việc tờ séc có giá trị thật sự và có chuyển được hay không, anh ta làm phần việc của mình trong cả hai trường hợp có thể xảy ra đúng như giao thức qui định, đơn giản vì anh ta sẽ được trả tiền công trong cả hai trường hợp.

Ví dụ 2:

Nhà băng cũng có thể đứng ra làm trọng tài cho ALice và Bob. Bob sử dụng một cái séc có chứng nhận của nhà băng để mua bán với Alice:

- Bob viết một séc và chuyển cho nhà băng.
- Sau khi cầm một số tiền từ tài khoản của Bob bằng giá trị của tờ séc, nhà băng ký chứng nhận lên séc và chuyển trả lại cho Bob.

– Alice giao xe cho Bob cùng lúc Bob đưa Alice tờ séc có chứng nhận của nhà băng.

– Alice chuyển séc vào nhà băng.

Giao thức này thực hiện được bởi vì Alice tin tưởng vào chứng nhận của nhà băng, tin rằng nhà băng cầm giữ số tiền của Bob cho cô ta mà không sử dụng nó vào đầu tư ở bất cứ đâu.

Tư tưởng này được đem áp dụng vào thế giới máy tính, tuy nhiên ở đây xuất hiện một số vấn đề nhất định đối với hệ thống máy tính:

– Có thể dễ dàng tìm thấy và đặt lòng tin vào một bên thứ ba trung gian (trọng tài) nếu ta biết và có thể nhìn tận mặt họ. Tuy nhiên nếu hai bên tham gia giao thức đã nghi ngờ nhau thì việc cùng đặt lòng tin vào một bên thứ ba nào đó nằm đâu đó khuất diện trên mạng máy tính cũng trở nên có thể đáng ngờ.

– Mạng máy tính phải tốn thêm chi phí để quản lý và bảo trì máy tính trọng tài.

– Luôn luôn có những khoảng trễ vốn gắn liền với bất kỳ một giao thức có trọng tài nào.

– Trọng tài phải tham gia vào mọi giao dịch trên mạng, điều đó có nghĩa ở đó sẽ trở nên một điểm thắt nút cổ chai (bottleneck), dễ tắc trên mạng một khi giao thức đã được triển khai cho một ứng dung rộng rãi. Tăng cường số trọng tài có thể giúp tránh bế tắc này nhưng lại làm tăng thêm chi phí để quản lý bảo trì những máy tính có trọng tài đó.

– Bởi vì tất cả mọi người trên mạng đều tin trọng tài, dễ gây ra ở đây một điểm nhạy cảm chịu áp lực tấn công tập trung từ các kẻ rình rập để phá hệ thống.

4.2. Giao thức có người phân xử

Để yên tâm giao dịch, Alice và Bob cần mời một trọng tài có uy tín cao, tuy nhiên ở đây sẽ nảy sinh vấn đề về việc phải trả số tiền xứng đáng cho người này, rõ ràng là không phải không đáng kể. Vì vậy người ta đã nảy sinh ý nghĩ chia giao thức có trọng tài tham dự (arbitrated protocol) thành hai phân giao thức (subprotocol) ở hai cấp dưới:

– Một là một giao thức không cần đến trọng tài, thực hiện bất kỳ khi nào muốn tiến hành giao dịch.

– Hai là một arbitrated giao thức chỉ được sử dụng khi Alice và Bob cãi nhau và muốn có người phân xử.

Vì thế trong trường hợp này ta không dùng khái niệm người trọng tài (arbitrated) với nghĩa là người phải trực tiếp tham gia vào giao thức, mà sử dụng người phân xử (adjudicator), bao hàm ý nghĩa người này không cần phải có mặt khi Alice và Bob tiến hành giao dịch mà chỉ được mời đến khi Alice và Bob yêu cầu giải quyết tranh cãi.

Cũng giống như trọng tài, người phân xử phải không có quyền lợi liên can đến giao dịch của Alice và Bob, và được cả hai người này tin tưởng. Anh ta không tham gia trực tiếp vào giao dịch như trọng tài nhưng sẽ đứng ra để xác định xem là giao dịch có được tiến hành đúng không và xác định bên sai bên đúng nếu như có tranh cãi.Nhưng điểm khác biệt giữa trọng tài và người phân xử là người phân xử không phải luôn luôn cần thiết, nếu có tranh cãi thì mới cần người phân xử (không có tranh cãi thì thôi).

Các thẩm phán là những người phân xử chuyên nghiệp. Khác với công chứng viên, một thẩm phán - người mà sẽ chỉ được biết đến hợp đồng này khi nào một trong hai người Alice hay Bob lôi người kia ra toà. Giao thức dùng cho ký kết hợp đồng này có thể được hình thức hoá như sau:

Ví dụ:

Tại mọi thời điểm:

- Alice và Bob thoả thuận các điều khoản trong hợp đồng.
- Alice ký hợp đồng.
- Bob ký hợp đồng.

Khi có tranh cãi cần giải quyết:

- Alice và Bob đến gặp quan toà nhờ phân xử.
- Alice đưa ra chứng cớ của cô ta.
- Bob trình bày các chứng cớ của anh ta.
- Quan toà xem xét các chứng cớ và phán quyết.

Ý tưởng dùng người phân xử này có thể đem vào áp dụng trên máy tính. Trong những giao thức thế này nếu có một bên tham gia mà không trung thực thì dữ liệu lưu được từ giao thức sẽ cho phép người phân xử sau này phát hiện được ai là người đã lừa dối. Như vậy thay vì ngăn chặn trước sự lừa đảo, giao thức người phân xử sẽ phát hiện được lừa dối nếu xảy ra, thực tế này khi được phổ biến rộng rãi sẽ có tác dụng ngăn chặn, làm lùi bước những kẻ có ý định lừa đảo.

4.3. Giao thức tự phân xử

Giao thức tự phân xử là loại tốt nhất trong số các giao thức. Loại giao thức này tự bản thân nó có thể đảm bảo được tính công bằng, không cần đến trọng tài hay một thẩm phán để phân xử khi tranh cãi. Nghĩa là giao thức loại này được chế ra sao cho không thể có các kẽ hở cho tranh cãi nảy sinh. Nếu có bên nào cố ý sai luật thì tiến trình sẽ cho phép phía bên kia phát hiện ra ngay và giao thức dừng lại ngay lập tức. Điều mong muốn cho tất cả các giao thức đều nên chế tạo như thế, nhưng đáng tiếc là không phải lúc nào cũng có giao thức loại này cho mọi tình huống.

5. Các dạng tấn công đối với giao thức

Nếu như giao thức được coi như một nghi thức giao tiếp để các bên làm việc với nhau thì đối với cryptography giao thức, bên dưới cái vỏ "ngoại giao" đó là các kỹ thuật, các thuật toán mật mã được vận dụng, cài đặt trong các bước cụ thể của giao thức. Các tấn công của kẻ phá hoại nhằm phá hoại tính an ninh của hệ thống cũng như xâm phạm tính bí mật riêng tư của thông tin, có thể hướng vào một trong các yếu tố sau: các xử lý kỹ thuật, các thuật toán mật mã hay là chính bản thân giao thức.

Trong phần này, chúng ta hãy gác lại khả năng thứ nhất - giả sử rằng các kỹ thuật và thuật toán mật mã đều là an toàn; chúng ta chỉ xem xét khả năng thứ hai, tức là phân tích các dạng tấn công có thể, trong đó kẻ thù lợi dụng các kẽ hở logic để kiếm lợi hay phá hoại. Các dạng tấn công có thể phân thành hai loại chính như sau:

– Với dạng tấn công thụ động: kẻ địch chỉ đứng ngoài nghe trộm chứ không can thiệp hay ảnh hưởng gì đến giao thức. Mục đích của nó là cố gắng quan sát và thu lượm thông tin. Tuy nhiên thông tin nghe trộm được chỉ ở dạng mã hoá, do đó kẻ địch cần phải biết cách phân tích, giải mã thì mới dùng được (cipher only attack). Mặc dù hình thức tấn công này không mạnh nhưng rất khó phát hiện vì kẻ địch không gây động.

– Với dạng tấn công chủ động (active attack): kẻ địch là một thế lực trong mạng, nắm nhiều khả năng và phương tiện để có thể chủ động tấn công can thiệp, gây ảnh hưởng phức tạp đến giao thức. Nó có thể đóng giả với một cái tên khác can thiệp vào giao thức bằng những thông báo kiểu mới, xoá bỏ những thông báo đang phát trên đường truyền, thay thế thông báo thật bằng thông báo giả, ngắt ngang các kênh thông tin hay sửa chửa vào các kho thông tin trên mạng. Các khả năng khác nhau này là phụ thuộc vào tổ chức mạng và vai trò của kẻ địch trên mạng.

Kẻ tấn công trong tấn công thụ động (Eve) chỉ cố gắng thu lượm thông tin từ các bên tham gia giao thức, thông qua thu nhập các thông báo truyền tin giữa các bên để phân tích giải mã. Trong khi đó, kẻ tấn công chủ động (Mallory) có thể gây ra các tác hại rất phức tạp đa dạng. Kẻ tấn công có thể có mục đích đơn thuần là tóm được tin mà nó quan tâm, nhưng ngoài ra nó có thể gây ra các phá hoại khác như phá hoại đường truyền truy nhập vào những hệ thống thông tin mà chỉ dành cho những người có đủ thẩm quyền.

Kẻ địch trong tấn công chủ động thật sự rất nguy hiểm, đặc biệt là trong các giao thức mà các bên khác nhau không nhất thiết phải tin nhau. Hơn nữa phải nhớ rằng kẻ địch không phải chỉ có thể là những kẻ xa lạ bên ngoài mà nó có thể là một cá nhân hợp pháp trong hệ thống, thậm chí ngay chính là người quản trị mạng. Ngoài ra còn có thể có nhiều cá nhân liên kết với nhau thành một nhóm kẻ địch, làm tăng lên sự nguy hiểm cho giao thức.

Một điều cũng có thể xảy ra là Mallory lại chính là đối tác trong giao thức. Anh ta có thể có hành động lừa dối hoặc là không chịu tuân theo giao thức. Loại kẻ địch này được là kẻ lừa đảo (cheater). Kẻ lừa đảo thuộc loại thụ động thì có thể làm đúng theo giao thức nhưng lại cố tình thu nhặt thêm thông tin từ các bên đối tác hơn là được phép theo qui định. Kẻ lừa đảo chủ động thì phá vỡ giao thức trong một cố gắng lừa dối. Rất khó để giữ an toàn cho một giao thức nếu như phần lớn các bên tham gia đều là những kẻ lừa đảo chủ động, tuy nhiên đôi khi người ta cũng có các biện pháp để các bên hợp pháp có thể dò ra được sự lừa đảo đang diễn ra. Tất nhiên các giao thức cũng cần phải được bảo vệ để chống lại những kẻ lừa đảo loại thụ động.

TÀI LIỆU THAM KHẢO

[1] Nik Goots, Boris Izotov, Alex Moldovyan and Nik Moldovyan, "*Modern Cryptography-Protect Your Data with Fast Block Ciphers*", A-LIST Publishing , 2003.

[2] Whitfield Diffie, Martin E. Hellman, "*New Directions in Cryptography*", IEEE transactions on information theory, Vol. IT-22, No. 6, November 1976.

[3] Randy Nichols (LANAKI), "*Classical cryptography course*", 1995. http://www.forturecity.com/course/LANAKI.html

[4] A.Menezes, P. van Oorchot, and S.Vanstone, "*Hand book of Applied Cryptography*", CRC Press, 1996. http://www.cacr.math.uwaterloo.ca/hac

[5] Douglas R.Stinson, "*Cryptography: theory and practice*", CRC Press, 1995.http://www.mindspring.com/~pate/stinson/

[6] Bruce Schneier, "*Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C (cloth)*", MIST Press, 1996.

[7] Gil Held, "*Learn Encryption Techniques with BASIC and C++*", CRC Press, 1998.

[8] FIPS 186 - (DSS)http://www.itl.nist.gov/fipspubs/fip186.htm

[9] Jean Berstel, Dominique, "*Theory of code*", Academic Press Inc, 1985.

[10] C. Shannon, "*Communication theory of secret systems*" (tạp chí khoa học), 1949.

[11] RSA library. www.fpt.rsa.org/PKI

[12] "*System and Network Security*". http://www.cs.ncl.ac.uk/old/modules/2000-01/csc331/notes/

[13] "*Cryptography and Computer Security*". http://www.cs.adfa.edu.au/teaching/studinfo/csc/lectures/

[14] http://www.securitydynamics.com/rsalabs/changelenges/factoring/rsa155.html.

[15] "*Data security and cryptography*". http://www.islab.oregonestate.edu/koc/ece575

[16] "*OPT8 Advanced Cryptography*". http://www.isg.rhul.ac.uk/msc/teaching/opt8/macs.pdf

Đề 1:

Câu 1 : Cho hệ mã Hill có M = 2 và ma trận khóa $A = \begin{bmatrix} 12 & 5 \\ 3 & 7 \end{bmatrix}$ hãy thực hiện mã hóa với xâu S = "HARD".

Câu 2 : Vẽ mô hình quản lý khóa dựa vào hệ mã khóa công khai. Giải thích rõ các chức năng và các bước thực hiện.

Câu 3: Các mệnh đề sau đúng hay sai, giải thích?
1. So với tấn công chủ động tấn công thụ động nguy hiểm hơn.
2. Giao thức 3 bước Shamir hỗ trợ khả năng xác thực hóa nguồn gốc thông điệp.
3. Cơ chế mã móc xích an toàn hơn cơ chế bảng tra mã điện tử
4. Một trong các yếu điểm của các hệ mã mật khóa công khai là chậm.
5. Giao thức 3 bước Shamir là giao thức trao đổi thông tin không cần trao đổi khóa.
6. Các hệ mã mật RSA, ElGamma, Knapsack được gọi là các hệ mã mật khóa công khai vì khóa của chúng đều được công khai hóa.

Đề 2:

Câu 1 : Vẽ lược đồ chế độ sử dụng mã khối móc xích CBC. Mô tả thuật toán sinh và giải mã.

Câu 2 : Cho khóa K = $\begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix}$ và tin gốc là 'July' xác định trên trường Z_{26}. Tìm tin mã theo giải thuật Hill – cipher.

Câu 3: Các mệnh đề sau đúng hay sai, giải thích?
1. Tất cả có 4 loại hàm băm: các hàm băm dựa vào các hệ mã khối (chẳng hạn như DES), các hàm băm dựa vào các phép tính số học, các hàm băm đặc biệt và các hàm băm dựa vào các hệ mã khóa công khai.
2. Một trong các yếu điểm chính của hệ Knapsack là việc lưu khóa cần bộ nhớ lớn.
3. Chuẩn mã hóa dữ liệu (DES) không còn an toàn nên không còn được dùng trong thực tế.
4. Để tăng tính bảo mật cho DES có thể mã hóa nhiều lần với các khóa khác nhau.
5. Trong hệ mã ElGamma luôn xuất hiện hiện tượng lộ bản rõ.
6. Để sử dụng cơ chế bảng tra mã điện tử (EBC) khi cài đặt không cần có một gía trị khởi tạo IV.

Đề 3:

Câu 1 : Vẽ lược đồ chế độ sử dụng mã khối phản hồi CFB. Mô tả thuật toán sinh và giải mã.

Câu 2 : Cho véc tơ siêu tăng A = (1, 2, 4, 8, 16, 32, 64, 128), m = 301, u = 31, và tin gốc (bản rõ) là 10. Tìm tin mã (bản mã) theo giải thuật Knapsack.

Câu 3: Các mệnh đề sau đúng hay sai, giải thích?

1. Trong chế độ mã móc xích thông điệp được chia thành n khối, nếu như khối thứ i bị lỗi trước khi đem mã hóa thì sẽ làm ảnh hưởng tới các khối mã hóa sau đó.
2. Cho N = 2000, khi đó giá trị hàm Ơ le của N: $\Phi(N) = 800$.
3. Giao thức 3 bước Shamir là giao thức trao đổi thông tin không cần trao đổi khóa.
4. Các hệ chữ ký điện tử hoạt động theo 3 bước: sinh chữ ký, gửi chữ ký và kiểm tra chữ ký.
5. Các hệ mã mật SKC và PKC đều cho phép sử dụng trong mô hình chữ ký điện tử.
6. Cơ chế mã móc xích an toàn hơn cơ chế bảng tra mã điện tử.

Đề 4:

Câu 1 : Vẽ lược đồ giải thuật sinh mã DES và giải thích các công thức được dùng.

Câu 2 : Cho véc tơ siêu tăng a = (1, 2, 4, 8, 16, 32, 64, 128), m = 300, w = 29, và tin gốc là 16. Tìm tin mã theo giải thuật Knapsack.

Câu 3: Các mệnh đề sau đúng hay sai, giải thích?

1. Từ luật Kierchoff suy ra muốn tăng độ an toàn của một hệ mã mật cần sử dụng thuật toán mã hóa càng phức tạp càng tốt.
2. So với kiểu tấn công thụ động kiểu tấn công chủ động khó phát hiện hơn và nguy hiểm hơn.
3. Giao thức 3 bước Shamir là giao thức trao đổi thông tin không cần trao đổi khóa.
4. Một trong các yếu điểm chính của hệ Knapsack là việc lưu khóa cần bộ nhớ lớn.
5. Điều kiện để giao thức 3 bước Shamir hoạt động là:
$$E_{Z2}^{-1}(E_{Z1}(E_{Z2}(X))) = E_{Z2}(X).$$
6. Các hệ mã mật khóa công khai thường được gọi là PKC trong đó PKC có nghĩa là Private Key Cryptography.

Đề 5:

Câu 1 : Vẽ lược đồ sinh khóa từ khóa chính của DES và giải thích các công thức được dùng.

Câu 2 : Cho p = 13, q = 23, e = 173, và tin mã là 122. Tìm tin gốc theo giải thuật RSA.

Câu 3: Các mệnh đề sau đúng hay sai, giải thích?

1. Cơ chế CBC là cơ chế sử dụng mã khối đơn giản nhất và dễ dùng nhất.
2. Trong cơ chế ECB nếu một khối nào đó bị hỏng trước khi đưa vào mã hóa sẽ làm ảnh hưởng tới tất cả các khối mã hóa đứng trước nó.
3. Khóa mã hóa của chuẩn mã hóa dữ liệu có độ dài bằng 56 bit.
4. Các chế độ sử dụng mã khối đều sử dụng các đơn vị khối dữ liệu 64 bit..
5. Trong hệ mã ElGamma luôn xuất hiện hiện tượng lộ bản rõ.
6. Cơ chế mã móc xích an toàn hơn cơ chế bảng tra mã điện tử.