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 - 2008

Tên học phần: An toạn bảo mật thông tin **Loại học phần**: **II Bộ môn phụ trách giảng dạy**: Khoa học máy tính.
Khoa phụ trách: Công nghệ thông tin
Mã học phần: **Tổng số TC**: **3**

TS tiết	Lý thuyết	Thực hành/ Xemina	Tự học	Bài tập lớn	Đồ án môn học
75	45	30	0	0	0

Điều kiện tiên quyết:

Sinh viên cần hoc xong ca c hoc phần:

- Lập trịnh hươ ng đối tươn g
- Cấu trục dự liệu
- Phân tịch, thiết kế vạ đạnh gia thuật toa n.

Mục đích của học phần:

Truyền đạt cho sinh viên những kiến thức cơ bản về các lĩnh vực riêng trong an toàn bảo mật máy tính:

- Các giải thuật mã hóa trong truyền tin.
- Các thuật toán tạo hàm băm và chữ ký điện tử.
- Các mô hình trao chuyển khóa.
- Các mô hình chứng thực và các giao thức mật mã.

Nội dung chủ yếu:

Gồm 2 phần:

- Phần lythuyết: cung cấp cạc ly thuyết về thuật toạn mạho a, các giao thức. -

Phần lập trịnh: cài đặt các hệ mã, viết cạc ư ng duṇ g sự duṇ g ca c hệ mạ mật **Nội dung chi tiết của học phần:**

Tên chương mục	**Phân phối số tiết**				
	TS	LT	Xemine	BT	KT
Chương I. Giới thiệu nhiệm vụ của an toàn và bảo mật thông tin.	4	3	1	0	0
1.1. Các khái niệm mở đầu. 1.1.1. Thành phần của một hệ thống thông tin 1.1.2. Những mối đe dọa và thiệt hại đối với hệ thống thông tin. 1.1.3. Giải pháp điều khiển kiểm soát an toàn bảo mật 1.2. Mục tiêu và nguyên tắc chung của ATBM. 1.2.1. Ba mục tiêu. 1.2.2. Hai nguyên tắc 1.3. Giới thiệu chung về các mô hình mật mã. 1.3.1. Mô hình cơ bản trong truyền tin và luật Kirchoff. 1.3.2. Những giai đoạn phát triển của lý thuyết mã hóa.		1 1 1	1		

Chương II. Một số phương pháp mã hóa cổ điển.	13	5	5	2	1
2.1. Phương pháp mã đơn giản. 2.1.1. Mã hoán vị trong bảng Alphabet. 2.1.2. Mật mã cộng tính. 2.2.3. Mật mã nhân tính. 2.1.4. Phân tích mã theo phương pháp thống kê. 2.2. Phương pháp mã bằng phẳng đồ thị tần xuất. 2.2.1. Mã với bảng thế đồng âm. 2.2.2. Mã đa bảng thế: giải thuật mã Vigenre và One time pad. 2.2.3. Lý thuyết về sự bí mật tuyệt đối. 2.2.4. Đánh giá mức độ bảo mật của một phương pháp mã hóa. Kiểm tra		2 3	2 3	1 1	1
Chương III. Mật mã khối.	16	8	7	1	0
3.1. Khái niệm. 3.1.1. Điều kiện an toàn cho mật mã khối 3.1.2. Nguyên tắc thiết kế. 3.2. Chuẩn mạhọa dữ liệụDES 3.2.1. Lịch sử của DES 3.2.2. Cấu trúc vòng lặp DES. 3.2.3. Thuật toán sinh khóa con 3.2.4. Cấu trúc hàm lặp. 3.2.5. Thuật toán giải mã DES. 3.2.6. Đánh giá mức độ an toàn bảo mật của DES. 3.2.7. TripleDES 3.3. Chuẩn mạhọacaocấpAES 3.3.1. Giớithiệuvề AES 3.3.2. Thuật toạnmạhọa 3.3.3. Thuật toạngiạima 3.3.4. CàiđặtAES 3.4 Một số chế độ sử dụng mã khối. 3.4.1. Chế độ bảng tra mã điện tử 3.4.2. Chế độ mã móc xích 3.4.3. Chế độ mã phản hồi		1 3 3 1	3 3 1	0,5 0,5	
Chương IV. Hệ thống mã với khóa công khai.	16	6	7	2	1

4.1. Khái niệm khóa công khai. 4.1.1. Đặc trưng và ứng dụng của hệ mã khóa công khai. 4.1.2. Nguyên tắc cấu tạo hệ khóa công khai 4.2. Giới thiệu một số giải thuật PKC phổ biến. 4.1.1. Hệ mã Trapdoor Knapsack. 4.1.2. Hệ mã RSA		1 1 2	1 3	2	
4.1.3. Hệ mã ElGamal Kiểm tra		2	3		1
Chương V. Chữ ký điện tử và hàm băm.	12	7	5	0	0
5.1. Chữ ký điện tử. 5.1.1. Định nghĩa. 5.1.2. Ứng dụng của chữ ký điện tử 5.2. Giới thiệu một số hệ chữ ký điện tử 5.2.1. Hệ chữ ký điện tử RSA 5.2.2. Hệ chữ ký điện tử ElGamal 5.2.3. Chuẩn chữ ký điện tử DSA 5.3. Hàm băm. 5.3.1. Định nghĩa. 5.3.2. Sinh chữ ký điện tử với hàm băm 5.4. Một số hàm băm thông dụng 5.4.1. Hàm băm MD5 5.4.2. Hàm băm SHA1		0,5 3 0,5 3	2 1,5 1,5		
Chương VI. Quản lý khóa trong hệ thống mật mã	8	5	3	0	0
6.1. Quản lý khóa đối với hệ SKC 6.1.1. Giới thiệu phương pháp quản lý khóa. 6.2. Quản lý khóa trong các hệ PKC 6.2.1. Giao thức trao chuyển khóa Needham – Schoeder 6.2.2. Giao thức trao đổi khóa Diffie-Hellman 6.2.3. Giao thức Kerberos		1 1 1 1 1	1 2		
Chương VII. Giao thức mật mã	6	3	2	0	1

7.1. Khái niệm giao thức mật mã 7.1.1. Định nghĩa giao thức mật mã 7.1.2. Mục đích giao thức mật mã. 7.1.3. Các bên tham gia vào giao thức mật mã 7.2. Tìm hiểu thiết kế các giao thức mật mã điển hình 7.2.1. Một số dạng tấn công đối với giao thức mật mã. 7.2.2. Giới thiệu một số giao thức mật mã. 7.3. Kiểm tra.		1 2	2		1

Nhiệm vụ của sinh viên: Lên lớp đầy đủ và chấp hành mọi quy định của Nhà trường.

Tài liệu học tập:

1. Phan Đình Diệu. *Lý thuyết mật mã và An toàn thông tin.* Đại học Quốc Gia Hà Nội.

2. Douglas R. Stinson. *Cryptography Theory and practice*. CRC Press. 1995. 3. A. Menezes, P. VanOorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press. 1996.

4. William Stallings. *Cryptography and Network Security Principles and Practices, Fourth Edition*. Prentice Hall. 2005.

5. MichaelWelschenbach. *Cryptography in C and C++*. Apress. 2005.

Hình thức và tiêu chuẩn đánh giá sinh viên:

- Sinh viên phải làm các bài kiểm tra trong quá trình học và thực hành. Thi vấn đáp. - Sinh viên phải bảo đảm các điều kiện theo Quy chế của Nhà trường và của Bộ.

Thang điểm : Thang điểm 10.

Điểm đánh giá học phần: Z = 0,3 X + 0,7 Y**.**

MỤC LỤC

DANH MỤC HÌNH VẼ

Danh mục bảng

DANH MỤC BẢNG

Lợi noi đầu

LỜI NÓI ĐẦU

Từ trqớc công nguyên con ngqời 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 đqợc số hóa và xử lý trên máy tính, đqợc truyền đi trong một môi trqờng 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 troọng 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 hoc 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 chqơng đqợc 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 đqợc 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 , nhqng ngqời thân đã luôn động viên, góp ý cho tôi trong quá trình biên soạn . Xin gqi lợi cạm ơn tợi Thạc sy Nguyễn Định Dqơng, ngqời đạđoc va cho nhqng nhận xét , góp ý quí báu cho phần viết về hệmakhoa công khai dqa trên cạc đqợng cong Elliptic. Xin gqi lợi cạm ơn sâu sắc tợi Thạc sỹ Phạm Tuấn Đạt, ngqời đạhiệu định một cạch kycạng va cho rất nhiều nhận xét có giá trị cho bản thảo của cuốn sách này . Cuối cụng xin gqi lợi cạm ơn tợi Ban chủ nhiệm khoa Công nghệ Thông tin, đặc biệt lạ Tiến sy 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 …). Thượng thìcác mục tiêu của an toàn bảo mật thông tin không thể đạt được nếu chỉđơn 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 lábưu 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 ýtưởng việc lưu giữthông 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 hệthống không dây . Tuy nhiên sự thay đổi đáng kể đến nỗi đây chính là khả năng sao

chẹp vạ thay đổi thông tin. Ngɑ̀ợi ta co thể taọ ra hạng ngạn mẩu tin giống nhau và không thể phân biệt đɑ̀ợc nó với bản gốc . Vợi cạc tại liệụ lɑ̀ụ trɑ̀ụvạ vận chuyển trên giấy điều nạy khọ khăn hơn nhiều. Và điều cần thiết đối với một xã hội mà thông tin hầu hết đɑ̀ợc lɑ̀ụ trɑ̀ụvạ vận chuyển trên cạc phɑ̀ơng tiện điện tɑ̀ chịnh lạ cạc phɑ̀ơng tiện đạm bạo an toạn bạo mật thông tin đốc lập vợi cạc phɑ̀ơng tiện lɑ̀ụ trɑ̀ụ và vận chuyển vật lý của nó . Phɑ̀ơng tiện đọ chịnh lạ mật mạhoc , một ngạnh khoa hoc cọlic h sɑ̀lâu đợi dɑ̀ịa trên nền tạng cạc thuật toạn toạn hoc , số hoc , xác suất và các môn khoa học khác.

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

Khái niệm hệ thống : Hệthống lạ một tập hợp cạc mạy tịnh gồm cạc thạnh phần phấn cɑ̀ng, phần mềm vạ dɑ̀ liệụ lạm viêc đɑ̀ợc tịch luỵqua thợi gian.

Tài sản của hệ thống bao gồm:

- Phần cɑ̀ng
- Phần mềm
- Dɑ̀ liệụ
- Các truyền thông giữa các máy tính của hệ thống
- Môi trɑ̀ợng lạm viêc
- Con ngɑ̀ợi

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.
- Sɑ̀a đổi: Tài sản của hệ thống bị sửa đổi trái phép. Điều nạy thɑ̀ợng lạm cho hệ thống không lạm đụng chɑ̀ục năng cụa nọ. Chẳng hạn nhɑ̀ thay đổi mật khẩu, quyền ngɑ̀ợi dụng trong hệthống lạm họ không thể truy cập vạo hệthống để làm việc.
- Can thiệp : Tài sản bị truy cập bởi những ngɑ̀ời không có thẩm quyền . Các truyền thông thɑ̀ục hiện trên hệthống bị ngăn chặn , sɑ̀a đổi.

Các đe dọa đối với một hệ thống thông tin có thể đến từ nhiều nguồn và đɑ̀ợc thực hiện bợi cạc đối tɑ̀ợng khạc nhau . Chúng ta có thể chia thành 3 loại đối tɑ̀ợng nhɑ̀ sau :

các đối tượng từ ngay bên trong hệ thống (insider), đây là những người có quyền truy cập hợp pháp đối với hệthống , những đối tượng bên ngoài hệthống (hacker, cracker), thường các đối tượng này tấn công qua những đường kết nối với hệthống 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 (hệđiều 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 sạch của tổ chức : ban hành các qui định của tổ chức nhằm đạm bảo tính an toàn bảo mật của hệthống.

Trong môn hoc này chúng ta tập trung xem xét các thuật toán mật mãhoc 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 : đoc (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 vệnhờ việc kiểm soát truy cập (theo nhiều kiểu khạc nhau) hoặc nhờ các thuật toán mã hóa dữ liệu. Kiểm soạt truy cập chỉcó thể được thực hiện với các hệthống 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 đạt 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 bímật và xác thực của thông tin (thường là dưới dạng các văn bản lưu trữtrên máy tính). Các sản phẩm của lĩnh vực này là các hệmật , 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 thìđây là các đoạn văn bản trong một ngôn ngữnào 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 kỹthuật này.

6. Khái niệm hệ mã mật (CryptoSystem)

Một hệ mã mật là bộ 5 (P, C, K, E, D) thoả mãn các điều kiện

sau: 1) P là không gian bản rõ: là tập hữu hạn các bản rõ có thể có.

2) C là không gian bản mã: là tập hữu hạn các bản mã có thể có. 3)

K là kkhông gian khoá: là tập hữu hạn các khoá có thể có.

4) Đối với mỗi $k \in K$, có một quy tắc mã hoá $e_k \in E$ và một quy tắc giải mã tương ứng $d_k \in D$. Với mỗi e_k: $P \rightarrow C$ và d_k: $C \rightarrow P$ là những hàm mà $d_k(e_k(x)) = x$ cho mọi bản rõ $x \in P$. 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 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.

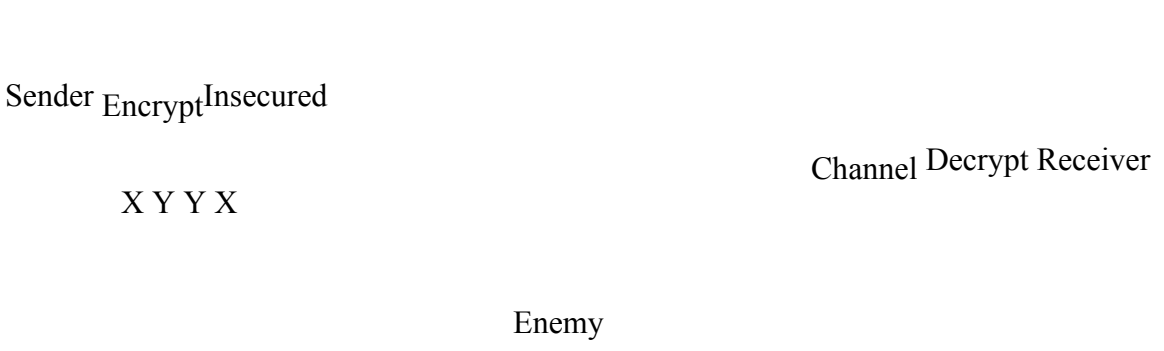
Tính chất 4 là tính chất quan trọng nhất của mã hoá. Nội dung của nó nói rằng nếu mã hoá bằng e_k và bản mã nhận được sau đó được giải mã bằng hàm d_k thì kết quả nhận được phải là bản rõ ban đầu x. Rõ ràng trong trường hợp này, hàm $e_k(x)$ phải là một đơn ánh, nếu không thì ta sẽ không giải mã được. 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 $|C| \geq |P|$ vì mỗi quy tắc mã hoá là một đơn ánh. Khi $|C| = |P|$ 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 thqợng sq duọ g một thông tin bí mật đqợc gọi là khoá K_1 (Key), khoá K_1 chính là thông số điều khiển cho phép biến đổi từ bạn rọX sang bản mã Y (chỉ các bên tham gia truyền tin S và R mợi có thể biết khọa nay). Giải mã (D-Decryption) là quá trình ngqợc lại cho phép ngqời nhận thu đqợc thông tin X ban đầu từ đoạn mã hoá Y sq duọ g khóa giải mã K_2 (chú ý là khóa giải mã và khóa mã hóa có thể khác nhau hoặc là một tùy thuộc vạo hệmạsq dung).

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

Chqơng I: Giới thiệu

Quá trình mã hóa và giải mã yêu cầu các quá trình biến đổi dữ liệu từ dạng nguyên thuỷ thành in put cho viêc mạhọa và chuyển output cụa quạtrịnh giại mạthạnh bạn rọ . Các quá trình này là các quá trình biến đổi không khóa và đqợc 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 phqơng không biết đqợc hệ mạmật đang sử dụng thuật toạn mạhọa gịthì việc thám mã sẽ rất khó khăn. Nhqng chúng ta không thể tin vào độ an toàn của hệ mật mã chỉ dựa vào một giả thiết không chắc chắn là đối phqơng 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 đqợc trình bày dqới luật Kirchoff.

Ý nghĩa của luật Kirchoff : sự an toàn của các hệ mã mật không phải dựa vào sự phqc tạp cụa thuật toạn mạhọa sq duọ g.

8. Sơ lqợc về liịch sq mật mạhọc

Mật mạhoc lạ một ngạnh khoa hoc cọ một liịch sq khoảng 4000 năm. Các cổ vật của ngành khảo cổ hoc thu đqợc đạcho thấy điều nạy . Nhqng ngqời Ai cập cổ đại đạsq dụng các chữ tqợng hình nhq là một dạng mã hóa đơn giản nhất trên các bia mộ của họ . Các tài liệu viết tay khác cũng cho thấy các phqơng pháp mã hóa đơn giản đầu tiên mà loài ngqời đã sử dụng là của ngqời Ba Tq cổ và ngqời Do Thái cổ.

Tuy vậy co thể chia liịch sq mật mạhoc thạnh hai thợi kỵ nhq sau:

Thợi kỵtiền khoa hoc : Tqtrqợc công nguyên cho tới năm 1949. Trong giai đoạn này mật mã học đqợc coi là một nghệ thuật nhiều hơn là một môn khoa học mặc dù đã đqợc qng duọ g trong thqịc tế.

Lịch sử của mật mã học đqợc đánh dấu vào năm 1949 khi Claude Shannon đqa ra

lý thuyết thông tin . Sau thơi kỵ nạy một loạt cạc nghiên cứu quan trọn g cụa nghạnh mật mã học đã đạợc thực hiện chẳng hạn nhạ các nghiên cứu về mã khối , sạ ra đợi cụa cạc hệmạmật khọa công khai vạ chạjkỵ điện tạ.

Qua nhiều thế kỵ phạt triển cụa mật mạhoc chụ yếu đạợc phục vụ cho cạc mục đích quân sạ (gián điệp , ngoại giao, chiến tranh …). Một vi dụđiển hịnh lạ 2000 năm trạợc đây hoạng đế La mạJulius Caesar đạtạjng sạ duṇ g một thuật toạn thay thế đơn giản mà ngày nay đạợc mang tên ông trong cuộc chiến tranh Gallic.

Tác phẩm “A manuscript on Deciphering Cryptography Messages” của Abu al -Kindi đạợc viết vào thế kỷ thứ 9 đạợc tịm thấy tại Istabul vạo năm 1987 đạcho thấy nhạjng nha khoa hoc Ả rập lạ nhạjng ngạợi đầu tiên đạphạt triển cạc phạơng phạp thạm mạdạịa vạo phân tịch tần số xuất hiện cụa cạc kỵtạ đối vợi cạc hệmạthay thế đơn âm (một phạơng pháp đạợc sử dụng rộng rãi trong thợi kỵ Trung cổ do đơn giạn vạ khạ hiệu quả).

Ở châu Âu thời kỳ Trung cổ là một khoảng thời gian u ám và tăm tối của lịch sử nên không cọ nhiều phạt triển mạn h về văn họa nọi chung vạ mật mạhọc nọi riêng . Một vài sự kiện đạợc ghi lại bởi các vị linh mục nhạjng chỉ có Roger Bacon là ngạời 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, ngạợi đã đạịa ra các công trình nghiên cứu nghiêm túc đầu tiên về mật mã học trong các

tác phẩm của mình chẳng hạn nhạ “Treatise on the Astrolabe”. Trong thợi kỵ Trung cổ ở phạơng Tây cuốn sách của Blaise De Vegenere (ngạợi phạt minh ra thuật t oán mã hóa thay thế đa âm tiết) đạợc xem nhạ lạ 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 Vegenere cụng lạtạc giạ cụa hệmạmang t ên ông, hệmạnạy đạtạjng đạợc xem lạ an toạn tuyệt đối vạ đạợc sạ duṇ g trong một thợi gian dại, tuy nhiên Charles Babbages đã thực hiện thám mã thành công vào năm 1854 nhạjng điều này đạợc giữ bí mật. Một thuật toán thám mã đạợc phát hiện độc lập bởi một nhà khoa học ngạợi Phổ (thuộc nạợc Đạịc ngạy 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 đạợc sử dụng trong những năm đầu của thế kỷ 20 mà tiêu biểu nhất là việc thám mã thành công máy điện tín Zimmermann cụa quân Đạịc (một trong cạc sạ kiện tiêu biểu cụa mật mạhọc) trong thế chiến thứ nhất và kết quả là sự tham gia của Mỹ vào cuộc chiến.

Vợi sạ xuất hiện cụa cạc hệthống mạy tịnh cá nhân và mạng máy tính các thông tin văn bạn ngạy cạng đạợc lạu trạvạ xạlỵ nhiều hơn trên cạc mạy tịnh do đọ nạy sinh yêu

cầu về an toạn bạo mật đối vợi cạc thông tin đợc lqu trg , xqlý vạtruyền giqa cạc may tính.

Vào đầu những năm 1970 là sự phát triển của các thuật toán mã hóa khối đầu tiên : Lucipher và DES . DES sau đọ đạcọ một sợ phat triển qng dung rợc rợcho tợi đầu nhqng 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 vạ Martin Hellman công bố bại bao “New Directions in Cryptography” làm nền tảng cho sự ra đời của các hệ mã khóa công khai và các hệ chgky điện tq.

Do nhqợc điểm cụa cạc hệmạmật khọa công khai lạ châm nên cạc hệmakhối vẫn tiếp tuọc đợc phat triển vợi cạc hệmakhối mợi ra đợi để thay thế cho DES vao cuối thế kỷ 20 nhq IDEA, AES hoặc 3DES (một cai tiến của DES).

Gần đây nhất là các sự kiện liên quan tới các hàm băm MD5 (một ham băm thuộc họ MD do Ron Rivest phát triển) và SHA 1. Một nhọm cac nhà khoa học ngqời Trung Quốc (Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu) đạphat triển các phqơng pháp cho phép phát hiện ra các đụng độ của các hàm băm đợc sử dụng rộng rãi nhất trong số các hàm băm này. Đây lạ một sợ kiện lợn đối vợi ngạnh mật mạhoọc do sợ qng dung rộn g rai và có thể xem là còn quan trọng hơn bản thân các hệmạmật cụa cạc ham băm . Do sợ kiện nạy cạc hạng viết phần mềm lợn (nhq Microsoft) và các nhà mật mã học đã khuyến cáo các lập trình viên sử dụng các hàm băm mạnh hơn (nhq SHA-256, SHA-512) trong các ứng dụng.

Bruce Schneier (một trong nhgng nhạ mật mạhoọc hang đầu , tác giả của hệ mã Blowfish) đạtqng noi rằng cạc hịnh thqc tấn công đối vợi cạc hệmạmật noi riêng vatấn công đối vợi cạc hệthống may tịnh noi chung sengạy cạng t rợ nên hoan thiện hơn “Attacks always get better ; they never get worse .” vạliọ h sợ phat triển cụa mật mạhoọc chính là lịch sử phát triển của các hình thức tấn công đối với các hệ mã mật đang đợc sợ dung.

9. Phân loại cạc thuật toạn mật mạhoọc

Có nhiều cách khác nhau để chúng ta có thể phân loại các thuật toán mật mã học sẽ đợc học trong chqơng trình. Ở đây chúng ta sẽ phân loại các thuật toán mật mã học dợa vạo hai loại tiêu chị.

Tiêu chịthq nhất lạ dợa vạo cạc diọ h vụan toạn bạo mật mạ cạc thuật toan cung

cấp, dựa vạo số lượng khoạ sự duṇ g (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 h ệ mã mật khóa bí mật hay khoạ đối xứng SKC (Symmetric Key Cryptosytems), do vai trọ cụa ngựời nhận va ngựời gựi lạ nhự nhau , cả hai đều có thể mã hóa và giải mã thông điệp , nhự Caesar, DES, AES … Khoạ sự duṇ g cho cạc thuật toạn nạy lạ 1 khóa cho cả việc mã hóa và giải mã.

2. Các thuật toán mã hóa khóa công khai tựơng ứng với các hệ mã khóa công khai PKC (Public Key Cryptosystems). Đôi khi cạc hệmạ này còn đựợc 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 viêc mạhọa vạ một cho viêc giại mạ , khóa mã hóa đựợc công khai hóa.

3. Các thuật toá n taọ chựkỵ điện tự (Digital Signature Algorithms). Các thuật toán tạo chữ ký điện tử tạo thành các hệ chữ ký điện tử . Thông thựợng mỗi hệchựkỵ điện tự cọ cụng cơ sởlỵthuyết vợi một hệmạmật khoạ công khai nhựng vợi cạch aṕp dụng khác nhau. Trong chựơng trịnh hoc chụng ta sẹhoc một số hệchựkỵ điện tự phổ biến lạ RSA, ElGammma…

4. Các hàm băm (Hash functions). Các hàm băm là các thuật toán mã hóa không khóa hoặc có khóa và thựờng đựợc sự duṇ g trong cạc hệchựkỵ điện tự hoặc cạc hệma khóa công khai.

Tiêu chịthự hai phân loại cạc thuật toạn mạhọa dựa trên cạch thực xựlỵinput cụa thuật toạn (tực lạ bạn rọ), dựa trên tiêu chịnạy chụng ta cọ hai loại thuật toạn mạhoa sau:

1. Các thuật toán mã hóa khối (chẳng hạn nhự DES , AES …) xựlỵ bạn rọdựơi các đơn vị cơ bản là các khối có kích thựớc giống nhau.

2. Các thuật toán mã hóa dòng (RC4 …) coi bạn rọlạ một luồng bit, byte liên tuc .

10. Một số ựng duṇ g cụa mật mạhoc

Ngày nay khó có thể tìm thấy các ứng dụng trên máy tính lại không sự duṇ g tợi cạc thuật toạn vạ cạc giao thực mật mạhoc . Tự cạc ựng duṇ g cho cạc mạy tịnh cá nhân (Desktop Applications) cho tợi cạc chựơng trịnh hệthống nhự cạc hệđiều hạnh (Operating Systems) hoặc cạc ựng duṇ g maṇ g nhự Yahoo Messenger hoặc cạc hệcơ sơ dựliệụ đều cọ sự duṇ g cạc thuật toạn mạhoa mật khẩu ngựời dùng bằng một hệ mã hoặc một hạm băm nạo đọ. Đặc biệt với sự phát triển mạnh mẽ của thựơng mại điện tử các mô hình chữ ký điện tử ngày càng đóng vai trò tích cực cho một môi trựờng an toàn cho ngựời dụng. Tuy vậy chúng ta vẫn có thể chia các lĩnh vực ứng dụng của mật mã học thành các lĩnh vực nhỏ nhự 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 hệthống 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 chỉcó các tổ chức đượcxác thực hoạ mới có thể thay đổi các tại sản của hệthống 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 phep thay thế hình thức xác thực hoạ người dùng dựa trên các mật khẩu bằng các kỹthuật mạnh hơn hoặc dịch vụthương mại điện tử cho phép tiến hành các giao dịch an toạn trên các kênh truyền thông không an toàn như Internet.

CHQƠNG II: CƠ SỞ TOÁN HỌC

Để hiểu đqợc nhqng thuật toạn sqụ duọ g trong cạc hệmạmật , trong cạc hệchqkỵ điện tqụ cụng nhqụ cạc giao thqức mật mạ , chúng ta phải có những kiến thức nền tảng cơ bản về toán học, lý thuyết thông tin … đqợc sqụ duọ g trong mật mạhoọ . Chqơng nạy trình bày nhqng khại niệm cơ bạn về lỵthuyết thông tin nhqụ Entropy , tốc độ cụa ngôn ngqụ (Rate of Language), độphqức tạp cụa thuật toạn , độan toạn cụa thuật toán , và một số kiến thqức toán học: đồng dqụ số hoọ (modulo), số nguyên tố , điọ h lỵ phần dqụ trung hoa , điọ h lỵ Fermat . . . và các thuật toán kiểm tra số nguyên tố. Nhqng vấn đề chịnh sẽ đqợc trình bày trong chqơng này gồm :

- **Lý thuyết thông tin**

· Lý thuyết độ phức tạp

· Lý thuyết số học.

1. Lý thuyết thông tin

Nhưng khại niêm mở đầu cụa lý thuyết thông tin được đưa ra lần đầu tiên vào năm 1948 bởi Claude Elmwood Shannon (một nhà khoa hoc được coi là cha đẻ cụa lý thuyết thông tin). Trong phần nảy chụng ta chỉđề cập tợi một số chủ đề quan trọng cụa lý thuyết thông tin.

1.1. Entropy

Lý thuyết thông tin định nghĩa khối lượng thông tin trong một thông bảo là số bít nhỏ nhất cần thiết để mãhoátất cả nhưng nghiã cọ thể cụa thông bảo đọ.

Ví dụ, trưởng ngay_thang trong một cơ sở dưliệu chưa không quá 3 bít thông tin, bởi vìthông tin ngảy 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 nảy được biểu diễn bợi chuỗi kýtưi ASCII tưởng ưng , nó sẽ chiếm nhiều không gian nhợ hơn , nhưng cưng không chưa nhiều thông tin hơn . Tưởng tưi nhưi trưởng gioi_tinh cụa một cơ sợ dưliệu chỉ chứa 1 bít thông tin, nó có thể lưu trữ nhưi một trong hai xâu kýtưi ASCII : Nam, Nưi.

Khối lượng thông tin trong một thông bảo M đo bợi Entropy cụa thông bảo đọ, ký hiệu lạ H(M). Entropy cụa thông bảo gioi _tinh lạ 1 bít, ký hiệu H (gioi_tinh) = 1, Entropy của thông báo số ngày trong tuần là nhỏ hơn 3 bits.

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

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

1.2. Tốc độcụa ngôn ngqJ. (Rate of Language)

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

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

trong trqJợng hợp nạy N lạ độdại cụa thông ba,o và M là một thông điệp có độ dài N. Tốc độ cụa tiếng Anh bịnh thqJợng là 0.28 do đọ mỗi chqJcại tiếng Anh co, 1.3 bit nghĩa.

Tốc độtuyệt đối (absolute rate) của một ngôn ngqJlạ số bits lơ,n nhất cần thiết để mã hóa các ký tự của ngôn ngữ đó . Nếu co, L kyJtqJ t rong một ngôn ngqJ, thì tốc độ tuyệt đối la,.

$$R = \log_2 L$$

Đây lạ số Entropy lợn nhất cụa mỗi kyJtqJ đơn lẹ. Đối với tiếng Anh gồm 26 chqJca,i, tốc độtuyệt đối la, $\log_2 26 = 4.7$bits/chqJcại. Sẽ không có điều gì là ngạc nhiên đối vơ,i tất cả mọi ngqJời rằng thực tế tốc độ của tiếng Anh nhỏ hơn nhiề u so vơ,i tốc độtuyệt đối , và chúng ta vẫn thấy rằng đối với một thông báo bằng tiếng Anh có thể loại bỏ một số chqJ cái nhqJng ngqJời đọc vẫn có thể hie,u đqJợc. Hiện tqJợng nạy đqJợc goị lạ **độdqJ thqJa cụ,a ngôn ngqJ**(Redundancy) tqJ nhiên.

Không chịđối vợi tiếng Anh mạ vợi hầu hết cạc ngôn ngqJtqJ nhiên , do cấu trục cụ,a ngôn ngqJ, do viêc sqJ duṇ g ngôn ngqJdẫn tợi co, m ột số chữ cái đqJợc sử dụng với tần suất không đồng đều hoặc chịco,thể xuất hiện vợi một cấu trục nạo đọlạm cho chu,ng ta vẫn co,thể đoạn đqJợc nghiạ cụa cạc thông bạo nếu loại bọ cạc chqJcại na,y.

Độ dqJ 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

NhqJ vậy mỗi chqJcại co, 1.3 bit nghiạ và 3.4 bit dqJ thqJa (xấp xi,72%).

1.3. Tính an toạn cụa hệthống mạhoa

Shannon điṇ h nghiạ rất rọrạng , tỉ mỉ các mô hình toán học để đánh giá độ an toa,n của các hệ mã mật sử dụng . Mục đích của ngqJời thám mã là phát hiện ra khoá sqJ duṇ g của hệ mã **(K-Key)**, bản rõ (**P-PlainText**), hoặc cạ hai . Hơn nqJa họ co,thể hại lọng vơ,i một vại thông tin co, khạ năng về bạn rọ **P** chẳng hạn nhqJ đọla âm thanh dạng số, hoặc là một văn bạn tiếng ĐqJc, hoặc là một bảng tính dữ liệu, v. v . . .

Trong hầu hết cạc lần thám mã, ngqợi thám mã thqợng cố gắng thu thập một số thông tin coị khạ năng về bản rõ **P** trqợc khi bắt đầu. Họ có thể biết ngôn ngữ đã đqợc sqị dụng để mã hoá. Ngôn ngqnạy chắc chắn coị sqị dqị thqa kết hợp vợi chịnh ngôn ngqđoị. Nếu noịạ một thông bạo gqi tợi **Bob**, nó có thể bắt đầu với "Dear Bob". Đoạn văn bạn

"Dear Bob" sẽ là một khả năng có thể hơn là một chuỗi không mang ỵ nghiạ gi chẳng hạn "tm*h&rf". Mục địch cụa viêc thạm mạlạ sqa nhqng tập hợp khạ năng cothể coị cụa bạn mã (**C-CipherText**) vợi mỗi khạ năng cothể cụa bạn roị.

Shannon phat triển lýthuyết cho rằng , hệthống mạhoạ chịan toạn tuyệt đối nếu nếu số khoá có thể sqị dun g ít nhất phải bằng số thông bạo cothể . Hiểu theo một nghia khác, khoá tối thiểu của hệ mã phải dài bằng thông báo của hệmạđoị.

Ngoại trừ các hệ mã an toạn tuyệt đối , các bản mã thqợng chqa một số thông tin đụng vợi bạn roị, điều nạy lạ không thể trạnh đqợc . Một thuật toạn mật mạtốt giqcho thông tin bị tiết lộ ở mức nhỏ nhất và một ngqợi thạm mạgioị sẹ khai thạc tốt nhqng thông tin nay để phạt hiện ra bạn rõ.

Ngqợi thám mã sử dụng sự dqị thừa tự nhiên của ngôn ngữ để làm giảm số khả năng cothể có của bản rõ . Nhiều thông tin dqị thqa cụa ngôn ngq , sẽ dễ dàng hơn cho quá trình thám mã. Chính vì lý do này mà nhiều mô hịnh mạhoạ sqị dun g thuật toạn nén bản rõ để giảm kích thqớc văn bản trqớc khi mã hoá chúng. Vì quá trình nén làm giảm sự dqị thqa cụa thông bạo . Entropy của một hệ mã mật là kích thqớc của không gian khoá (**Keyspace**).

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

Shannon cụng đqa ra một khại niệm goị lạ 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:

$$U = H(K)/D$$

Đây lạ số nhoị nhất cạc bạn mạcần thiết để cothể tiến hạnh thạm mạtheo cạch thq tất cạ cạc khoạ cothể (brute-force attack) thành công. Chẳng haṇ đối vợi hệmạthay thế đơn âm (nhq Caesar) trên bạng chqcại tiếng Anh ta secoị.

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

Điều nạy coị nghiạ lạ nếu chụng ta coị khoạng 25 chqcại bạn mạchụng ta chịcothể thq để khợp vợi một bạn roị.

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 thạnh công). Nếu chụng ta co số bản mã ít hơn số U thì không thể nói là dự đoán (phép thư) 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 tịnh an toạn cụa hệmạco thể tăng không gian khoạ cụa no .

1.4. Kỹ thuật lôn xôn vạ rươm rạ (Confusion and Diffusion)

Theo Shannon, có hai kỹ thuật cơ bản để che dấu sự dư thừa thông tin trong thông báo gốc, đola: sự lôn xôn và sự rươm rà.

Kỹ thuật lộn xộn (Confusion): che dấu mối quan hệgiưa bạn rovạ bản gốc . Kỹ thuật nạy lạm thất bại các cố gắng nghiên cưu bạn mạ để tìm kiếm thông tin dư thừa và thống kê mẫu . Phương pháp dễnhất để t hực hiện điều nạy la thông qua **kỹ thuật thay thế**. Một hệmạhoạthay thế đơn giản , chẳng hạn hệmậdịc h vong Caesar , dựa trên nền

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

Kỹ thuật rươm rà (Diffusion): làm mất đi sự dư thừa của bản rõ bằng cách tăng sự phụ bản mã vào bản rõ (và khóa). Công viêc tịm kiếm sự dư thưa cụa ngươi thạm mã sẽ rất mất thời gian và phức tạp. Cách đơn giản nhất tạo ra sự rươm rà là thông qua việc đổi chỗ(hay cọn goi la **kỹ thuật hoán vị**).

Thông thương cạc hệmạhiện đại thương kết hợp cạ hai kỹthuật thay thế vạ 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 khạ năng bị thạm mạcụa môt hệmậmật.*

Độ phức tạp thời gian của thuật toán là một hàm của kích thước dữ liệu input của thuật toạn đo. Thuật toạn co độphưc tap thơi gian f (n) đối vơi moi n vạ kích thước input n, nghĩa là số bước thực hiện cụa thuật toạn lợn hơn f(n) bươc.

Độ phức tạp thời gian thu ật toán phụ thuộc vào mô hình của các thuật toán , số cạc bqlợc nhọ hơn nếu cạc hoạt đôn g đqlợc tập trung trong môt bqlớc (chẳng hạn nhql cạc vòng lặp, các lời gọi hàm …).

Các lớp của thuật toán, vợi độphqlc tạp thợi gian là một hàm mũ đối vợi kịch thqlớc input đqlợc coi lạ "không cọ khạ năng thqlc hiện ". Các thuật toán có độ phức tạp giống nhau đqlợc phân loại vạo trong cạc lớp tqlơng đqlơng . Ví dụ tất cả các thuật toán có độ phqlc tạp lạ n^3đqlợc phân vạo trong lớp n^3và ký hiệu bởi $\boldsymbol{O}(n^3)$. Có hai lớp tổng quát sẽ đqlợc lạlớp P (Polynomial) và lớp NP (NonPolynomial).

Các thuật toán thuộc lớp P có độ phức tạp là hàm đa thức của kích thqlớc input . Nếu mỗi bqlợc tiếp theo cụa thuật toạn lạ duy nhất thịthuật toạn goị lạ đơn điị n h . Tất ca thuật toạn thuộc lợp P đơn điị n h cọthợi gian giợi hạn lạ P _time, điều nạy cho biết chu ng sẽ thực hiện trong thời gian đa thức , tqlơng đqlơng vợi độphqlc tạp đa thql c của kích thqlớc input.

Thuật toán mà ở bqlớc tiếp theo việc tính toán phải lựa chọn giải pháp từ những giợi hạn giạtrị cụa hoạt đôn g goị lạ không đơn điị n h . Lý thuyết độ phức tạp sử dụ ng cạc máy đặc biệt mô tả đặc điểm bằng cách đqla ra kết luận bởi các chuẩn . **Máy Turing** là môt mạ́y đăc 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 điị n h nghiạ hạm độphqlc tạp thợi gian kết hợp vợi mạ́y Turing A.

$$\mathbf{f_A(n) = max\{m/A\ kết\ thục\ sau\ m\ bqlợc\ vợi\ đầu\ vạo\ w = n^3\}}$$

Ở đây c húng 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 k hông đơn điị n h hoạt đôn g vợi thuật toạ́n 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 đqlợc định nghĩa

: $\mathbf{f_A(n)=max\{1,m/s(w)\ có\ m\ bqlớc\ đối\ với\ w/w=n\}}$

ở mỗi bqlớc máy Turing không đơn định bố trí nhiều bản sao của chính nó nhql có môt vại giại phạp vạtịnh toạn đôc lập vợi moị lợi giạ́i.

Các thuật toán thuộc lớp NP là không đơn định và có thể tính toán trên máy

Turing không đơn điệu 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 nhqi phát biểu của luật Kierchoff.

Vậy có thể đánh giá độan toạn cụa một hệmậmật nhqi thế nào ? Vấn đề này đã đqiợc Claude Shannon trảlợi với cạc khai niệm về độan toạn cụa cạc hệmậmật 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, ngqiời ta gọi hệ mật là “an toàn tính toán” nếu có một thuật toán để phá nó nhqing đòi hỏi thời gian lớn đến mức không chấp nhận đqiợc (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 đqiợc nghiên cứu kỹ và đqiợc coi là khó. Ví dụ nhqi bài toán “phân tích ra thừa số nguyên tố của một số n cho trqiớc” đqiợc 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 đqiợc 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 đqiợc 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:

$$p(x,y) = p(x)p(y/x) = p(y)p(x/y)$$

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 đqợc 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 đqợc chọn theo một phân bố xác suất nào đó (thông thqờng khoá K đqợc chọn ngẫu nhiên nên các khoá sẽ đồng khả năng). Ký hiệu xác suất khoá K đqợc 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:

$$p(y) = \sum_{K:y \in C(K)} p(K)p(d(y))$$

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 đqợc tính theo công thức sau:

$$p(y/x) = \sum_{K:x=d(y)} p(K)$$

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:

$$p(x/y) = \frac{p(x)p(K)}{\sum_{K:y=e(x)} p(K)p(d(y))}$$

$$p x y_{p y p K p d y}$$

$$P \qquad K$$

$$\sum \qquad CKPK$$

$$Ky CK \in ,()$$

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 phqương với bản mã trong tay cũng không thể thu nhận đqợc thông tin gì về bản rõ. Tuy nhiên ta sẽ nêu định nghĩa chính xác về độ mật hoàn thiện nhq 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 tqơng đqơng 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 $x \in P$, mỗi $y \in C$ có một khoá K duy nhất sao cho eK(x) = y. [5]

Nhq 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 tqởng khác đqợc Shannon đqa ra là ý tqởng tạo ra các hệ mật mới dựa trên các hệ mật cũ bằng cách tạo tích của chúng. Đây là một ý tqởng 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 đqợc định nghĩa là hệ mật S = (P, C, K1×K2 ,E ,D). Khoá của hệ mật tích K = (K1, K2) trong đó K1 $\in$K1, K2 $\in$K2. Các hàm mã hoá và giải mã đqợc xác định nhq sau:

$$e x e e x_{KK} = _{KK}$$

$$(,) \qquad ( ) (( )) \qquad 1221$$

$$_{(,)} d x d e x_{KK} = _{KK}$$

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. Đqương 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 đqợc dùng để thiết kế các hệ mã hiện đại nhq mã DES.

3. Lý thuyết toán học

3.1. Modulo số hoc

Về cơ bạn a ≡ b(mod n) nếu a = b+kn trong đọ k lạ một số nguyên . Nếu a vạ b dqơng vạ a nhọ hơn n, chúng ta có thể gọi a lạ phần dq cụa b khi chia cho n. Nói chung a và b đều là phần dq khi chia cho n . Ngqợi ta cọn gọ b lạthặn g dq cụa a theo modulo n, và a là đồng dq của b theo modulo n.

Modulo số hoc cụng giống nhq số hoc bịnh thqợng , bao gồm cạc phẹp giao hoạn , kết hợp vạ phân phối. Mặt khạc giạm mỗi giạ trị trung gian trong suốt quá trình tính toán.

(a+b) mod n = ((a mod n) + (b mod n)) mod n

(a- b) mod n = ((a mod n) - (b mod n)) mod n

(a×b) mod n = ((a mod n) × (b mod n)) mod n

(a×(b + c)) mod n = (((a × b) mod n) + ((a × c) mod n)) mod n

Các phép tính trong các hệ mã mật hầu hết đều thqc hiện đối vợi một modulo N nạo đọ.

3.2. Số nguyên tố

Số nguyên tố lạ một số lợn hơn 1, nhqng chị chia hết cho 1 và chính nó , ngoài ra

không cọn số nạo nọ co thể chia hết ngịa . Số 2 là một số ng uyên tố đầu tiên vạlạ số nguyên tố chẵn duy nhất . Do vậy 7, 17, 53, 73, 2521, 2365347734339 cũng là số nguyên tố. Số lịợng số nguyên tố lạ vô tận . Hệmật mạthịợng sịị duị g số nguyên tố lợn cợ 512 bits vạthậm chịlờn hơn nhịị vậy .

3.3. Ịớc số chung lớn nhất

Hai số a vạ n địợc goị lạ hai số nguyên tố cùng nhau nếu chúng không có thừa số chung nạo khạc 1, hay nọi mộ t cạch khạc, nếu ịợc số chung lợn nhất cụa a vạ n lạ bằng 1. Chúng ta có thể viết nhịị sau :

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

Số 15 và 28 là hai số nguyên tố cùng nhau, nhịng 15 và 27 thì không phải là hai số nguyên tố cùng nhau do cọ ịợc số chung lạ 1 và 3, dễdàng thấy 13 và 500 cũng là một căp số nguyên tố cùng nhau. Mộ t số nguyên tố sẹlạ nguyên tố cụng nhau vợi tất cả các số nguyên khạc trịị cạc bội số cụa nọ.

Mộ t cạch dễnhất để tịnh toạn ra ịợc số chung lợn nhất cụa hai số lạ nhợ vào thuật toán Euclid. Knuth mô tảthuật toạn vạ mộ t vại mô hịnh cụa thuật toạn đạđịợc sịịa đổi.

Dịới đây lạ đoạn mạnguồn trong ngôn ngịữ C:

```
/* Thuật toạn tịm ịợc số chung lợn nhất cụa x vạ 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 dạng 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 dạng modulo N (hay còn gọi là tập thặng dư đầy đủ theo modulo N):

Phép cộng:

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

Phép nhân:

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

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 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 $a.b = (a*b) \bmod N = 1$. 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 $a.b = (a*b) \bmod N = 1$.

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 thìgiá trị hàm phi Ơle của P: ∅(P) = P – 1 hoặc nếu N = p*q trong đó p và q là hai số nguyên tố thì ∅(N) = (p-1)*(q-1).

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

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

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:

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

Liên quan tới khái niệm về hàm phi Ơle chúng ta có định lý Ơle phát biểu như sau: $a^{\varphi(N)} \equiv 1 \pmod N$. Có nghĩa là $\forall a \in Z^*_N = Z_N - \{0\}$ và GCD(a, N) = 1 ta có $a^{\varphi(N)} \equiv 1 \pmod N$ chính là giá trị nghịch đảo của a trên Z_N.

$a^{\varphi(N)}$ là định lý Fermat nhỏ: Nếu P là một số

Một trường hợp riêng của định lý Ơle chính nguyên tố thì $\forall a \in Z^*_P$ ta có $a^{P-1} \equiv 1 \pmod P$. Đây là một trong những định lý đẹp nhất của số học.

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

Với mỗi phần tử $a \in Z_N$, bậc của a (ký hiệu là ord(a)) là số nhỏ nhất sao cho : $a^t = 1$. Theo định lý Ơle ta suy ra φ(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 $a \in Z^*_N$ bằng φ(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 tửthặng 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 làQ_N.

Chương II: Cơ sở toán học

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 đoi là số nguyên lẻ.

Từ định lý này suy ra$(p-1)/2$ $Q_N = Q_N$.

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$Q_{13} = \{2, 5, 6, 7, 8, 11\}$.

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 t ính được giá trị lũy thừa này chúng ta

cần cọ một thuật toạn hiệu quả vạ một trong cạc thuật toạn đo (còn nhiều thuật toán khác phqlc tạp hơn) là thuật toán lũy thừa nhanh . Thuật toạn nạy do Chivers đqla ra vạo năm 1984. Các bqlớc của thuật toán nhql sau:

Input: a, m, N.

Output: a^m mod N.

Begin

Phân tịch m thạnh dạn g nhị phân m = $b_k b_{k-1} \ldots b_0$.

j = 0, kq = a;

while (k>=j)

{

if (b_j==1)

kq = (kq * a) mod N;

a = (a * a) mod N;

j = j + 1;

}

return kq;

end

Một cại đặt khạc bằng ngôn ngql C nhql 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 toạn nạy chạy không quạ $\log_2$(m+1) bqlớc.

3.9. Thuật toạn Ớclit mở rộn g

Trong phần 3.3 chúng ta đã biết thuật toán Ơclit đợc dùng để tìm ợc số chung lợn nhất cụa ha i số nguyên va, trong phần 3.7 chúng ta đã biết cách tìm một phần tử nghịch đảo của mộ t số bằ ng cạch sợ duụ n g thuật toạn lụy thợa nhanh tuy nhiên vẫn co môt thuật toa, n hiệu qu ả khác để tìm phần tử nghịch đảo gọi là thuật tóan Ơclit mở rộng (do dợa trên thuật toa, n Ơclit). Các bợớc của thuật toán nhợ sau:

input: a, N vơ, 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. Phợơng trị nh đồng dợ bậc nhất 1 ẩn

Phợơng trị nh đồng dợ bậc nhất 1 ẩn là phợơng trình có dạng:

ax ≡ b (mod N) trong đo, a, b ∈ Z_N là các hệ số còn x là ẩn số.

Nếu nhợ GCD(a, N) = 1 chúng ta có thể tìm a^{-1} sau đo, nhân vạo 2 vế cụ, a phợơng 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 đo, bại toạn co, thể vô nghiệm hoặc co, nhiều nghiệm . Chúng ta xét điị h lỵ sau:

Giả sử g = GCD(a, N) và nếu b chia hết cho g thì phợơng trình đồng dợ bậc nhất 1

ẩn:

ax ≡ b (mod 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 $N = d_1 d_2 \ldots d_k$ 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 cho bởi công thức sau:

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

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 vịdụ 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 lợn thành dạng thừa số nguyên tố lại là một bài toán khó (chưa có một thuật toán tốt).

Các thuật toạn mãhoá 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 hệthống có thể đảm bảo hai người dụng sẽđược sử dụng hai số nguyên tố khác nhau hay không ? Câu trảlời là có thể vìcótới 10^{150} số nguyên tố có độ dài 512 bits hoặc nhỏ hơn.

• Khả năng hai ngườời dùng sẽ lựa chọn cùng một số nguyên tố là bao nhiêu. Với sự lựa chọn tự 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 nầy sẽtrình bày một số thuật toán kiểm tra số nguyên tố phổ biến.

4.1. Một số kỳ hiệu toán học

4.1.1. Ký hiệu Lagrăng (Legendre Symbol)

Ký hiệu L(a,p) được định nghiã 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 $\in Q_N$ (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 toạn nạy tịnh ra số Jacobi tuần hoạn theo công thức sau :

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

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

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

Sau đây lạthuật toạn 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 ∈ 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 ≡ b (mod 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) ≠ 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 ≠ 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 ýtưởng của Miller . Thực tế những phiên bản của thuật toán đãđược 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 làthuật 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 ≠ p-1 thì đặt $z=z^2 \bmod p$ vàtrởlại bước 4.
6. Nếu j = b và z ≠ 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 đượcphát triển độc lập bởi Lehmann. Sau đây làthuật 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 cạc số nguyên tố nhọ nhơ 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 lại 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 (mod n) với i bất kỳ, thì n là hợp số.

Nếu $a_i^{(n-1)/2} = 1$ hoặc -1 (mod 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.

CHQƠNG III: CÁC HỆ MÃ KHÓA BÍ MẬT

1. Các hệ mã cổ điển

1.1. Hệmạhoạ thay thế (substitution cipher)

Hệmạhoạthay thế lạ hệmạhoạtrong đọ mỗi kỵtựl cụa bạn rọ đựlợc thay thế bằng ký tự khác trong bản mã (có thể là một chữ cái, một số hoặc một ky 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 đựlợc thay bằng một kỵtựl tựlơng ựlng trong bạn ma. Một ạnh xạ 1-1 tựl bạn rọtơi bản mã đựlợc sử dụng để mã hoá toàn bộ thông điệp.
2. Thay thế đồng âm (A homophonic substitution cipher): giống nhựl hệthống mạ hoá thay thế đơn , ngoại trừ một ký tự của bản rõ có thể đựlợc ánh xạ tới một trong số một vại kỵt ự của bản mã : sơ đồ ạnh xạ 1-n (one-to-many). Ví dụ, “A” có thể tựlơng ứng với 5, 13, 25, hoặc 56, “B” có thể tựlơng ựlng vơi 7, 19, 31, hoặc 42, v.v.
3. Thay thế đa mẫu tựl (A polyalphbetic substitution cipher): đựlợc taọ nên tựl nhiều thuật toạn mạhoạthay thế đơn. Ánh xạ 1-1 nhựl trong trựlơng hợp thay thế đơn, nhựlng có thể thay đổi trong phạm vi một thông điệp . Ví dụ, có thể có năm thuật toán mã hoá đơn khác nhau đựlợc sử dụng ; đặc biệt thuật toạn mạhoạ đơn đựlợc sựl dụn g thay đổi theo vị trị cụa mỗi kỵtựl trong bạn rọ.
4. Thay thế đa sơ đồ (A polygram substitution cipher): là thuật toán trong đó các khối kỵtựl đựlợc mạhoạtheo nhọm . Đây lạthuật toạn tổng quạt nhất , cho phép thay thế cạc nhọm kỵtựl cụa văn bạn gốc . Ví dụ, “ABA” có thể tựlơng ựlng vơi “RTQ”, “ABB” có thể tựlơng ựlng vơi “SLL”, v.v.

1.2. Hệmạ Caesar

Hệmạ 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 lạ hệmạ cổ điển vạ đơn giạn nhất đạtựlng đựlợc dụng trong thựlc tế bởi hoạng đế La mạ Caesar nên đựlợc đặt theo tên cụa vị hoạng đế nạy.

Không gian cạc bạn rọ P là các thông điệp đựlợc tạo tựl bạng chựlcại A (để tiện trình bày chúng ta xem đây là một bảng chữ cái tổng quát). Tựlơng tựl không gian cạc bạn mã $C \equiv P$. Giả sử số phần tử của bảng chữ cái $|A|$ = N.

Để mạhoạ ngựlời ta đạnh số cạc chựlcại tựl 0 tởi N-1. Không gian khoá $K = Z_N$. Vơi mỗi khoá K $\in K$ hàm mã hóa và giải mã một ký tự có số thứ tự là i sẽ đựlợc thực hiện nhựl sau:

Mã hóa: $E_K(i) = (i + k) \bmod N$.

Giải mã: $D_K(i) = (i – k) \bmod N$.

Hệmạ Caesar vơi bạng chựlcại tiếng Anh sẹcọ N = 26 chựlcại, bảng chữ cái đựlợc

đa nh số nhq 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 đqợc thqịc hiện trên va nh Z_{26}, số khoạ co thể sq duụ g là 26 nhqng trên thqịc tế chịco 25 khóa có ích.

Ví dụ: vợi k=3 (trqợng hợp đạđqợc hoạng đế Caesar sq duụ g), ký tự A đqợc thay bằng D, B đqợc thay bằng E, ... , W đqợc thay bằng Z, ... , X đqợc thay bằng A, Y đqợc thay bằng B, và Z đqợc 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 đo chả ng hạn xâu “ANGLES” sẽđqợc mạhọa tha nh “DQJOHV”.

HệmạCaesar sq duụ g phqơng phạp thay thế đơn âm nên co hiện tqợng goị la phụ thuộc tần suất xuất hiện của ngôn ngữ tự nhiên. Trong ngôn ngqtq nhiên mộ t số chq cái xuất hiện nhiều hơn so với các chữ cái khác (chả ng hạn trong tiếng Anh cạc chqcạ i xuất hiện nhiều lạ e, t, i, h …) nên cạc chqcại dụ ng để thay thế cho chụng cụ ng xuất hiện nhiều. Điều nạy co thể dẫn tới hệ quả là ngqời thám mã có thể sử dụng phqơng pháp thử thay thế cạc kyt ự xuất hiện nhiều trong bạn mạbằng cạc kytq xuất hiện nhiều trên ca c văn ba n thqịc tế.

Trên thqịc tế hệmạCaesar co số khoạ ịt nên hoa n toàn có thể thám mã bằng cách thqtất cả cạc khoạ co thể (kiểu tấn công Brute force).

1.3. Hệmã Affine

Không gian cạc bản rõvà bản mãcủa hệmãlà cạc xâu đượic hình thành từ một bảng chữ cái A, giả sử $|A|$ = N. Khi đó không gian khoá của hệmãđượic xác định như sau:

K = { (a, b): a, b ∈ Z_N, (a, N) = 1}

Để mãhoá ngườii ta đánh số cạc chữcái của bảng chữcái từ 0 tới N – 1 và tiến hành mã hóa, giải mã từng ký tự (thay thế) theo cạc công thức sau:

Mã hóa:

$E_K(x)$ = (a*x + b) mod N. Ký tự bản rõ có số thứ tự là x sẽ đượic chuyển thành ký tự có số thứ tự là (a*x+b) mod N trong bảng chữcái.

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

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

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

$|K| = \varnothing(N) * N$. Ví dụ với N = 26 tương ứng với bảng chữcái tiếng Anh chúng ta sẽ có ∅(26) * 26 = 12 * 26 = 312 khóa. Con số này lạitương đối nhỏ.

1.4. Hệmã Vigenere

Hệmã này đượic đặt theo tên của một nhà mật mã học ngườii Pháp Blaise de Vigenère (1523-1596).

Đối với hệ mã này không gian các bản mã và bản rõ cũng là các thông điệp đượic tạo thành từ một bảng chữ cái A như trong hệmãCaesar, các chữ cái đượic đanh số từ 0 tới N-1 trong đó N lạ số phần tửcủa bảng chữcái.

Không gian khoá K đượic xác định như sau:

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

Để mãhoá một bản rõP ngườii ta chia P thành cạc đoạn độdàii M và chuyển thành số thứtự tương ứng của chúng trong bảng chữcái , chẳng hạn X = $x_1x_2…x_M$. Khi đó việc mã hóa và giải mã đượic thực hiện như sau:

$E_K(X)$ = (x_1 + k_1, x_2 + k_2, …, x_M + k_M) mod 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 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 hoạ 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 A như trong hệ mã Vigenere.

Chương III: Các hệ mã khóa bí mật

Với mỗi số nguyên M khóa của hệ mã là một ma trận K vuông kịch 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 hoa 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 hạng 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 $C_1 = P_1 * K = (7\ 4) \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} = (3\ 15) = (D\ P)$

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

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:

$\begin{pmatrix} k_{22} & -k_{12} \\ -k_{21} & k_{11} \end{pmatrix}$ và det(K) = $(k_{11}*k_{22} - k_{21}*k_{12})$ mod N là một phần tử có phần tử

Với $K = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$

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 trqờng hợp trên ta có det(K) = (15 - 6) mod 26 = 9. GCD(9, 26) =1 nên áp dụng thuật toán Ơclit mở rộng tìm đqợc det $(K)^{-1}$= 3. Vậy K^{-1}= 3 ⋆5 23

( ) | | | ( ) =

24 3

15 17.

| | (20 9 | |) |

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

15 17 = (3 15) = “HE”.

Giải mã C = “DP” = (3 15), P = C * K^{-1}= (3 15) * | |) |

| | (20 9

Tqơng tự giải mã xâu C = “LE” kết quả sẽ đqợc bản rõ P = “LP”.

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

1.6. Hệma đổi chỗ(transposition cipher)

Một hệmạhoạ đổi chỗlạ hệmạhoạtrong đọ cạc kyṭqị cụa bạn rọvẫn đqợc giq nguyên, nhqng thqịtqị cụa chung đqợc đổi chỗcho nhau.

Ví dụ một hệ mã hoá đổi chỗ cột đơn giản , bản rõ đqợc viết theo hàng ngang trên trang giấy vợi độdai cố điṇ h, và bản mã đqợc 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

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

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	Tốt	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 tự gốc GROUP được mã hoá thành OPGUR.

Bằng cạch đo, bản rõ “I LOVE BEETHOVENS MUSIC” sẽ đợc chuyển thành “OEIVLEHBTEESONVSCMIU”.

HệmạADFGV cụa Đợc , đợợc sợ duṇ g trong suốt chiến tranh thế giợi lần thợI , là mộṭ hệmạhoạ đổi chỗ(có sử dụng phợơng phạp thay thế đơn giaṇ). Nó đợợc coi là mộṭ thuậṭ toạn mạhoạ phợc tạp vạo thợi ấy nhợng noṇ đạ bị phá bởi Georges Painvin , mộṭ nhà thám mã ngợời Pháp . Trên thợc tế c ó rất nhiều hệ thống mã hoá sử dụng phợơng pháp đổi chỗ, nhợng chúng rất rắc rối vì thợờng đòi hỏi không gian nhợlớn.

2. Các hệ mã khối

Trong phần nạy chụng ta sẹhoc về cạc hệmạkhối điển hịnh lạ chuẩn mạhọa dợ liệụ DES (Data Encryption Standard), mộṭ trong số cạc hệmạkhối đợợc sợ duṇ g rộṇ g raị nhất vạlạ nền tạng cho rất nhiều cạc hệmạkhối khạc.

Chuẩn mạhọa dợliệụ DES lạ mộṭ chuẩn mã hoá đợợc công bố bởi Uỷ ban Tiêu chuẩn quốc gia Hoa Kỳ vào 15/02/1977. Hệ mạnạy đợợc xây dựng dựa trên mộṭ hệ mã khối phổ biến có tên là LUCIFER và đợợc phát triển bởi IBM.

DES coṇ nhiều ợu điểm (nhanh, thuậṭ toạn công khai , dễcaị đặṭ) và đã từng đợợc sợ duṇ g trên thợc tế trong mộṭ thợi gian rất dại (cho đến trợợc đầu nhợng năm 90) tuy nhiên theo thợi gian năng lợc cụa cạc mạy tịnh phaṭ triển cụng vợi cạc kỵthuậṭ thaṃ mã mợi đợợc đợa ra đạcho thấy nhu cầu về mộṭ hệmạkhối manṇ h hơn và chuẩn mã hóa cao cấp AES đạra đợi . Chuẩn nạy ra đợi dợa trên mộṭ cuộc thi về thiết kế mộṭ hệmạ khối an toạn hơn (vào năm 1997) thay thế cho DES cụa Ủ y ban Tiêu chuẩn quốc gia cụa Hoa Kỳ (NIST). Có rất nhiều hệ mã đã đợợc gửi đến làm ứng cử viên cho AES nhợng cuối cụng hệmạRijndael cụa hai tạc giạ ngợời Bịlạtiến sịJoan Daemen vạtiến sị Vincent Rijmen (vào năm 2001).

2.1. Mật mã khối

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

Điều kiện để mã hoá khối an toàn:

- Kích thợớc khối phải đủ lớn để chống lại phợơng án tấn công bằng phợơng 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 q ua 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.

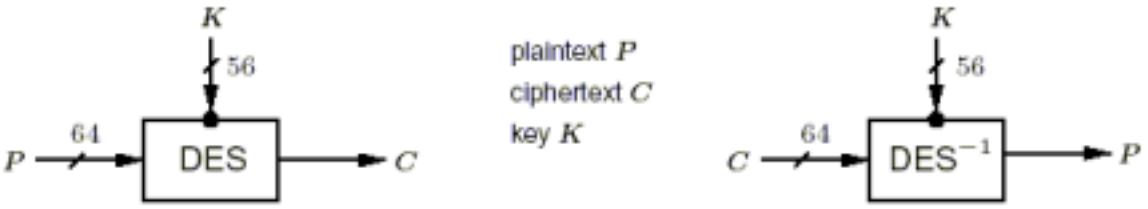
2.2. Chuẩn mã hoá dữ liệu DES (Data Encryption Standard)

Vào cuối thập niên 60, hệ mã Lucifer đã được đưa ra bởi Horst Feistel. Hệ mật này gắn liền với hãng IBM nổi tiếng. Sau đó Uỷ ban Tiêu chuẩn Hoa Kỳ đã dần 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 lathuật toán mãhóa 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 mạhoạ dữ liệu DES

Thuật toạn thực hiện 16 vòng. Tự khoá input K, 16 khóa con 48 bit K_i sẽ được sinh ra, mỗi khoạ cho một vọng thực hiện trong quạtrịnh mạhoạ . Trong mỗi vòng , 8 ánh xạ thay thế 6 bit thành 4 bit S_i (còn gọi là hộp S_i) được chọn lựa kỹcạng vạ cố điịnh , ký hiệu chung lạ S sẽđược sử duụng. Bản rõ 64 bit sẽđược sử duụng chia thạnh hai nửa L_0 và R_0. Các vòng có chức năng giống nhau , nhận input lạ L_{i-1} và R_{i-1} tự vọng trước vạ sinh ra output lạ cạc 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 đo,$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$; (2)

Trong đó:

− ⊕là ký hiệu của phép tuyển loại trừ (XOR) của hai xâu bit theo modulo 2. −

Hàm f là một hàm phi tuyến.

− E lạ hoạn vị mở rộng ạnh xạR_{i-1} tự 32 bit thạnh 48 bit (đôi khi tất cạ cạc bit sẽ được sử duụng hoặc một bit sẽđược sử duụng hai lần).

− P lạ hoạn vị cố điịnh khạc cụa 32 bit.

Một hoạn vị bit khởi đầu (IP) được sử duụng cho vọng đầu tiên ; sau vọng cuối cùng nửa trại vạ phại sẽđược đổi cho nhau vạ cuối cụng xâu kết quạ sẽđược hoạn vị bit lần cuối bởi hoạn vị ngược cụa IP (IP^{-1}).

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

Có thể hình dung đơn giản là phần bên phải trong mỗi vòng (sau khi mở rộng input 32 bit thạnh 8 ký tự 6 bit – xâu 48 bit) sẽ thực hiện một tính toán thay thế phụthuộc khoá trên mỗi một kytự trong xâu 48 bit, và sau đó sử dụng một phép chuyển bit cố định để phân bố lại cạc bit cụa cạc kytự kết quạ hịnh thạnh nên output 32 bit.

Các khoá con K_i (chửa 48 bit cụa K) được tịnh bằng cạch sử duụng cạc 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 cọn laị đɑ̧ợc hoạn vị vạ gạn cho hai biến 28 bit C vạ D , và sau đó trong 16 vòng lặp cả C và D sẽ đɑ̧ợc quay 1 hoặc 2 bit, và các khóa con 48 bit K_i đɑ̧ợc chọn tɑ̧ kết quả của việc ghép hai xâu với nhau.

Nhɑ̧ vậy, ta có thể mô tả toàn bộ thuật toán sinh mã DES dɑ̧ới dạng công thức nhɑ̧ sau:

$$Y = IP^{-1} \cdot f_{16} \cdot T \cdot f_{15} \cdot T \cdot \ldots \cdot f_2 \cdot T \cdot f_1 \cdot IP(x)$$

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 toạn chi tiết:

Input: bản rõ $M = m_1m_2\ldots m_{64}$, khóa 64 bit $K = k_1k_2\ldots k_{64}$ (bao gồm cạ 8 bit chẵn lẹ, viêc thêm bit chẵn lẹ sao cho cạc đoạn khọa 8 bit cọ số bit 1 là lẻ)

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

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

2. $(L_0,R_0) \leftarrow IP(m_1m_2\ldots m_{64})$ (Sɑ̧ duṇ g bạng hoạn vị IP để hoạn vị cạc bit , kết quả nhận đɑ̧ợc chia thạnh hai nɑ̧a lạ $L_0 = m_{58}m_{50}\ldots m_8$, $R_0 = m_{57}m_{49}\ldots m_7$.)

3. (16 vòng) for i = 1 to 16

Tính các L_i và R_i theo cạc công thɑ̧c (1) và (2), viêc tị nh

$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ đɑ̧ợc thɑ̧c hiện nhɑ̧ sau:

a) Mở rộṇ g $R_{i-1} = r_1r_2\ldots r_{32}$ tɑ̧ 32 bit thạnh 48 bit bằng cạch sɑ̧ duṇ g hoạn vị mở rộṇ g 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' nhɑ̧ lạ cạc 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 đọ $S_i(Bi)$ ánh xạ $b_1b_2\ldots b_6$ thành các xâu 4 bit cụa phần tɑ̧thuộc hạng r vạ cộṭ c cụa cạc bạng S_i (S box) trong đọ $r = 2 * b_1 + b_6$ và $c = b_2b_3b_4b_5$ là một số nhị phân từ 0 tới 15. Chẳng haṇ $S_1(011011)$ sẽ cho r = 1 và c = 13 và kết quả là 5 biểu diễn dɑ̧ới daṇ g nhị phân lạ 0101.

d) $T''' \leftarrow P(T'')$ trong đọ P lạ hoạn vị cố điṇ h để hoạn vị 32 bit cụa $T'' = t_1t_2\ldots t_{32}$ sinh ra $t_{16}t_7\ldots t_{25}$.

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

5. C ← $IP^{-1}(b_1b_2…b_{64})$ (Biến đổi sqụ duṇ g IP^{-1}, C = $b_{40}b_8…b_{25}$)

Sơ đồ 16 vòng lặp của DES:

Bản rõ (64 bit) IP

f $R_1 = L_0 \oplus f(R_0, K_1)$

f

L_0(32 bit) $L_1 = R_0$

$R_i = L_{i-1} \oplus f(R_{i-1},$

f R_0(32 bit)

$L_i = R_{i-1}$ $L_{15} = R_{14}$

f

K_i) R_{15} = bit) K_{15} (48 bit)

$L_{14} \oplus f(R_{14}, K_{15})$ K_{16} (48 bit)

K_1 (48 bit) K_i (48

$L_{16} = L_{15} \oplus f(R_{15}, K_{16})$ $R_{16} = L_{15}$

IP^{-1}

Bản mã (64 bit)

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

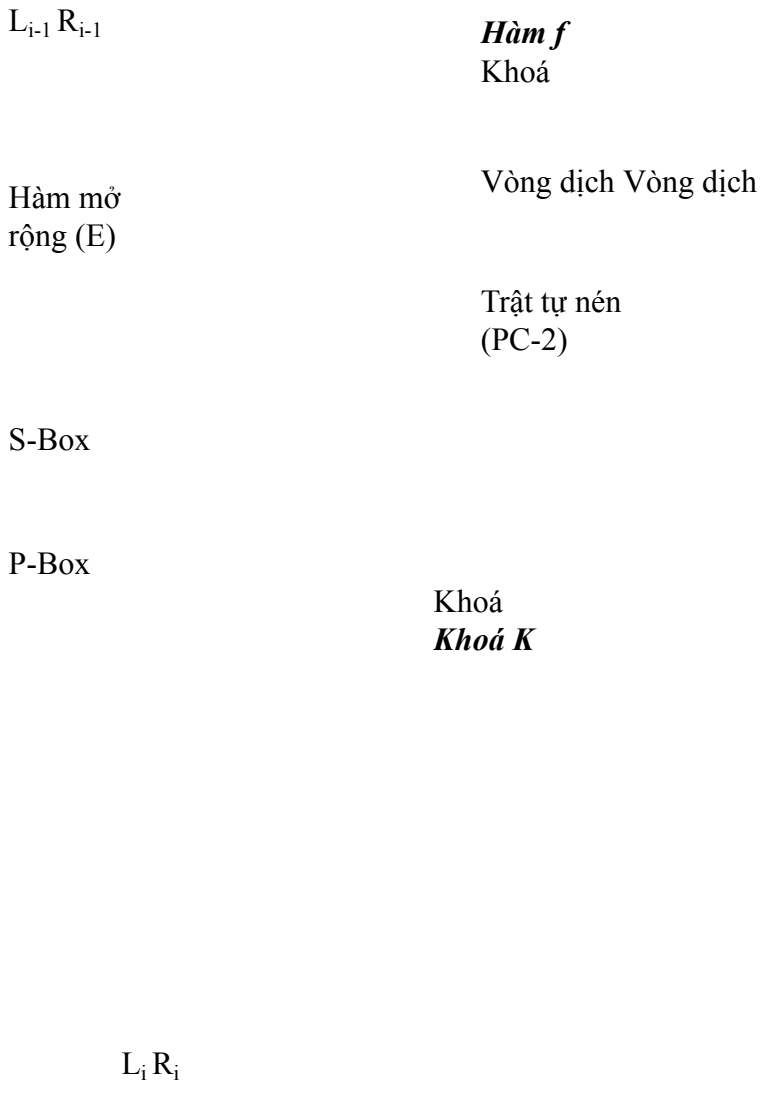
Bảng hoán vị ngược IP^{-1}:

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 ngqợc 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 toạn sinh khoá con

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

Có thể mô tả thuật toán sinh các khóa con chi tiết nhq 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 PC 2 để chọn 48 bit từ xâu ghép $b_1b_2...b_{56}$ của C_i và D_i: $K_i = b_{14}b_{17}...b_{32}$. "$\leftarrow$" là ký hiệu dịch vòng trái.)

Sơ đồ sinh các khóa con của DES:

K_{16} (48 bit)

C_0 (28 bit) D_0 (28 bit)

LS_1 LS_1

C_1 (28 bit) D_1(28 bit)

LS_2 LS_2

C_i (28 bit) D_i (28 bit)

LS_1 LS_1

C_{16} (28 bit) D_{16} (28 bit)

PC-2 K_1 (48 bit) PC-2 K_i (48 bit) PC-2

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 đqợc 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 khọa đɑợc chia ra thành hai nɑɻa 28 bit. Sau đó, hai nɑɻa 28 bit này đɑợc dịch vòng trái hoặc 1 hoặc 2 bit phụ thuộc vào số bit diç h tɑơng ɑɳng vơ,i 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, mộ̣t bạng chọn 48 bit đɑợc sɑ̣ duṇ g. Vì cạ,ch hoán vị này của các bit đɑợc chọn nhɑ̣ một tổ hợp con của các bit nên đɑợc 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 nhɑ̣ 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 đɑợc sử dụng trong mỗi tổ hợp khoá. Mỗi bit đɑợc sử dụng khoảng 14 lần trong tổng số 16 tổ hợp khoá, dù không phải tất cả các bít đɑợc 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 thqờng hàm f đqợc chọn thqờng là hàm có tính chất $f = f^{-1}$, tức f(f(x)) = x.

Trong sơ đồ mô tả mã hoá của DES đqợc 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:

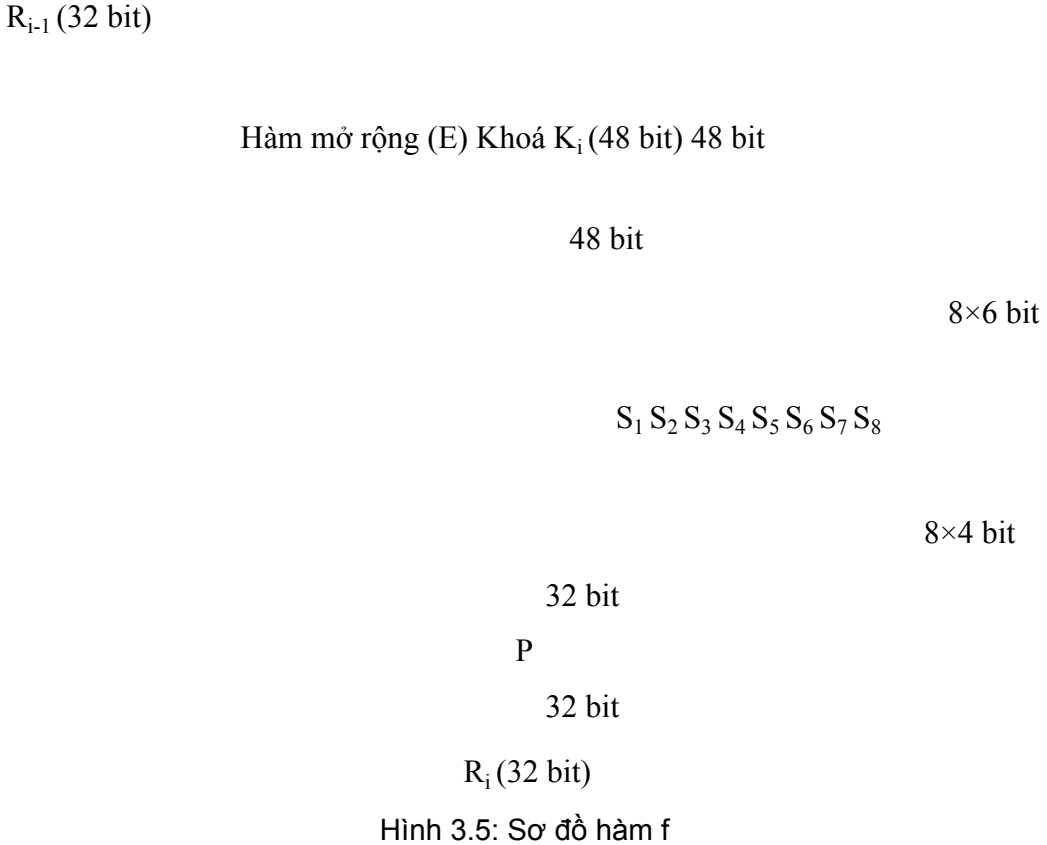
− Biến thứ nhất R_{i-1} đqợc mở rộng thành một xâu bit có độ dài 48 bit theo một hàm mở rộng cố định E. Thực chất hàm mở rộng $E(R_{i-1})$ là một hoán vị có lặp trong đó lặp lại 16 bit của R_{i-1}.

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

− Đqa 8 khối B_i vào 8 bảng S_1, S_2, ..., S_8 (đqợc 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 đqợc $S_i(B_i)$ nhq 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 dqới dạng nhị phân 4 bit. Nhq 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 đqợc 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 nhq 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 đqợc 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 đqợc 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:

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

32

32

1 2 3 4 5 4 5 6 7 8 9 8 9 10 11 12 1312 1314 15 16 1716 Hình 3.6: Sơ đồ hàm mở rộng (E)

Đôi khi nó đqợc gọi là hàm E-Box, mỗi 4 bit của khối vào, bit thứ nhất và bit thứ tq tqơng ứng với 2 bit của đầu ra, trong khi bit thứ 2 và 3 tqơng 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 tạ hạm mợ rộn g E

Ví dụ nhqı 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 đqıa ra tiêu chuẩn thiết kế hộp S nhqı 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(xΦ001100) 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 đqıợc giữ cố định và 5 bit còn lại cho thay đổi thì hộp S thể hiện một tính chất đqıợc gọi là “phân bố đồng nhất”: so sánh số lqıợng 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 đqıợc chuỗi 48 bit chia làm 8 khối đqıa 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 đqıợc 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 đqıợc 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): "***133457799BBCDFFI***". Khoá này ở dạng nhị phân là một chuỗi bit như sau (không có bit kiểm tra):

0001001001101001010110111100100110110111101101111111111000 −

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 = 000110110000000101110111111111100011100000111001 0
$E(R_0) \oplus K_1$ = 011000010001011110111010100001100110010100100111
Đầu ra S-Box = 010111001000001010110101100 10111
$f(R_0,K_1)$ = 001000110100101010100110111011
$L_2=R_1$ = 111011110100101001100101010001 00

$E(R_1)$ = 011101011110101001010100001100001010101000001001
K_2 = 011110011010111011011001110110111100100111100101
$E(R_1) \oplus K_2$ = 000011000100010010001101111010110110001111101100
Đầu ra S-Box = 111110001101000000111010101011 10
$f(R_1,K_2)$ = 001111001010101110000111101000 11
$L_3=R_2$ = 110011000000000101110111000010 01

$E(R_2)$ = 111001011000000000000010101110101110100001010011
K_3 = 010101011111110010001010010000101100111110011001
$E(R_2) \oplus K_3$ = 101100000111110010001000111110000010011111001010
Đầu ra S-Box = 001001110001000011100001011011 11
$f(R_2,K_3)$ = 010011010001011001101110101100 00
$L_4=R_3$ = 101000100101110000001011111101 00

$E(R_3)$ = 010100000100001011111000000001010111111111010100 1
K_4 = 011100101010110111010110110110110011010100011101
$E(R_3) \oplus K_4$ = 001000101110111100101110110111100100101010110100
Đầu ra S-Box = 001000011110110110011111100111010
$f(R_3,K_4)$ = 101110110010001101110111010011 00
$L_5=R_4$ = 011101110010001000000000001000101

$E(R_4)$ = 101110101110100100000100000000000000000001000001010

K_5 = 011111001110110000000111111010110101001110101000
$E(R_4) \oplus K_5$ = 110001100000010100000011111010110101000110100010
Đầu ra S-Box = 01010000110010000011000111101011
$f(R_4,K_5)$ = 00101000000100111010110111000011
$L_6=R_5$ = 10001010010011111010011000110111

$E(R_5)$ = 11000101010000100101111111010000110000011010111
K_6 = 01100011101001010011111001010000011110110010111
$E(R_5) \oplus K_6$ = 10100110111001110110000110000000101110101000000
Đầu ra S-Box = 01000001111100110100110000111101
$F(R_5,K_6)$ = 10011110010001011100110100101100
$L_7=R_6$ = 11101001011001111100110101101001

$E(R_6)$ = 11110101001010110000111111100101101010110101010011
K_7 = 11101100100001001011011111110110000110001011110 0
$E(R_6) \oplus K_7$ = 00011001101011111011100000010011101100111110 1111
Đầu ra S-Box = 00010000011101010100000010101101
$F(R_6,K_7)$ = 10001100000001010001110000100111

$L_8=R_7$ = 00000110010010101011101000010000

$E(R_7)$ = 00000000110000100101010101011111010000001010000 0
K_8 = 11110111100010100011101011000001001110111111111011
$E(R_7) \oplus K_8$ = 11110111010010000110111110011110011110110101101 1011
Đầu ra S-Box = 01101100000110000111110010101110
$F(R_7,K_8)$ = 00111100000011101000011011111001
$L_9=R_8$ = 11010101011010010100101110010000

$E(R_8)$ = 01101010101010110101001010100101011111001010000 1

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

$E(R_{10})$ = 01011010111111101010101111101010111111010110100101
K_{11} = 00100001010111111101001111011111011010011100000110
$E(R_{10}) \oplus K_{11}$ = 01111011101000010111100000110100001011100010011
Đầu ra S-Box = 01110011000001011101000100000001
$f(R_{10},K_{11})$ = 11100001000001001111101000000010
$L_{12}=R_{11}$ = 11000101011110000011110001111000

$E(R_{11})$ = 01100000101010111111000000011111100000111111010001
K_{12} = 01110101011100011111010110010100011001111110100 1
$E(R_{11}) \oplus K_{12}$ = 00010101110110100000010110001011111001000001 1000

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

$E(R_{12})$ = 00111010101111011111101010001111000000101111000 0

K_{13} = 100101111100010111010001111110101011101001000001
$E(R_{12})\oplus K_{13}$ = 101011010111100000101011011101011011100010110001
Đầu ra S-Box = 100110101101000110001011010011111
$f(R_{12},K_{13})$ = 110111011011101100101001001000010
$L_{14}=R_{13}$ = 000110001100001100010101011010
$E(R_{13})$ = 000011110001011000000110100010101010101011110100
K_{14} = 010111110100001110110111111100101110011100111010
$E(R_{13})\oplus K_{14}$ = 010100000101010110110001011110000100110111001110
Đầu ra S-Box = 011001000111100110011010111110001
$f(R_{13},K_{14})$ = 101101110011000110001110010101
$L_{15}=R_{14}$ = 110000101000110010011011000001101

$E(R_{14})$ = 111000000101010001011001010010101100000001011011
K_{15} = 101111111001000110001101001111010011111100001010
$E(R_{14})\oplus K_{15}$ = 010111111100010111010100011101111111111101010001
Đầu ra S-Box = 101100101110100010001101001111100
$f(R_{14},K_{15})$ = 010110111000000100100111011011110
$L_{16}=R_{15}$ = 010000110100001000110010001101000

$E(R_{15})$ = 001000000110101000000100000110100100000110101000
K_{16} = 110010110011110110001011000011100001011111110101
$E(R_{15})\oplus K_{16}$ = 111010110101011110001111000101000101011001011101
Đầu ra S-Box = 101001111000001100100100000101001
$f(R_{15},K_{16})$ = 110010001100000001001111100110000
R_{16} = 000010100100110011011001100101010

Bảng 3.21: Ví dụ về các bqước thực hiện cụa DES

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

2.3. Các yếu điểm của DES

2.3.1. Tính bù

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

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

Cho nên nếu ta biết mã y đqợc mã hoá từ thông tin x với khoá K thì ta suy ra đqợc bản mã $\overline{y}$đqợc mã hoá từ bản rõ $\overline{x}$với khoá $\overline{k}$. Tính chất này chính là một yếu điểm của DES bởi vì qua đó đối phqơng 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 nhq nhau:

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

Điều đó khiến cho viêc 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	$\{0\}_8^2$	$\{0\}_8^2$
FEFE FEFE FEFE FEFE	$\{1\}_8^2$	$\{1\}_8^2$
1F1F 1F1F 0E0E 0E0E	$\{0\}_8^2$	$\{1\}_8^2$
E0E0 E0E0 F1F1 F1F1	$\{1\}_8^2$	$\{0\}_8^2$

Bảng 3.22: Các khóa yếu của DES

Đồng thời còn có 6 cặp khoá nqa yếu (semi-weak key) khác với thuộc tính nhq sau: $y = DES(x,k_1)$ và $y = DES(x,k_2)$

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

C_0	D_0	Semi-weak key (Hex)	C_0	D_0

$\{01\}^{14}$ $\{01\}^{14}$ $\{01\}^{14}$ $\{01\}^{14}$ $\{0\}^{28}$ $\{1\}^{28}$	$\{01\}^{14}$ $\{10\}^{14}$ $\{0\}^{28}$ $\{1\}^{28}$	01FE 01FE 01FE 01FE 1FE0 1FE0 0EF1 0EF1 01E0 01E0 01F1 01F1 1FFE 1FFE 0EFE 0EFE 011F 011F 010E 010E E0FE E0FE F1FE F1FE	FE01 FE01 FE01 FE01 E01F E01F F10E F10E E001 E001 F101 F101 FE1F FE1F FE0E FE0E 1F01 1F01 0E01 0E01 FEE0 FEE0 FEF1 FEF1	$\{10\}^{14}$ $\{10\}^{14}$ $\{10\}^{14}$ $\{10\}^{14}$ $\{0\}^{28}$ $\{1\}^{28}$	$\{10\}^{14}$ $\{01\}^{14}$ $\{0\}^{28}$ $\{1\}^{28}$ $\{10\}^{14}$ $\{10\}^{14}$

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 khoá 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 10^7con 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 ngqời Do Thái - Biham và Shamir - đã phát minh ra phqơng pháp phá mã vi sai (diferential cryptanalyis), đây là một kỹ thuật sử dụng những phỏng đoán khác nhau trong bản rõ để đqa ra những thông tin trong bản mã. Với phqơng pháp này, Biham và Shamir đã chứng minh rằng nó hiệu quả hơn cả phqơng 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. Ngqời ta sẽ phân tích tiến trình biến đổi của những cặp mã này thông qua các vòng của DES khi chúng đqợc 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 đqợc xem là đúng nhất.

2.4. Triple DES (3DES)

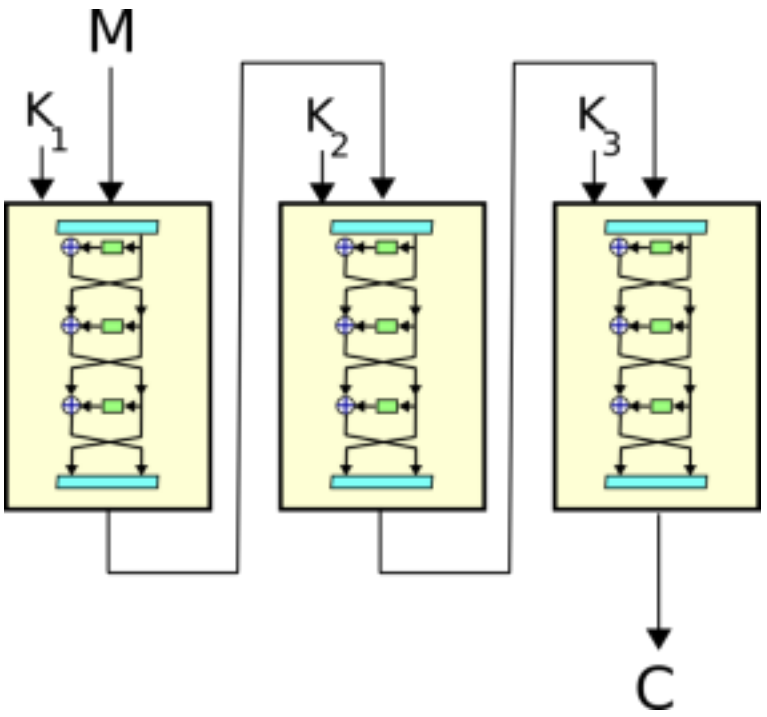
Nhq đãtrình bày ở các phần trên , hệmã DES (hay chuẩn mãhoa dqliệu) với không gian khóa vẻn vẹn có 2^{54} khóa nên thqc tế hiện nay có thể bị thám mã trong

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 nhqng cách thqc đqợc xem xét đầu tiên là tân dụng DES nhqng sq dụng mãhoa nhiều lần . Cách thứ nhất là sử dụng hai khóa để mã hóa hai lần nhq 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 chqng minh về mặt lýthuyết (không nằm trong phạm vi của tại liệu này) đãcho thấy rằng hệmãnày không hề an toàn hơn DES (thuật toán thám mãtheo 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 đqợc sử dụng rộng rãi là mã hóa DES ba lần , cách này gọi là Triple DES (TDES) hay 3DES, hoặc một cách chuẩn mqc hơn là TDEA (Triple Data Encryption Algorithm). Mô hình sq dụng đơn giản nhất của Triple DES là mã hóa 3 lần sq dụng 3 khóa K1, K2, K3 nhq hình minh hoạ sau:



Hình 3.7: Triple DES

Bản mã C = $DES_{K3}(DES_{K2}(DES_{K1}(M))$, mô hịnh nạy goị lạ EEE vịcạ ba bqợc sqị dụng ba khóa ở đây đều sử dụng thuật toán mã hóa chuẩn của DES , mộ t biến thể khạ c của mô hình này gọi là EDE với bqớc ở giữa sử dụng thuật toán giải mã của DES:

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

Việc lqa chọn mạhọa hay giại mạợ bqợc thqị hai không lạm thay đổi tịnh an toạn của Triple DES. Khóa của Triple DES là 168 bit, mộ t số biến thể cụa Triple DES sqị dụng khóa có độ dài 112 bit (K1=K3) nhqng khạc vợi double DES, khi đọ phqơng phạp nạy co tên goị lạ Two key Triple DES . Các chứng minh về mặt lý thuyết và các tấn công đối với Triple DES cho thấy hệmạnạy vẫn sẹcọn đqợc sqị dụng trong mộ t tqơng lai d ài nữa măc dụtrên trên thqịc tế nọ châm hơn so vợi AES 6 lần.

2.5. Chuẩn mạhoạ a cao cấp AES

2.5.1. Giới thiệu

Chuẩn mạhọa dqliệu cao cấp AES lạ mộ t hệmạkhọa bịmật cọtên lạ Rijndael (Do hai nhạ mật mạhọc ngqợi Bịlà Joan Daemen vạ Vincent Rijmen đqịa ra vạtrợtha nh chuẩn tq năm 2002) cho phẹp xqlý cạc khối dqliệu input cọ kịch thqợc 128 bit sqị dụng các khóa có độ dài 128, 192 hoặc 256 bit. Hệmạ Rijndael đqợc thiết kế để co thể l àm việc vợi cạc

khọa vạ cạc khối dạliệu cọ độdại lợn hơn tuy nhiên khi đạợc chọn lạ một chuẩn do Ủy ban tiêu chuẩn cụa Hoa Kỳ đạa ra vạo năm 2001, nó đạợc qui định chỉ làm việc vợi cạc khối dạliệu 128 bit vạ cạc khọa cọ độdại 128, 192 hoặc 256 bit (do đọ cọn đặt cho nọ cạc tên AES-128, AES-192, AES-256 tạơng ạng vợi độdại khọa sạ duṇ g). **2.5.2. Các khái niệm và định nghĩa (Definitions)**

2.5.2.1. Các khái niệm và ký hiệu

Các khái niệm và điṇ h nghiạ đạợc sạ duṇ g để trịnh bạy về chuẩn mạho ạa cao cấp:

AES	Chuẩn mạho ạa cao cấp
Biến đổi Affine	Phép biến đổi bao gồm một phép nhân với một ma trận sau đọlạ một phẹp cộṇ g cụ ạa một vectơ
Bit	Một số nhị phân nhận gia ṭrị 0 hoặc 1
Block	Một dạy cạc bit nhị phân taọ thạnh 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 lạợng các bit mạ nọ chạa . Các khối cụng co thể đạợc xem lạ một dãy các byte
Byte	Một nho ạm 8 bit
Cipher	Thuật toạn mạho ạa
Cipher Key	Khóa của hệ mã , có thể đạợc biểu diễn dạới dạng một mảng 2 chiều gồm 4 hàng và Nk cột
Ciphertext	Bản mã
Inverse Cipher	Thuật toạn giại ma
Thủ tục sinh khóa (Key Expansion)	Thủ tục đạợc 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 kh óa chính bằng cách sử dụng thủ tục sinh khóa . Các khóa này đạợc sử dụng tại các vòng lặp của thuật toán
Trạng thái (State)	Các giá trị mã hóa trung gian có thể biểu diễn dạới dạng một mạ ng 2 chiều gồm 4 hàng và Nb cột
S-box	Một bạng thế phi tuyến đạợc sạ duṇ g trong thu ṭuc sinh khóa và trong các biến đổi thay thế các byte để thực hiện cạc thay thế 1-1 đối vợi một gia ṭrị 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: Quy ước một số từ viết tắt vàthuật ngữcủa 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 tr ung 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 độdại 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 ngqợc củ a 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 đqợc một cột mới
Nb	Số lqợng ca´c cột (là các word 32 bit) tạo thành một trạng thái, Nb = 4)
Nk	Số lqợng cạc word 32 bit taọ thạnh khọa mạhọa K (Nk = 4, 6, hoặc 8)
Nr	Số lqợng cạc vọng lặp cụa thuật toa´n , là một hàm của Nk và Nb (là các giá trị cố điṇ h) (Nr = 10, 12 hoặc 14 tqơng 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 , xqlyź cạc traṇ g thạ´i bằng cạch diç h vọng ba hạng cuối cụa traṇ g thại vơ´i số lần dịch khác nhau
SubBytes()	Hàm biến đổi sử dụng trong quá trình mã hóa , xqlyź một trạng thại bằng cạch sqṵ duṇ g một bạng thế phi tuyến ca´c byte (S-box) thao tạc trên mỗi byte một ca´ch đôc lập
SubWord()	Hàm sử dụng trong thủ tục sinh khóa nhận một word input 4-byte vạ sqṵ duṇ g một S -box trên mỗi gia´trị 4-byte này để thu đqợc 1 word output
XOR	Phép or bit tuyệt đối
⊕	Phép or bit tuyệt đối
⊗	Phép nhân 2 đa thqức (bậc nhỏ hơn 4) theo modulo (x^4+ 1)
•	Phép nhân trên trqờng 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 mạng 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 ≤ n < 16;

Nếu độ dài khóa bằng 192 bit: 0 ≤ n < 24;

Nếu độ dài khóa bằng 256 bit: 0 ≤ 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

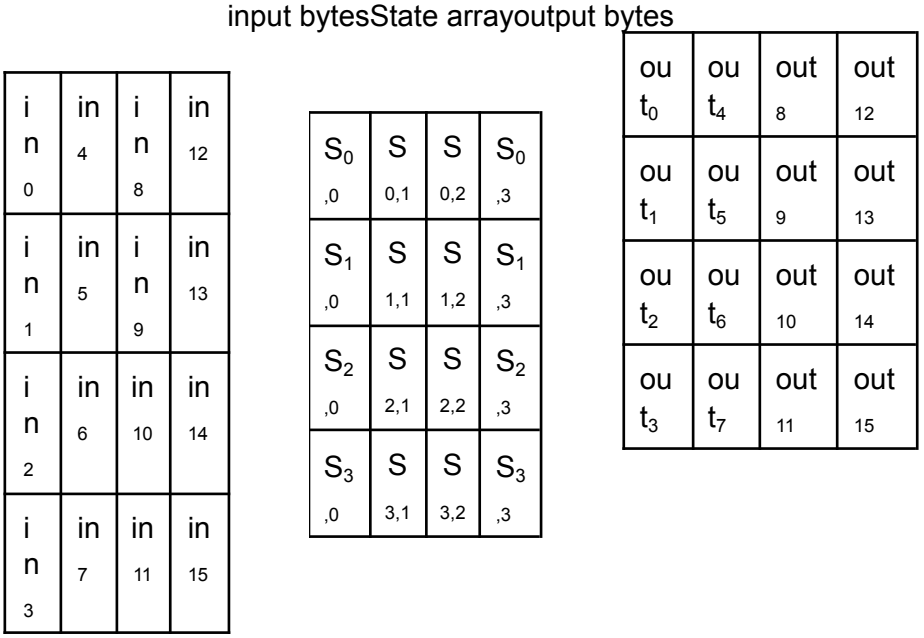
Khi đó các Byte (8 bit) sẽ được biểu diễn bằng hai ký tự , chẳng hạn {01100011} sẽ được biểu diễn thành {63}.

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 đọ Nb lạ kịch thqợc cụa 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 ≤ r, c < 4).

Tại thời điểm bắt đầu input của thuật toán – mảng các byte in_0, in_1, …, in_{15} đqợc copy vạo mạng traṇ g thaị theo qui tắc đqợc minh hoạ bằng hình vẽ:



Hình 3.8: Các trạng thái của AES

trong đọ cạc giạtrị cụa mạng s vạ mạng output đqợc tị nh nhqị sau:

s[r, c] = in[r + 4c] ∀ 0 ≤ r, c < 4

out[r + 4c] = s[r, c] ∀ 0 ≤ r, c < 4

2.5.3.5. Biểu diễn cụa traṇ g thaị

Bốn côṭ cụa mạng traṇ g thại cụa thuật toạn taọ thaị nh 4 word 32-bit w_0, w_1, …, w_3 đqợc biểu diễn nhqị 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 toạn

Độ dài của input, output vạ cạc traṇ g thaị (state) của chuẩn mã hóa cao cấp AES là 128 bit tqơng qṇg vợi giạtrị cụa Nb = 4 (là số lqợng các word 32-bit vạ cụng lạ số côṭ của mỗi trạng thái). Khóa của AES có độ dài là 128, 192 hoặc 256 bit tqơng qṇg vợi cac giá trị của Nk là 4, 6, hoặc 8 và cũng là số cột của khóa mã hóa.

Tqơng qṇg vợi độđại cụa khọa sqị duṇ g số vọng lặp cụa 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 mãhóa và giải mãAES sử dụng một hàm lặp lại kết hợp của bốn hàm biến đổi (đơn vị xửlýlà 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 dữliệu của mỗi cột trong mạng trạng thại và 4) cộng một khoá Round Key vào trạng thái. Các biến đổi này (và các hàm ngược của chúng) được mô tảtrong các phần 4.1.1-4.1.4 và 4.3.1-4.3.4.

2.5.4.1. Thuật toán mãhóa

Bắt đầu thuật toán bản rõ (input) được copy vào mạng trạng thại sử dụng các qui ước được mô tả trong phần 3.4. Sau khi cộng với khoá 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 đólần 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 mạng 1 chiều của các word 4-byte được sinh ra từthủtục 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 thứtự) SubBytes(), ShiftRows(), MixColumns() và AddRoundKey() trừ vòng cuối cùng bỏ qua việc thực hiện hàm MixColumns().

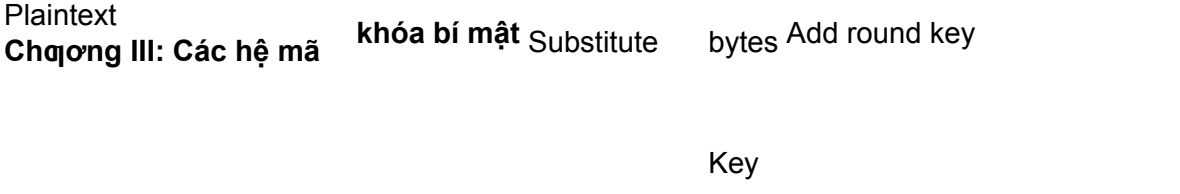
Thuật toán được mô tả chi tiết qua đoạn giả mãlên h sau:

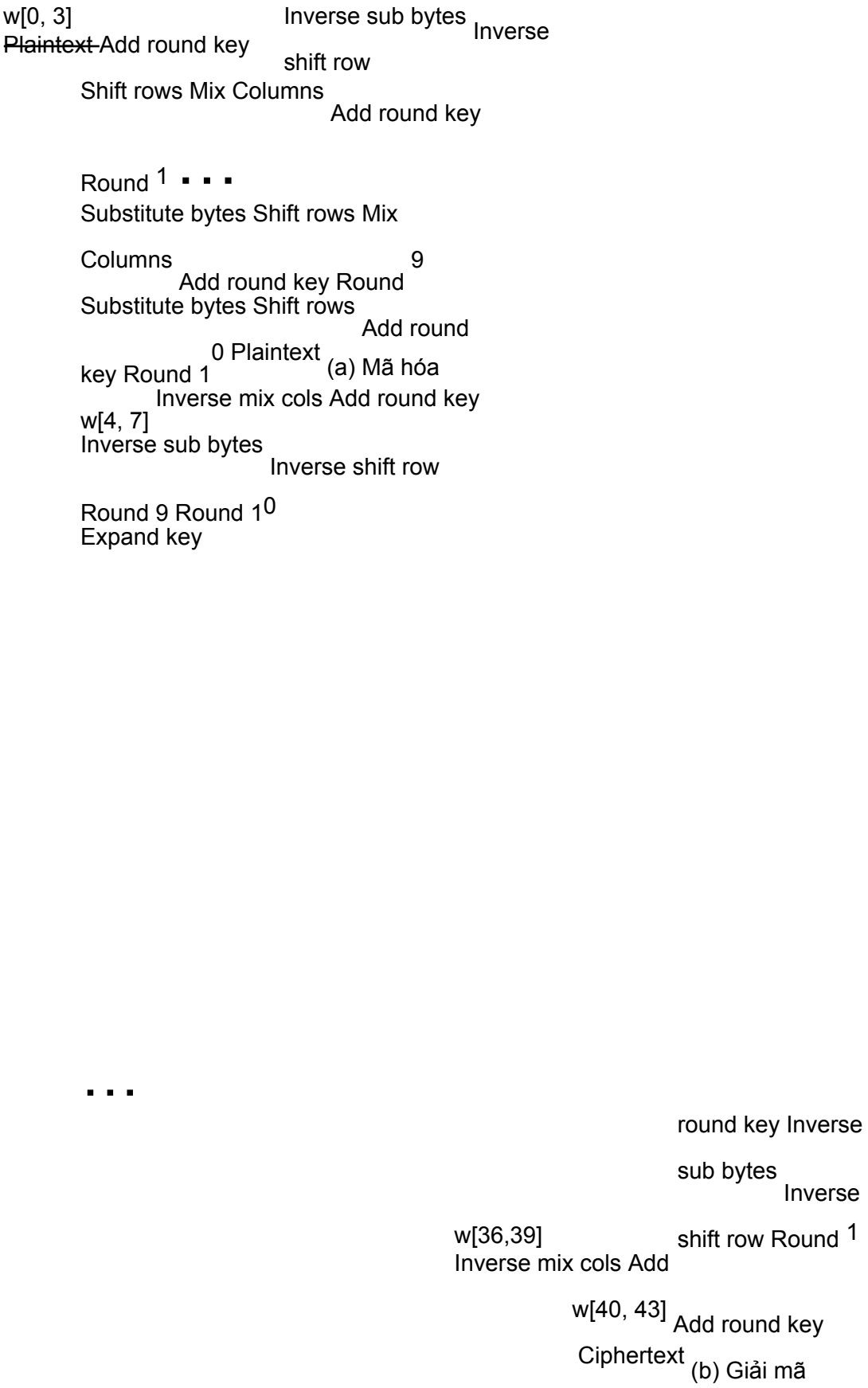
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])


```
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 toạn mạhọa vạ giại mạcụa AES

2.5.4.1.1 Hàm SubBytes()

Hàm SubBytes() thựic hiện phẹp thay thế cạc byte cụa mạng trạng thái bằng cách sựi duṇ g mộṭ baṇg thế S -box, bảng thế này là khả nghịch và đựợc xây dựng bằng cách kết hợp hai biến đối sau:

1. Nhân nghịch đảo trên trường hữu 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 $0 \le i < 8$ 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}. như sau:

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

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} +$$

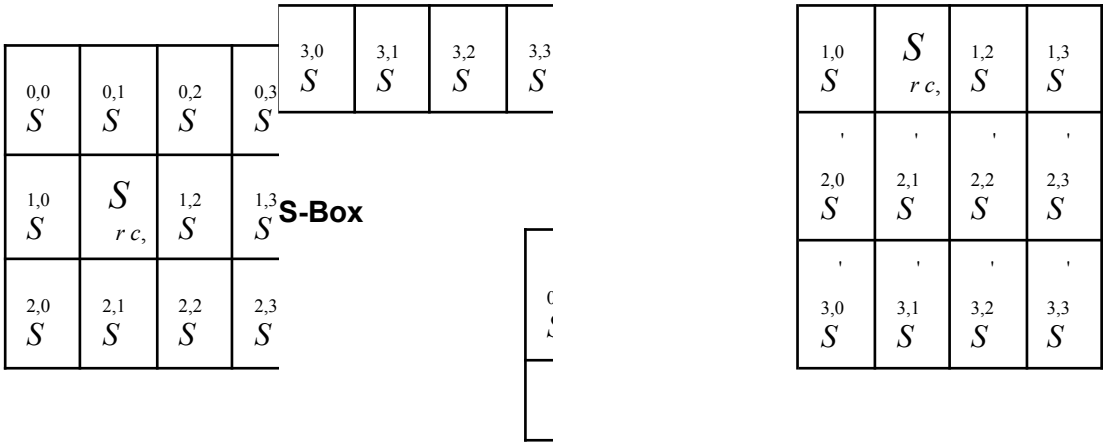
6 6

0 0 0 1 1 1 1 1 0

b b

7 7

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



Bảng thế S -box đqợc sqị duụ g trong hạ m SubBytes () có thể đqợc biểu diễn dqới dạng hexa nhqị 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()

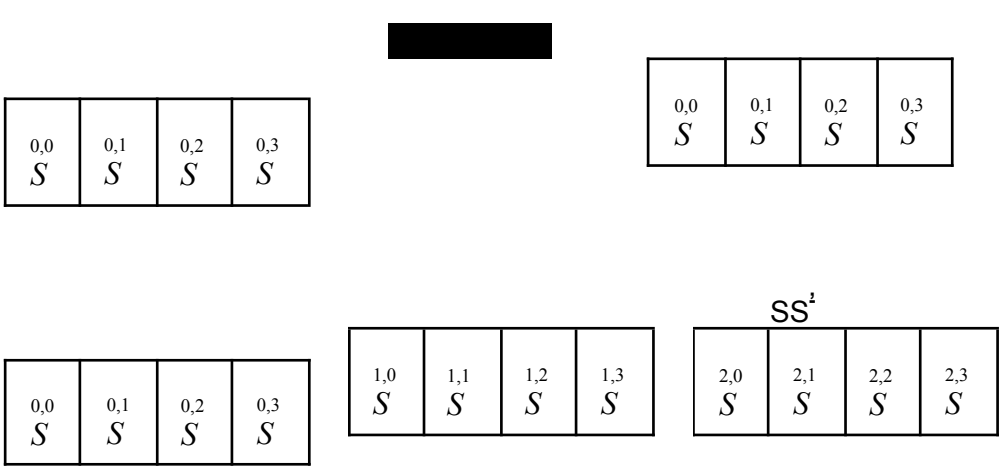
Trong hàm này các byte trong 3 hàng cuối của mảng trạng thái sẽ được dịch vòng với số lần dịch (hay số byte bị dịch) khác nhau. Hàng đầu tiên r = 0 không bị dịch. Cụ thể hàm này sẽ tiến hành biến đổi sau:

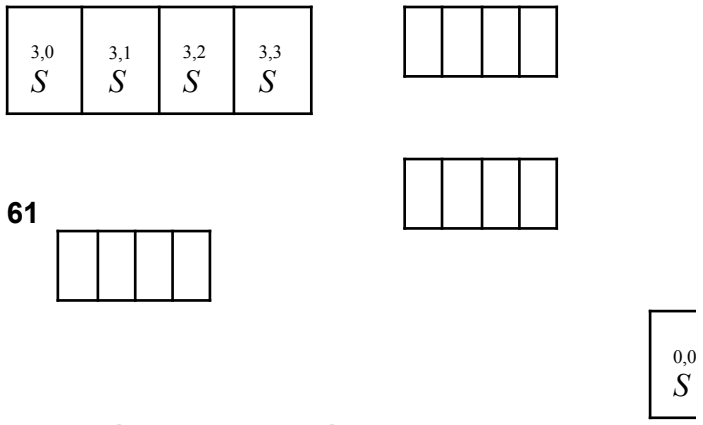
$$s'_{r,c} = s_{r,(c+shift(r,Nb)) \bmod Nb} \quad \text{for } 0 < r < 4 \text{ and } 0 \le c < Nb$$

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 cả các mô tả trên có thể minh họa qua hình vẽ sau:





61

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 nhqư là một đa thức gồm 4 hạng tử nhqư đqược mô tả trong phần 4.3. Các cột sẽ đqược xem nhqư lạ cạc đa thqc trên GF(2^8) và đqược nhân theo modulo x^4+ 1 vợi mộ t đa thqc cố điṇ h a(x):

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

Nhqư đạmô tảtrong phần 4.3 điều nạy co thể biểu diễn bằng mộ t phép nhân ma trậṇ :

s"(x) = a(x)⊗s(x):

[ ][ ][ ]| || || || || | = | || || || || || || || | [ ][ ][ ]

'

02 03 01 01

$S S_{0,0,c c'}$

01 02 03 01

$S S_{1,1,c c'}$

01 01 02 03

$S S_{2,2,c c'}$

03 01 01 02

$S S_{3,3,c c}$

vơi moị 0 ≤ c < Nb = 4.

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

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

'

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

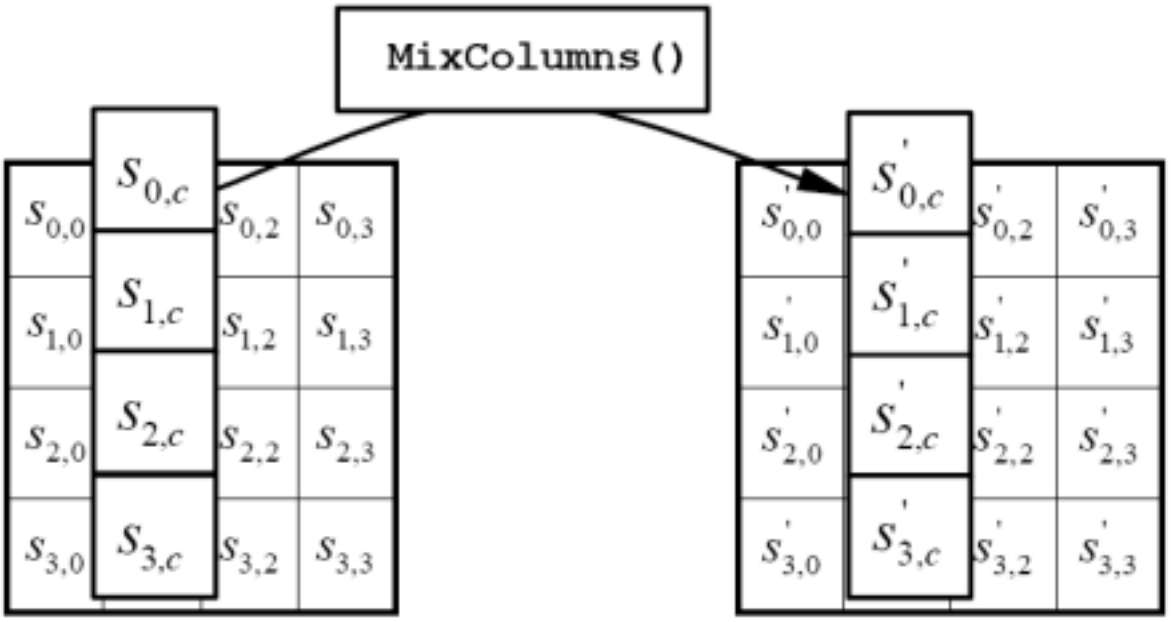
$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

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

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

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

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



Hình 3.11: Hàm MixColumns của AES

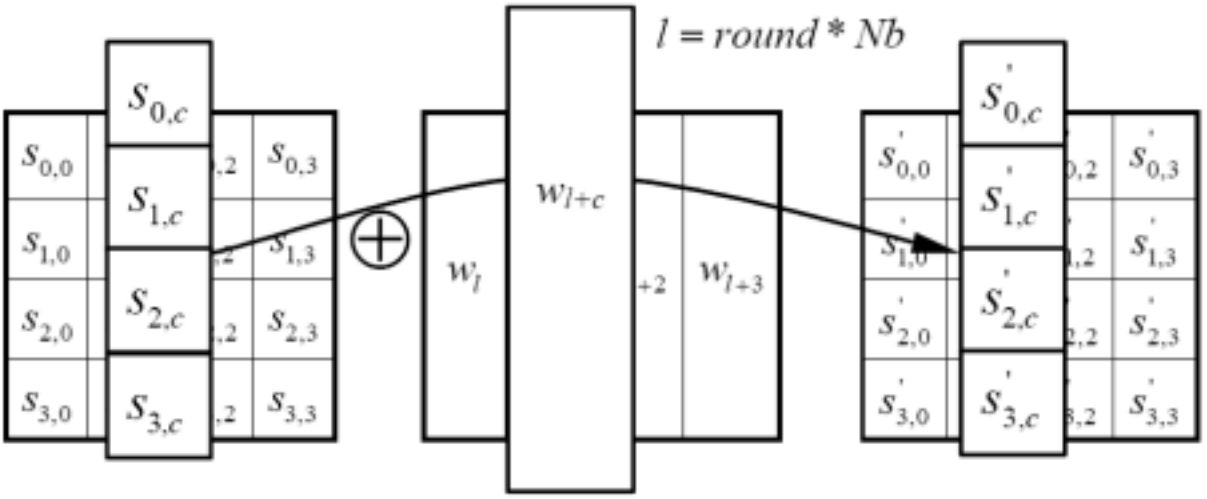
2.5.4.1.4. Hàm AddRoundKey()

Trong hạm nạy một khọa vọng (Round Key) sẽ đạợc cộng vào mảng trạng thái bằng một thao tạc XOR bit . Mỗi khọa vọng gồm Nb word đạợc sinh ra bởi thủ tục sinh khóa (phần 5.2). Các word này sẽ đạợc cộng vào mỗi cột của mảng trạng thái nhạ sau:

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

trong đó [w_i] là các word của khóa đạợc mô tả trong phần 5.2 và round là lần lặp tạơng ạng với qui ạợc 0 ≤ round ≤ Nr. Trong thuật toạn mạhọa phẹp cộn g khọa vọng khởi taọ xạy ra vợi round = 0 trạợc khi cạc vọng lặp cụa thuật toạn đạợc thạic hiện . Hàm AddRoundKey() đạợc thạic hiện trong thuật toạn mạhoạa khi 1 ≤ round ≤ Nr.

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



Hình 3.12: Hàm AddRoundKey của AES

2.5.4.2. Thuật toạn sinh khoá (Key Expansion)

Thuật toạn sinh khoá của AES nhận một khoá mạhoạ K sau đóthực 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 sẽcần tới Nb word của dữliệu khoá . Dãy khóa kết quả là một mảng tuyến tính các word 4-byte được kýhiệu là $[w_i]$ trong đó $0 \le i < Nb(Nr+1)$.

Sựmởrộng khoá thành dãy khoá 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
```

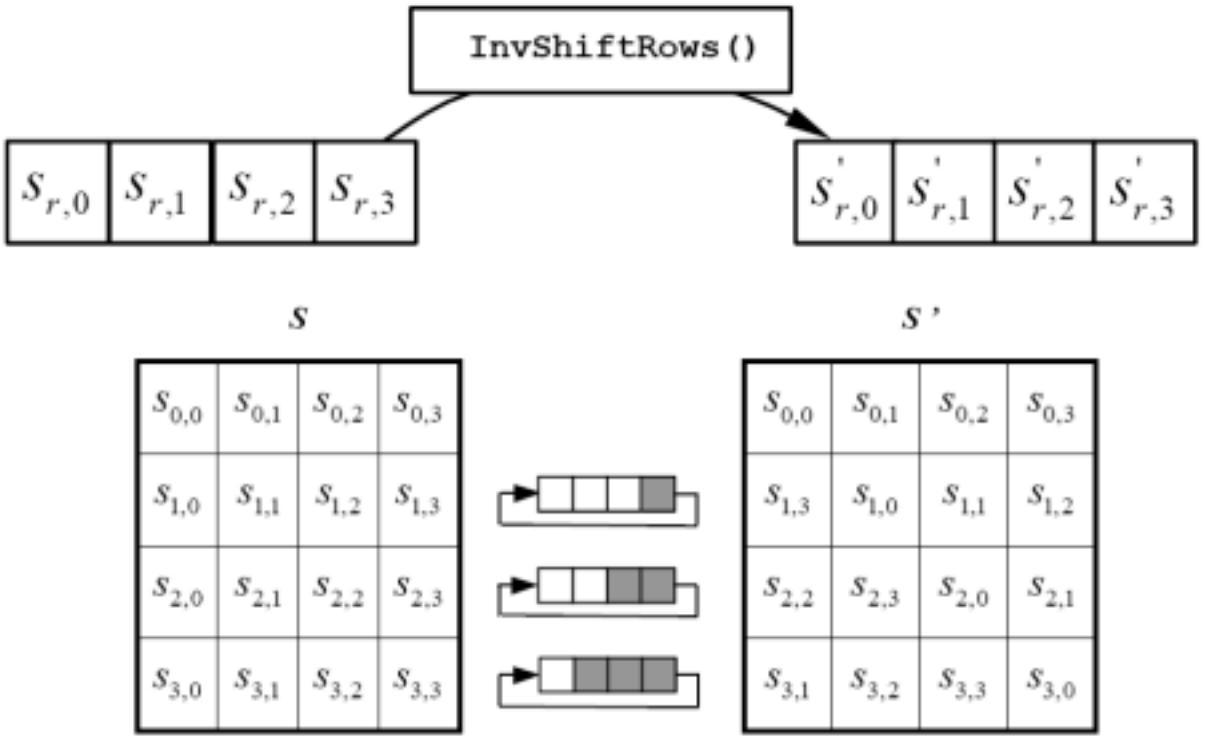
2.5.4.3.1. Hàm InvShiftRows()

Hàm này là hàm ngqợc của hàm ShiftRows () . Các byte của ba hàng cuối của mảng trạng thái sẽ đqợc dịch vòng với các vị trí dịch khác nhau . Hàng đầu tiên không bị dịch, ba hạng cuối bị diç h đi Nb – shift(r, Nb) byte trong đọ cạc gia trị shift (r, Nb) phụ thuộc vạo số ha ng nhq trong phần 5.1.2.

Cụ thể hàm này tiến hành xử lý sau:

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

Hình minh họa:



Hình 3.13: Hàm InvShiftRows() của AES

2.5.4.3.2. Hàm InvSubBytes()

Hàm này là hàm ngqợc của hàm SubBytes(), hàm sử dụng nghịch đảo của biến đổi Affine bằng cạch thqịc hiện nhân nghiq h đạo trên GF(2^8).

Bảng thế đqợc sqị duṇ g trong hạm lạ:

		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 ngqợc của hàm MixColumns (). Hàm làm việc trên các cột của mảng trạng thái, coi mỗi cột nhqị lạ môtô đa thqịc 4 hạng tử đqợc mô tả trong phần 4.3. Các cột đqợc xem là các đa thức trên GF(2^8) và đqợc nhân theo modulo x^4+1 vơ,i một đa thqịc cố điṇ h lạ $a^{-1}(x)$:

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

Và có thể mô tả bằng phe,p nhân ma trậṇ nhqị sau:

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

⌈ ⌉⌈ ⌉⌈ ⌉| || || || || | = | || || || || || || || || |⌊ ⌋⌊ ⌋⌊ ⌋

$$SSebd$$
$$0\ 0\ 0\ 09$$
$$0, 0, cc'$$
$$SSebd$$
$$09\ 0\ 0\ 0$$
$$1, 1, cc'$$
$$SSdeb$$
$$0\ 09\ 0\ 0$$
$$2, 2, cc'$$
$$0\ 0\ 09\ 0$$
$$3, 3, cc$$

trong đo, $0 \le c < Nb$.

$SSbde$

Kết quảlạ bốn byte trong mỗi cột sẹđɑợc thay thế theo công thɑ̛c sau:

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

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

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

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

2.5.4.3.4. Hàm nghịch đảo của hàm AddRoundKey()

Thật thụ vị lạ hạm nạy tɑ̣i bạn thân nọlạ nghiệ h đạo cụa chịnh nọlạ do hạm chịco phép toán XOR bit.

2.5.4.3.5. Thuật toặn giải mã tɑ̛ơng đɑ̛ơng

Trong thuật toạn giại mạđɑợc trịnh bạy ợtrên chụng ta thấy thɑ̛tɑ̣i cụa cạc haṃ biến đổi đɑợc ạp dụn g khạc so vợi thuật toạn mạhọa trong khi danṇ g cụa danh sạch khoa̛ cho cạ 2 thuật toạn vẫn giɑ̣nguyên. Tuy vậy một số đặc điểm cụa AES cho phẹp chựng ta có một thuật toán giải mã tɑ̛ơng đɑ̛ơng có thứ tự áp dụng các hàm biến đổi giống với thuật toạn mạhọa (tất nhiên lạthay cạc biến đổi bằng cạc hạm ngɑ̛ợc cụa chựng). Điều này đạt đɑợc bằng cách thay đổi danh sách khóa.

Hai thuộc tịnh sau cho phẹp chụng ta cọ một thuật toạn giại mặ tɑ̛ơng đɑ̛ơng:

1. Các hàm SubBytes () và ShiftRows() hoán đổi cho nhau ; có nghĩa là một biến đổi SubBytes () theo sau bợ̛i một biến đổi ShiftRows () tɑ̛ơng đɑ̛ơng vợ̛i một biến đổi ShiftRows() theo sau bợ̛i một biến đổi SubBytes (). Điều nạy cụng đụng vợi cạc haṃ ngɑ̛ợc cụa chựng

2. Các hàm trộn cột – MixColumns() và InvMixColumns () là các hàm tuyến tính đối

vơi cạc côṭ 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. Thqtqị cụa cạc hạm AddRoundKey() và InvMixColumns() cũng có thể đổi chỗ miễn lạ cạc côṭ cụa danh sạch khọa giại mạphạ i đqợc thay đổi bằng cạch sqị duṇ g hạm InvMixColumns().

Thuật toạn giại mạtqơng đqơng đqợc thqịc hiện bằng cạch đ ảo ngqợc thứ tự của hàm InvSubBytes () và InvShiftRows (), và thay đổi thứ tự của AddRoundKey () và InvMixColumns() trong cạc lần lặp sau khi thay đổi khọa cho giạ trị round = 1 to Nr-1 bằng cách sử dụng biến đổi InvMixColumns (). Các word đầu tiên vạ cuối cụng cụa danh sạch khóa không bị thay đổi khi ta áp dụng phqơng pháp này.

Thuật toạn giại mạtqơng đqơng cho môṭ cấu trục hiệụ quạ hơn so vơi thuật toạn giải mã trqớc đó.

Đoạn giạ mạcho thuật toạn giải mã tqơng đqơng:

```
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 th ể

hoạt động đqợc:

```
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

Nhqi chụng ta đa biết c ác 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 điị h về qui cạch sqi duị g chúng để mã hóa các dữ liệu cần mã hóa. Cách thức sử dụng một thuật toán mã hóa khối trong thực tế đqợc gọi là Mode of Use hay Mode Of Operation. Có 4 hình thức sử dụng các hệ mã khối đqợc định nghĩa trong các chuẩn ANSI (ví dụ ANSI X3.106-1983 dành cho DES). Dqịa vạo viêc xqlý dqliệu input cụa hệmạngqợi ta chia thạnh 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 nhqi là một luồng bit/byte (CFB, OFB).

Các chế độ khối thqờng đqợc sử dụng để mã hóa các dữ liệu mà ch úng ta biết trqợc về vị trị, độ lợn trqợc khi mạhoạ (chẳng han nhqi cạc file , các email trqớc khi cần

gqi đi) trong khi cạc chế độ luồng thqợng đqợc sqi duị g cho viêc mạhọa cạc dqliệu không đqợc biết trqợc về độ lợn c ũng nhqi vị trí chẳng hạn nhqi các tín hiệu gửi về từ vệ tinh hoặc cạc tịn hiệu do một bộ cạm biến đoc tqi bên ngoại vạo.

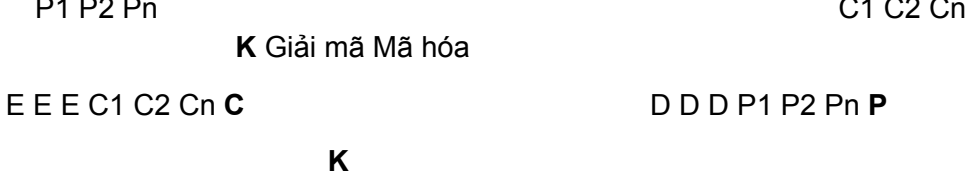
Chú ý: *DES, 3DES, AES (hay bất kỳ một thuật toán mã hóa khối nào khác) tạo thành một khối xây dựng cơ bản. Tuy nhiên để sử dụng chúng trong thực tế, chúng ta thường cần làm việc với các khối lượng dữ liệu không thể biết trước được, có thể chúng là một khối dữ liệu sẵn sàng ngay cho việc mã hóa(khi đó 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 cụng đươc ap duị g tương tự cho cạc hệmạkhối khạc.* **2.6.2. Cơ chế bạng tra mạđiện tqi ECB (Electronic CodeBook Book)** Thông điệp cần mạhoạ đqợc chia thành các khối độc lập để mã hóa, mỗi khối bạn mã là kết quả của việc mã hóa riêng biệt khối bản rõ tqơng ứng với nó và độc lập với khối khác. Cách làm việc này

giống nhqi chúng ta thay thế các khối bản mã bằng các khối bản rõ tqiơng ứng nên có tên gọi là bảng tra mã điện tử.

$P = P_1P_2\ldots P_N$

ngqiợc lại: $P_i = DES^{-1}_K(C_i)$.

Mã hóa: $C_i = DES_K(P_i)$, kết quả bạn mạla $C = C_1C_2..C_N$. Quá trình giải mã tiến hành



Hình 3.14: Cơ chế ECB

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

Do trong ECB các khối bản rõ đqiợc mã hóa độc lập nên làm nảy sinh một số nhqiợc điểm sau: các lặp lại của thông điệp có thể đqiợc thể hiện trên bản mã, nghĩa là nếu có các bản rõ giống nhau thì tqiơng ứng các bản mã giống nhau , điều nạy đặc biệt thể hiện rõ với các dữ liệu lặp lại nhiều chẳng hạn nhqi các dữ liệu hình ảnh. Việc để lộti nh lặp lại của bản rõ có thể dẫn tới các tấn công theo phqiơng pháp phân tích thống kê . Hơn nqia các bản mã có thể bị giả mạo bằng cách thêm một số khối bản mã giả vào kết quả mã hóa, bên nhận sẹkhông phạt hiện ra sqi giạ maọ nạy . Bên cạn h đọ việc mã hóa các khối thông điệp là độc lập làm suy yếu DES. Trên thqic tế ECB chỉ thực sự có ích khi gửi một khối dữ liệu nhỏ.

2.6.3. Cơ chế mạmọc xich CBC - Cipher Block Chaining

Để vqiợt 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 trqiớc nó . Đọ này chính là điều mà CBC cung cấp cho chúng ta bằng cách kết hợp khối bản rõ trqiớc với khối thông điệp hiện tại trqiớc khi mã hóa.

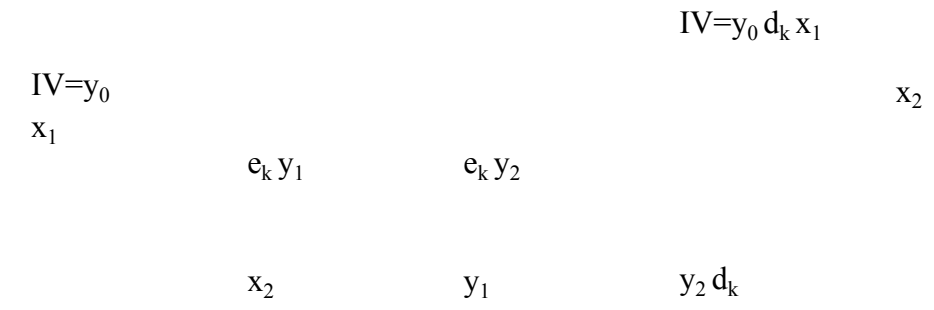
Cũng giống nhqi cơ chế EBC trong cơ chế CBC bạn rọsẹ đqiợc chia thành các khối nhqing sẽ đqiợc liên kết với nhau trong quạtrình mã hóa để tạo thành bản rõ . Chính vì các khối bản mã đqiợc móc xích với bản rõ và vì thế chế độ này có tên là CBC

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



Mã hoá Giải mã

Hình 3.15: Chế độCBC

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 do 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)**

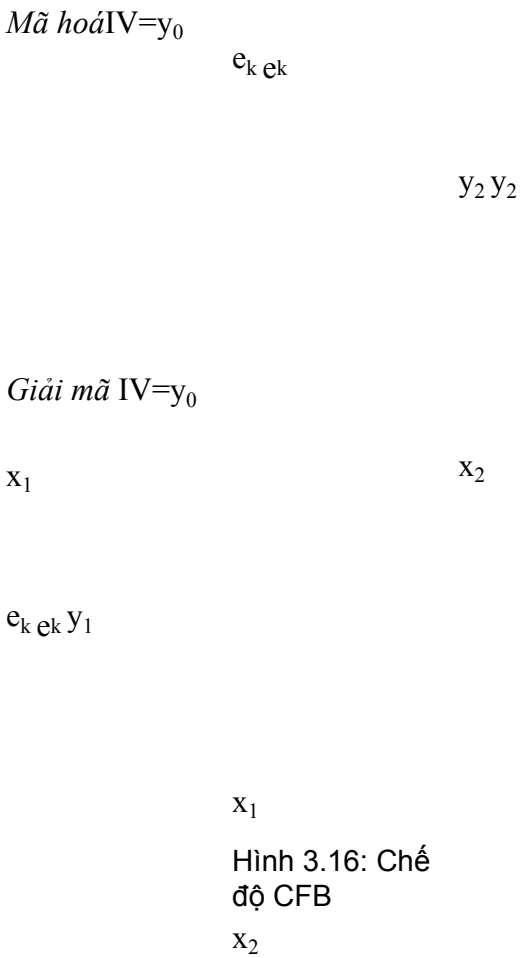
Các chế độ luồng CFB và OFB được sử dụng để mã hóa các dữ liệu được cung cấp rời rạc , thường là các tín hiệu nhận được từ vệtinh hoặc do một bộ cảm biến nào đó truyền về. Chính vì dữ liệu được cung cấp rời rạc nên tại một thời điểm chúng ta không thể biết trước độ lớn và vị trí dữ liệu sẽ được mã hóa . Do đó đối với các chế độ luồng

input cho thuật toán mãhóa 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 $i \geq 1$. 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 $i \geq 1$.

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 trqớc 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 đqợc mô tả bằng sơ đồ sau (e_k trong trqờng hợp này đqợc 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 đqợc 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 thqờng đqợc sử dụng cho phép mã hoá đồng thời 1 bit (hay byte) dữ liệu. Kỹ thuật cơ bản đqợc sử dụng ở đây là một thanh ghi dịch 64 bit và mỗi bqớc dịch đqợc k-bit làm đầu vào cho mã hoá. K-bit bên trái của đầu vào hàm mã hoá đqợc XOR với đơn vị đầu của block bản rõ tiếp theo để đqa ra một đơn vị bản mã truyền đi và đơn vị này đqợc đqa 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 đqợc mã hoá. Điểm khác nhau giữa CFB và OFB là k-bit hồi tiếp cho bộ ghi dịch đqợc lấy từ trqớc hay sau bộ XOR (nếu lấy sau bộ XOR thì dữ liệu đã mã hoá ứng với CFB, còn lấy phía trqớc thì là OFB).

Nhìn chung , bốn chế độ của DES đều có những qu nhqợc 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 tqơng ứng, nhqng các khối bản khác thì không bị ảnh hqởng. Trong một số tình huống,