MỤC LỤC

1. AN TOÀN DỮ LIỆU TRÊN MẠNG MÁY TÍNH

Ngày nay, với sự phát triển mạnh mẽ của công nghệ thông tin việc ứng dụng các công nghệ mạng máy tính trở nên vô cùng phổ cập và cần thiết. Công nghệ mạng máy tính đã mang lại những lợi ích to lớn.

Sự xuất hiện mạng Internet cho phép mọi người có thể truy cập, chia sẽ và khai thác thông tin một cách dễ dàng và hiệu quả. Các công nghệ E-mail cho phép mọi người có thể gửi thư cho người khác cũng như nhận thư ngay trên máy tính của mình. Gần đây có công nghệ E-business cho phép thực hiện các hoạt động thương mại trên mạng máy tính. Việc ứng dụng các mạng cục bộ trong các tổ chức, công ty hay trong một quốc gia là rất phong phú. Các hệ thống chuyển tiền của các ngân hàng hàng ngày có thể chuyển hàng tỷ đôla qua hệ thống của mình. Các thông tin về kinh tế, chính trị, khoa học xã hội được trao đổi rông rãi.

Tuy nhiên lại nảy sinh vấn đề về an toàn thông tin. Đó cùng là một quá trình tiến triển hợp logic: khi những vui thích ban đầu về một siêu xa lộ thông tin, bạn nhất định nhận thấy rằng không chỉ cho phép bạn truy nhập vào nhiều nơi trên thế giới, Internet còn cho phép nhiều người không mời mà tự ý ghé thăm máy tính của bạn.

Thực vậy, Internet có những kỹ thuật tuyệt vời cho phép mọi người truy nhập, khai thác, chia sẻ thông tin. Những nó cũng là nguy cơ chính dẫn đến thông tin của bạn bị hư hỏng hoặc phá huỷ hoàn toàn.

Có những thông tin vô cùng quan trọng mà việc bị mất hay bị làm sai lệch có thể ảnh hưởng đến các tổ chức, các công ty hay cả một quốc gia. Các thông tin về an ninh quốc gia, bí mật kinh doanh hay các thông tin tài chính là mục tiêu của các tổ chức tình báo nước ngoài về chính trị hay công nghiệp hoặc kẻ cắp nói chung. Bọn chúng có thể làm mọi việc có thể để có được những thông tin quý giá này. Thử tưởng tượng nếu có kẻ xâm nhập được vào hệ thống chuyển tiền của các ngân hàng thì ngân hàng đó sẽ chịu những thiệt hại to lớn như mất tiền có thể dẫn tới bị phá sản. Chưa kể nếu hệ thông thông tin an ninh quốc gia bị đe doạ thì hậu quả không thể lường trước được.

Theo số liệu của CERT(Computer Emegency Response Team - "Đội cấp cứu máy tính"), số lượng các vụ tấn công trên Internet được thông báo cho tổ chức này là ít hơn 200 vào năm 1989, khoảng 400 vào năm 1991, 1400 vào năm 1993, và 2241 vào năm 1994. Những vụ tấn

công này nhằm vào tất cả các máy tính có mặt trên Internet, các máy tính của tất cả các công ty lớn như AT&T, IBM, các trường đại học, các cơ quan nhà nước, các tổ chức quân sự, nhà băng... Một số vụ tấn công có quy mô khổng lồ (có tới 100.000 máy tính bị tấn công). Hơn nữa, những con số này chỉ là phần nổi của tảng băng. Một phần rất lớn các vụ tấn công không được thông báo, vì nhiều lý do, trong đó có thể kể đến nỗi lo bị mất uy tín, hoặc đơn giản những người quản trị hệ thống không hề hay biết những cuộc tấn công nhằm vào hệ thống của họ.

Không chỉ số lượng các cuộc tấn công tăng lên nhanh chóng, mà các phương pháp tấn công cũng liên tục được hoàn thiện. Điều đó một phần do các nhân viên quản trị hệ thống được kết nối với Internet ngày càng đề cao cảnh giác. Cũng theo CERT, những cuộc tấn công thời kỳ 1988-1989 chủ yếu đoán tên người sử dụng-mật khẩu (UserID-password) hoặc sử dụng một số lỗi của các chương trình và hệ điều hành (security hole) làm vô hiệu hệ thống bảo vệ, tuy nhiên các cuộc tấn công vào thời gian gần đây bao gồm cả các thao tác như giả mạo địa chỉ IP, theo dõi thông tin truyền qua mạng, chiếm các phiên làm việc từ xa (telnet hoặc rlogin).

Để vừa bảo đảm tính bảo mật của thông tin lại không làm giảm sự phát triển của việc trao đổi thông tin quảng bá trên toàn cầu thì một giải pháp tốt nhất là ***mã hoá*** thông tin. Có thể hiểu sơ lược mã hoá thông tin là che đi thông tin của mình làm cho kẻ tấn công nếu chặn được thông báo trên đường truyền thì cũng không thể đọc được và phải có một giao thức giữa người gửi và người nhận để có thể trao đổi thông tin, đó là các cơ chế mã và giải mã thông tin.

Ngày nay thì việc mã hoá đã trở nên phổ cập. Các công ty phần mềm lớn trên thế giới đều có nghiên cứu và xây dựng các công cụ, thuật toán mã hoá để áp dụng cho thực tế. Mỗi quốc gia hay tổ chức đều có những cơ chế mã hoá riêng để bảo vệ hệ thống thông tin của mình.

Một số vấn đề an toàn đối với nhiều mạng hiện nay:

- Một người dùng chuyển một thông báo điện tử cho một người sử dụng khác. Một bên thứ ba trên cùng mạng LAN này sử dụng một thiết bị nghe trộm gói để lấy thông báo và đọc các thông tin trong đó.

- Cũng trong tình huống trên bên thứ ba chặn thông báo, thay đổi các thành phần của nó và sau đó lại gửi cho người nhận. Người nhận không hề nghi ngờ gì trừ khi nhận ra thông báo đó là vô lý, và có thể

thực hiện vài hành động dựa trên các thành phần sai này đem lại lợi ích cho bên thứ ba.

- Người dùng log vào một server mà không sử dụng mật khẩu được mã hoá. Một người khác đang nge trộm trên đường truyền và bắt được mật khẩu logon của người dùng, sau đó có thể truy nhập thông tin trên server như người sử dụng.

- Một người quản trị hệ thống không hiểu về khía cạnh an toàn và yêu cầu của hệ thống và vô tình cho phép người dùng khác truy nhập vào thư mục chứa các thông tin hệ thống. Người dùng phát hiện ra họ có thể có được các thông tin hệ thống và có thể dùng nó phục vụ cho loợi ích của mình.

2. CÁC HỆ MÃ HOÁ CỔ ĐIỂN

2.1. HỆ MÃ HOÁ THAY THẾ (SUBSTITUTION CIPHER)

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

Có 4 kỹ thuật thay thế sau đây:

- *Thay thế đơn* (A simple substitution cipher): là hệ trong đó một ký tự của bản rõ được thay bằng một ký tự tương ứng trong bản mã. Một ánh xạ 1-1 từ bản rõ tới bản mã được sử dụng để mã hoá toàn bộ thông điệp.

- *Thay thế đồng âm* (A homophonic substitution cipher): giống như hệ thống mã hoá thay thế đơn, ngoại trừ một ký tự của bản rõ có thể được ánh xạ tới một trong số một vài ký tự của bản mã: sơ đồ ánh xạ 1-n (one-to-many). Ví dụ, “A” có thể tương ứng với 5, 13, 25, hoặc 56, “B” có thể tương ứng với 7, 19, 31, hoặc 42, v.v.

- *Thay thế đa mẫu tự* (A polyalphbetic substitution cipher): được tạo nên từ nhiều thuật toán mã hoá thay thế đơn. Ánh xạ 1-1 như trong trường hợp thay thế đơn, nhưng có thể thay đổi trong phạm vi một thông điệp. Ví dụ, có thể có năm thuật toán mã hoá đơn khác nhau được sử dụng; đặc biệt thuật toán mã hoá đơn được sử dụng thay đổi theo vị trí của mỗi ký tự trong bản rõ.

- *Thay thế đa sơ đồ* (A polygram substitution cipher): là thuật toán trong đó các khối ký tự được mã hoá theo nhóm. Đây là thuật toán tổng quát nhất, cho phép thay thế các nhóm ký tự của văn bản gốc. Ví dụ,

"ABA" có thể tương ứng với "RTQ", "ABB" có thể tương ứng với "SLL", v.v.

2.1.1. HỆ MÃ HOÁ CAESAR

Hệ mã hoá CAESAR là một hệ mã hoá thay thế đơn làm việc trên bảng chữ cái tiếng Anh 26 ký tự (A, B, ... , Z).

Trong hệ CAESAR và các hệ tương tự còn lại ta sử dụng các số tự nhiên thay cho các ký tự - đánh số các ký tự trong bảng chữ cái theo thứ tự: A là 0, B là 1, ... và Z là 25.

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

Các phép toán số học thực hiện theo modul 26. Có nghĩa là 26 đồng nhất với 0, 27 đồng nhất với 1, 28 đồng nhất với 2, ... Ví dụ:

$$2\times 17 + 5\times 9 = 79 = 1 + 3\times 26 = 1$$

Hệ CAESAR sử dụng thuật toán mã hoá trong đó mỗi ký tự được thay thế bởi một ký tự khác được xác định bằng cách dịch ký tự cần mã hoá sang phải k bước theo modul 26:

$$E_k(\alpha) = (\alpha + k) \text{ MOD } 26$$

với α là một ký tự, $0 \le k \le 26$, MOD là phép chia lấy phần dư.

Thuật toán giải mã tương ứng D_k là lùi lại k bước trong bảng chữ cái theo modul 26:

$$D_k(\alpha) = (\alpha - k) \text{ MOD } 26$$

Không gian khoá của hệ CEACAR bao gồm 26 số 0, 1, 2, ... 25.

Ví dụ: với k=3, A được thay bằng D, B được thay bằng E, ... , W được thay bằng Z, ... , X được thay bằng A, Y được thay bằng B, và Z được thay bằng C. Ta có:

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

Trong trường hợp này bản rõ "TRY AGIAN" được mã hoá thành "WUB DJDLQ", bản rõ "HELP ME" được mã hoá thành "KHOSPH". (Chú ý: các ký tự trống trong bản mã được bỏ đi để đảm bảo tính an toàn)

Thêm một vài ví dụ minh hoạ:

E_{25}(IBM) = HAL, E_6(MUPID) = SAVOJ,

E_3(HELP) = KHOS, E_1(HOME) = IPNF,

E_6(SAVOJ) = E_{20}(SAVOJ) = MUPID.

Hệ CAESAR là hệ mã hoá cũ và không an toàn vì không gian khoá của nó rất nhỏ, do đó có thể thám mã theo phương pháp vét cạn. Khoá giải mã có thể tính ngay ra được từ khoá mã hoá. Do chỉ có 26 khoá nên ta có thể thử lần lượt các khoá cho đến khi tìm được khoá đúng.

2.1.2. HỆ MÃ HOÁ VIGENERE

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

VIGENERE cũng giống như CAESAR, nhưng ở đây khoá được thay đổi theo từng bước. Hình vuông VIGENERE được sử dụng để mã hoá và giải mã.

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

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

Hình n. Hình vuông VIGENERE

Mỗi cột của hình vuông VIGENERE có thể xem như là một hệ CAESAR, với các khoá 0, 1, 2, ... , 25. Để mã hoá thì bản rõ được đọc từ các hàng và khoá được đọc từ các cột.

Ví dụ để mã hóa bản rõ PURPLE với từ khoá CRYPTO, đầu tiên ta tìm điểm giao nhau của hàng P và cột C, ta được R. Cứ như vậy ta được bản mã RLPEES. Ta sẽ thu được bản mã tương tự nếu ta thay đổi vai trò của hàng và cột trong khi mã hoá. Để giải mã bản mã RLPEES vừa mã hoá, ta nhìn vào hàng nào có chứa R trong cột C, theo cách này ta sẽ tìm được P. Và như vậy ta tìm được bản rõ là PURPLE.

Từ khoá thường được áp dụng một cách tuần hoàn. Nếu bản rõ dài hơn từ khoá thì từ khoá lại được bắt đầu lại từ đầu. Ví dụ, từ khoá CRYPTO được áp dụng với bản rõ có 15 ký tự là CRYPTO CRYPTO CRY.

Ta thấy rằng trong hệ mã hoá VIGENERE, với khoá có độ dài d thì sẽ có 26^d khoá hợp lệ. Vì vậy, chỉ cần với giá trị d nhỏ thì phương pháp thám mã vét cạn cũng đòi hỏi khá nhiều thời gian.

2.1.3. HỆ MÃ HOÁ HILL

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

Cả không gian bản rõ và bản mã đều là Σ^*, trong đó Σ là bản chữ cái tiếng Anh. Chúng ta sử dụng các số tự nhiên thay cho các ký tự và các phép toán số học được thực hiện theo modul 26 như đã nói ở phần trên.

Ta chọn một số nguyên (integer) d ≥ 2. Xét M là ma trận vuông d chiều. Các phần tử của M là các số nguyên từ 0 đến 25. Hơn nữa M phải là ma trận khả nghịch, tức là tồn tại M^{-1}. Ví dụ:

$$M = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \text{ và } M^{-1} = \begin{pmatrix} 15 & 17 \\ 20 & 9 \end{pmatrix}.$$

Để mã hoá, bộ d chữ cái của bản rõ được mã hoá cùng nhau. Trong các trường hợp sẽ xét dưới đây ta lấy d=2.

Quá trình mã hoá được thực hiện theo công thức:

$$MP = C$$

trong đó P và C được viết thành các vecter cột d chiều. Mỗi bộ d chữ cái của bản rõ được viết thành vecter P với các thành phần là các số biểu diễn các ký tự. Và C cũng thể hiện khối d ký tự của bản mã.

Còn khi giải mã ta phải dùng ma trận nghịch đảo M^{-1}:

$$P = CM^{-1}$$

Ví dụ, bản rõ "HELP" được viết thành hai vecter

$$P_1 = \begin{pmatrix} H \\ E \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \end{pmatrix} \text{ và } P_2 = \begin{pmatrix} L \\ P \end{pmatrix} = \begin{pmatrix} 11 \\ 15 \end{pmatrix}.$$

theo công thức mã hoá ta có

$$MP_1 = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix}\begin{pmatrix} 7 \\ 4 \end{pmatrix} = \begin{pmatrix} 33 \\ 34 \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \end{pmatrix} = \begin{pmatrix} H \\ I \end{pmatrix} = C_1 \text{ và}$$

$$MP_2 = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix}\begin{pmatrix} 11 \\ 15 \end{pmatrix} = \begin{pmatrix} 78 \\ 97 \end{pmatrix} = \begin{pmatrix} 0 \\ 19 \end{pmatrix} = \begin{pmatrix} A \\ T \end{pmatrix} = C_2$$

chúng ta thu được bản mã "HIAT".

2.2. HỆ MÃ HOÁ ĐỔI CHỖ (TRANSPOSITION CIPHER)

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

Ví dụ một hệ mã hoá đổi chỗ cột đơn giản, bản rõ được viết theo hàng ngang trên trang giấy với độ dài cố định, và bản mã được đọc theo hàng dọc (Hình 2).

Bản rõ: COMPUTER GRAPHICS MAY BE SLOW BUT AT LEAST IT'S EXPENSIVE

COMPUTERGR
APHICSMAYB
ESLOWBUTAT
LEASTITSEX
PENSIVE

Bản mã: CAELPOPSEEMHLANPIOSSUCWTITSBIUEMUTERATSGYAERBTX

Hình 2. Mã hoá thay đổi vị trí cột

Phương pháp này có các kỹ thuật sau:

- *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".

- *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

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

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.

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

Do vậy, bản rõ:

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

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

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

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

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

Vị trí đầu	Vị trí hoán vị	Từ	Mã hoá
1	3	G	O
2	5	R	P
3	1	O	G
4	4	U	U
5	2	P	R

Theo bảng trên, ký tự đầu trong khối 5 ký tự được chuyển tới vị trí thứ 3, ký tự thứ hai được chuyển tới vị trí thứ 5, ... Chẳng hạn từ gốc GROUP được mã hoá thành OPGUR.

Bằng cách đó, bản rõ "I LOVE BEETHOVENS MUSIC" sẽ được chuyển thành "OEIVLEHBTEESONVSCMIU".

Hệ mã ADFGV của Đức, được sử dụng trong suốt chiến tranh thế giới lần thứ I, đó là một hệ mã hoá đổi chỗ (có sử dụng thay thế

đơn giản). Nó được coi là một thuật toán mã hoá phức tạp vào thời ấy nhưng nó đã bị phá bởi Georges Painvin, một nhà thám mã người Pháp.

Mặc dù có rất nhiều hệ thống mã hoá sử dụng đổi chỗ, nhưng chúng rất rắc rối bởi vì nó đòi hỏi rất nhiều bộ nhớ.

3. CÁC VẤN ĐỀ VỀ MÃ HOÁ CHO MẠNG MÁY TÍNH

3.1. CÁC THUẬT NGỮ

1. *Hệ mật mã* là tập hợp các thuật toán và các thủ tục kết hợp để che dấu thông tin cũng như làm rõ nó.
2. *Mật mã học* nghiên cứu mật mã bởi các nhà mật mã học, người viết mật mã và các *nhà phân tích mã.*
3. *Mã hoá* là quá trình chuyển thông tin có thể đọc gọi là *bản rõ* thành thông tin không thể đọc gọi là *bản mã.*
4. *Giải mã* là quá trình chuyển ngược lại thông tin được mã hoá thành bản rõ.
5. *Thuật toán mã hoá* là các thủ tục tính toán sử dụng để che dấu và làm rõ thông tin. Thuật toán càng phức tạp thì bản mã càng an toàn.
6. Một *khoá* là một giá trị làm cho thuật toán mã hoá chạy theo cách riêng biệt và sinh ra bản rõ riêng biệt tuỳ theo khoá. Khoá càng lớn thì bản mã kết quả càng an toàn. Kích thước của khoá được đo bằng bit. Phạm vi các giá trị có thể có của khoá được gọi là *không gian khoá.*
7. *Phân tích mã* là quá trình hay nghệ thuật phân tích hệ mật mã hoặc kiểm tra tính toàn vẹn của nó hoặc phá nó vì những lý do bí mật.
8. Một *kẻ tấn công* là một người (hay hệ thống) thực hiện phân tích mã để làm hại hệ thống. Những kẻ tấn công là những kẻ thọc mũi vào chuyện người khác, các tay hacker, những kẻ nghe trộm hay những các tên đáng ngờ khác, và họ làm những việc thường gọi là cracking

3.2. ĐỊNH NGHĨA HỆ MẬT MÃ.

1. Hệ mật mã: là một hệ bao gồm 5 thành phần (P, C, K, E, D) thoả mãn các tính chất sau

P (Plaintext) là tập hợp hữu hạn các bản rõ có thể.

C (Ciphertext) là tập hợp hữu hạn các bản mã có thể.

K (Key) là tập hợp các bản khoá có thể.

E (Encrytion) là tập hợp các qui tắc mã hoá có thể.

D (Decrytion) là tập hợp các qui tắc giải mã có thể.

Chúng ta đã biết một thông báo thường được tổ chức dưới dạng bản rõ. Người gửi sẽ làm nhiệm vụ mã hoá bản rõ, kết quả thu được gọi là bản mã. Bản mã này được gửi đi trên một đường truyền tới người nhận sau khi nhận được bản mã người nhận giải mã nó để tìm hiểu nội dung.

Dễ dàng thấy được công việc trên khi sử dụng định nghĩa hệ mật mã :

$$E_K(P) = C \text{ và } D_K(C) = P$$

3.3. NHỮNG YÊU CẦU ĐỐI VỚI HỆ MẬT MÃ

Cung cấp một mức cao về độ tin cậy, tính toàn vẹn, sự không từ chối và sự xác thực.

- *Độ tin cậy*: cung cấp sự bí mật cho các thông báo và dữ liệu được lưu bằng việc che dấu thông tin sử dụng các kỹ thuật mã hóa.
- *Tính toàn vẹn*: cung cấp sự bảo đảm với tất cả các bên rằng thông báo còn lại không thay đổi từ khi tạo ra cho đến khi người nhận mở nó.
- *Tính không từ chối*: có thể cung cấp một cách xác nhận rằng tài liệu đã đến từ ai đó ngay cả khi họ cố gắng từ chối nó.
- *Tính xác thực*: cung cấp hai dịch vụ: đầu tiên là nhận dạng nguồn gốc của một thông báo và cung cấp một vài sự bảo đảm rằng nó là đúng sự thực. Thứ hai là kiểm tra đặc tính của người đang logon một hệ thống và sau đó tiếp tục kiểm tra đặc tính của họ trong trường hợp ai đó cố gắng đột nhiên kết nối và giả dạng là người sử dụng

3.4. CÁC PHƯƠNG PHÁP MÃ HOÁ

3.4.1. MÃ HOÁ ĐỐI XỨNG KHOÁ BÍ MẬT

- Định nghĩa

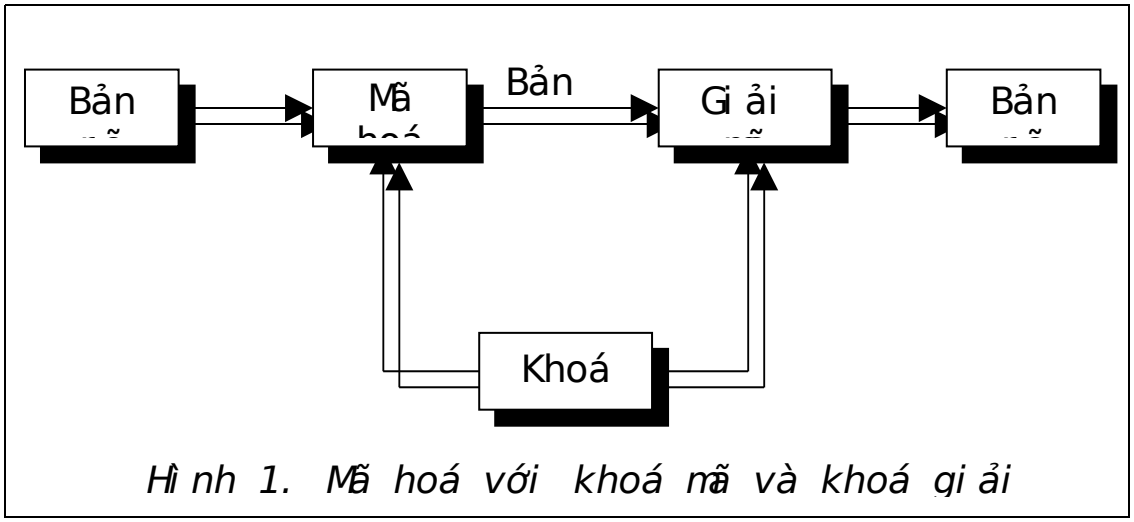
Thuật toán đối xứng hay còn gọi thuật toán mã hoá cổ điển là thuật toán mà tại đó khoá mã hoá có thể tính toán ra được từ khoá giải mã. Trong rất nhiều trường hợp, khoá mã hoá và khoá giải mã là giống

nhau. Thuật toán này còn có nhiều tên gọi khác như thuật toán khoá bí mật, thuật toán khoá đơn giản, thuật toán một khoá. Thuật toán này yêu cầu người gửi và người nhận phải thoả thuận một khoá trước khi thông báo được gửi đi, và khoá này phải được cất giữ bí mật. Độ an toàn của thuật toán này vẫn phụ thuộc và khoá, nếu để lộ ra khoá này nghĩa là bất kỳ người nào cũng có thể mã hoá và giải mã thông báo trong hệ thống mã hoá.

Sự mã hoá và giải mã của thuật toán đối xứng biểu thị bởi :

$$E_K(P) = C$$

$$D_K(C) = P$$



Hình 1. Mã hoá với khoá mã và khoá giải

Mã hoá và giải mã với một khoá

Trong hình vẽ trên thì :

K1có thể trùng K2, hoặc

K1 có thể tính toán từ K2, hoặc

K2 có thể tính toán từ K1.

- Nơi ứng dụng:
- Sử dụng trong môi trường mà khoá đơn dễ dàng được chuyển như là trong cùng một văn phòng. Cũng dùng để mã hoá thông tin để lưu trữ trên đĩa.
- Các vấn đề đối với phương pháp mã hoá này:

1.Các phương mã hoá cổ điển đòi hỏi người mã hoá và người giải mã phải cùng chung một khoá. Khi đó khoá phải được giữ bí mật tuyệt đối, do vậy ta dễ dàng xác định một khoá nếu biết khoá kia.

2.Hệ mã hoá đối xứng không bảo vệ được sự an toàn nếu có xác suất cao khoá người gửi bị lộ. Trong hệ khoá phải được gửi đi trên kênh an toàn nếu kẻ địch tấn công trên kênh này có thể phát hiện ra khoá.

3.Vấn đề quản lý và phân phối khoá là khó khăn và phức tạp khi sử dụng hệ mã hoá cổ điển. Người gửi và người nhận luôn luôn thông nhất với nhau về vấn đề khoá. Việc thay đổi khoá là rất khó và dễ bị lộ.

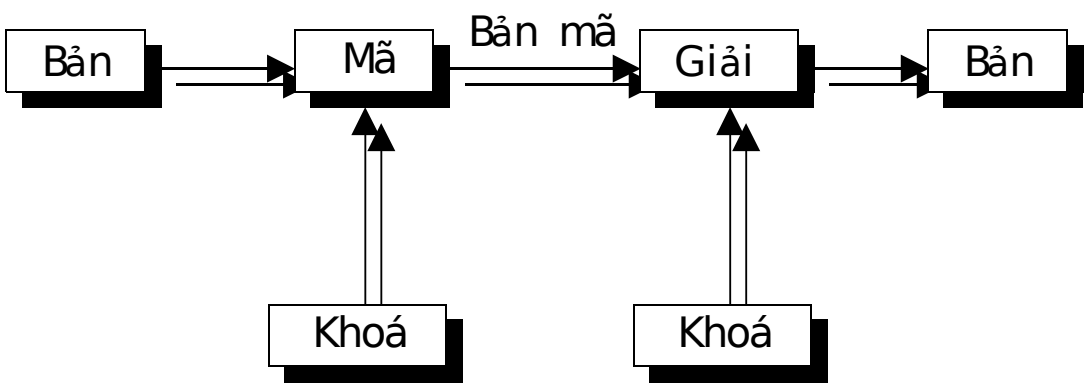
4.Khuynh hướng cung cấp khoá dài mà nó phải được thay đổi thường xuyên cho mọi người trong khi vẫn duy trì cả tính an toàn lẫn hiệu quả chi phí sẽ cản trở rất nhiều tới việc phát triển hệ mật mã cổ điển.

3.4.2. MÃ HOÁ PHI ĐỐI XỨNG KHOÁ CÔNG KHAI

- Định nghĩa

Vào những năm 1970 Diffie và Hellman đã phát minh ra một hệ mã hoá mới được gọi là *hệ mã hoá công khai* hay *hệ mã hoá phi đối xứng.*

Thuật toán mã hoá công khai là khác biệt so với thuật toán đối xứng. Chúng được thiết kế sao cho *khoá* sử dụng vào việc mã hoá là khác so với *khoá giải mã.* Hơn nữa khoá giải mã không thể tính toán được từ khoá mã hoá. Chúng được gọi với tên hệ thống mã hoá công khai bởi vì khoá để mã hoá có thể công khai, một người bất kỳ có thể sử dụng khoá công khai để mã hoá thông báo, nhưng chỉ một vài người có đúng khoá giải mã thì mới có khả năng giải mã. Trong nhiều hệ thống, khoá mã hoá gọi là *khoá công khai* (public key), khoá giải mã thường được gọi là *khoá riêng* (private key).



Hình 1. Mã hoá với khoá mã và khoá giải

K1 không thể trùng K2, hoặc

K2 không thể tính toán từ K1.

Đặc trưng nổi bật của hệ mã hoá công khai là cả khoá công khai (public key) và bản tin mã hoá (ciphertext) đều có thể gửi đi trên một kênh thông tin không an toàn.

- Nơi ứng dụng: Sử dụng chủ yếu trên các mạng công khai như Internet khi mà khoá chuyển tương đối khó khăn.
- Diffie và Hellman đã xác đinh rõ các điều kiện của một hệ mã hoá công khai như sau:

1. Việc tính toán ra cặp khoá công khai K_B và bí mật k_B dựa trên cơ sở các điều kiện ban đầu phải được thực hiện một cách dễ dàng, nghĩa là thực hiện trong thời gian đa thức.

2. Người gửi A có được khoá công khai của người nhận B và có bản tin P cần gửi đi thì có thể dễ dàng tạo ra được bản mã C.

$$C = E_{KB}(P) = E_B(P)$$

Công việc này cũng trong thời gian đa thức.

3. Người nhận B khi nhận được bản tin mã hóa C với khoá bí mật k_B thì có thể giải mã bản tin trong thời gian đa thức.

$$P = D_{kB}(C) = D_B[E_B(M)]$$

4. Nếu kẻ địch biết khoá công khai K_B cố gắng tính toán khoá bí mật thì khi đó chúng phải đương đầu với trường hợp nan giải, trường hợp này đòi hỏi nhiều yêu cầu không khả thi về thời gian.

5. Nếu kẻ địch biết được cặp (K_B,C) và cố gắng tính toán ra bản rõ P thì giải quyết bài toán khó với số phép thử là vô cùng lớn, do đó không khả thi.

3.5. CÁC CÁCH PHÂN TÍCH MÃ

Các thuật toán cho phần lớn các hệ mật mã là nổi tiếng nên chúng ta giả sử rằng những kẻ phân tích mã đã có thuật toán trong tay khi bắt đầu tấn công. Trong phần lớn các hệ mật mã, thuật toán để phân phối cho tất cả người sử dụng và sức mạnh của hệ thống nằm trong khoá cũng như phụ thuộc vào thuật toán mã hoá dữ liệu tốt như

thế nào. Và độ dài của khoá mã quyết định bản mã kết quả được mã tốt như thế nào và sẽ bảo vệ chống lại các cuộc tấn công brute-force. Tấn công brute-force là cách trong đó mọi khoá có thể được thử dùng để giải mã.

Nhiều nhà viết mật mã tin rằng các cuộc tấn công brute-force không thể thực hiện được khi khoá dài được sử dụng, thậm chí khi khả năng của máy tính đang lên. Tấn công brute-force đối với bản mã phải mã hoá với một khoá lớn (trên 100 bít) có thể mất hàng triệu hoặc hàng tỉ năm ngay cả khi với mạng máy tính mạnh hơn nữa việc thêm một bít đơn có thể làm tăng gấp đôi giá của việc phân tích bằng brute-force.

Tuy nhiên vẫn tồn tại một điểm yếu trong hệ thống trừ một vài khoá, làm giảm số các khoá cần được kiểm tra. Ví dụ, kẻ phân tích mã có thể khám phá ra rằng một thuật toán sinh ra các số ngẫu nhiên nhưng thực tế có một vài mẫu được lặp lại. Điểm yếu này của hệ thống có thể cung cấp một con đường để khám phá hệ thống.

Có một vài phương pháp chung để phân tích, dưới đây là danh sách theo thứ tự khả năng của từng phương pháp. Mỗi phương pháp trong số chúng giả sử rằng kẻ phân tích mã hoàn toàn có hiểu biết về thuật toán mã hoá được sử dụng.

1. *Chỉ có bản mã.* Trong trường hợp này, người phân tích chỉ có một vài bản tin của bản mã, tất cả trong số chúng đều đã được mã hoá và cùng sử dụng chung một thuật toán. Công việc của người phân tích là tìm lại được bản rõ của nhiều bản mã có thể hoặc tốt hơn nữa là suy luận ra được khoá sử dụng mã hoá, và sử dụng để giải mã những bản mã khác với cùng khoá này.

Giả thiết : $C_1 = E_k(P_1)$, $C_2 = E_k(P_2)$, . . . $C_i = E_k(P_i)$

Suy luận : Mỗi P_1, P_2, . . P_i, k hoặc thuật toán kết luận P_{i+1} từ $C_{i+1} = E_k(P_{i+1})$

2. *Biết bản rõ.* Người phân tích không chỉ truy cập được một vài bản mã mặt khác còn biết được bản rõ. Công việc là suy luận ra khoá để sử dụng giải mã hoặc thuật toán giải mã để giải mã cho bất kỳ bản mã nào khác với cùng khoá như vậy.

Giả thiết : P_1, $C_1 = E_k(P_1)$, P2, C2= $E_k(P_2)$, . . . Pi, $C_i = E_k(P_i)$

Suy luận : Mỗi k hoặc thuật toán kết luận P_{i+1} từ $C_{i+1} = E_k(P_{i+1})$

3. *Lựa chọn bản rõ.* Người phân tích không chỉ truy cập được bản mã và kết hợp bản rõ cho một vài bản tin, nhưng mặt khác lựa chọn bản rõ đã mã hoá. Phương pháp này tỏ ra có khả năng hơn phương pháp

biết bản rõ bởi vì người phân tích có thể chọn cụ thể khối bản rõ cho mã hoá, một điều khác có thể là sản lượng thông tin về khoá nhiều hơn.

Giả thiết : P_1, $C_1 = E_k(P_1)$, P2, $C2= E_k(P_2)$, . . . Pi, $C_i = E_k(P_i)$ tại đây người phân tích chọn P_1, P_2,. . . P_i

Suy luận : Mỗi k hoặc thuật toán kết luận P_{i+1} từ $C_{i+1} = E_k(P_{i+1})$

4. *Lựa chọn bản rõ thích hợp.* Đây là trường hợp đặc biệt của lựa chọn bản rõ. Không chỉ có thể lựa chọn bản rõ đã mã hoá, nhưng họ còn có thể sửa đổi sự lựa chọn cơ bản kết quả của sự mã hoá lần trước. Trong trường lựa chọn bản mã người phân tích có thể đã chọn một khối lớn bản rõ đã mã hoá, nhưng trong trường hợp này có thể chọn một khối nhỏ hơn và chọn căn cứ khác trên kết quả của lần đầu tiên.

Ví dụ tốt nhất về trường hợp biết bản rõ là những file được tạo ra bởi các từ khác nhau trong đó chứa các mã định dạng được ẩn và header file. Các tài liệu cũng chứa tên công ty và địa chỉ, bản quyền, và nhiều thông tin khác mà các nhà phân tích có thể lấy được một cách dễ dàng. Thực tế là, rất nhiều tài liệu được sử dụng trong thương mại điện tử có header chuẩn được sử dụng để định danh tài liệu cho các máy tính khác. Các nhà phân tích có thể tìm ra khoá bằng việc phân tích thuật toán đã mã bản rõ đã biết như thế nào.

Dưới đây là một vài kỹ thuật được các nhà phân tích để tấn công bản mã.

- *Differential cryptanalysis;* kỹ thuật này sử dụng một quá trình lặp để ước lượng mã được tạo ra sử dụng một thuật toán lặp khối (như DES). Liên kết bản rõ được mã hoá dưới cùng một khoá. Sự khác biệt được phân tích và các khoá có thể được xác định thông qua số các lần lặp. Kỹ thuật này được sử dụng thành công chống lại DES và FEAL-4.
- *Linear cryptanalysis*: kỹ thuật này cũng được sử dụng thành công để chống lại DES và FEAL-4. Các cặp bản rõ và bản mã kết quả được phân tích và một kỹ thuật xấp xỉ tuyến tính được sử dụng để xác định hoạt động của mã khối.
- *Algebraic attacks:*ký thuật này khám phá cấu trúc toán học trong các mật mã khối. Nếu cấu trúc tồn tại thì việc mã hoá đơn với một khoá có thể sinh ra các kết quả tương tự như việc

mã hoá đôi với hai khoá khác nhau. Kẻ phân tích sẽ có được ưu thế ở yếu điểm này.

Nói chung những nhà phân tích mã cần có thời gian và tài nguyên. Điều này làm cho các nhà phân tích gặp khó khăn nhất là khi các thuật toán và ký thuật mã hoá tốt hơn nhờ các tiến bộ kỹ thuật.

Dù rằng các cuộc tấn công DES là thành công nhưng các cuộc tấn công này không dễ dàng và rẻ. Phương pháp nhanh nhất là tấn công brute-force đã mất 3.5 giờ trên máy tính hàng triệu đôla. Differential cryptanalysis được sử dụng để tấn công, trong đó 247 lần mã được thực hiện trên một bản rõ được chọn, cuộc tấn công quá khó sẽ không được đề cập trong thực tế. Trong một cuộc tấn công khác Linear cryptanalysis được dùng để tìm ra khoá DES trong 50 ngày, sử dụng 12 computer workstation.

4. MỘT SỐ THUẬT TOÁN MÃ HOÁ CƠ BẢN

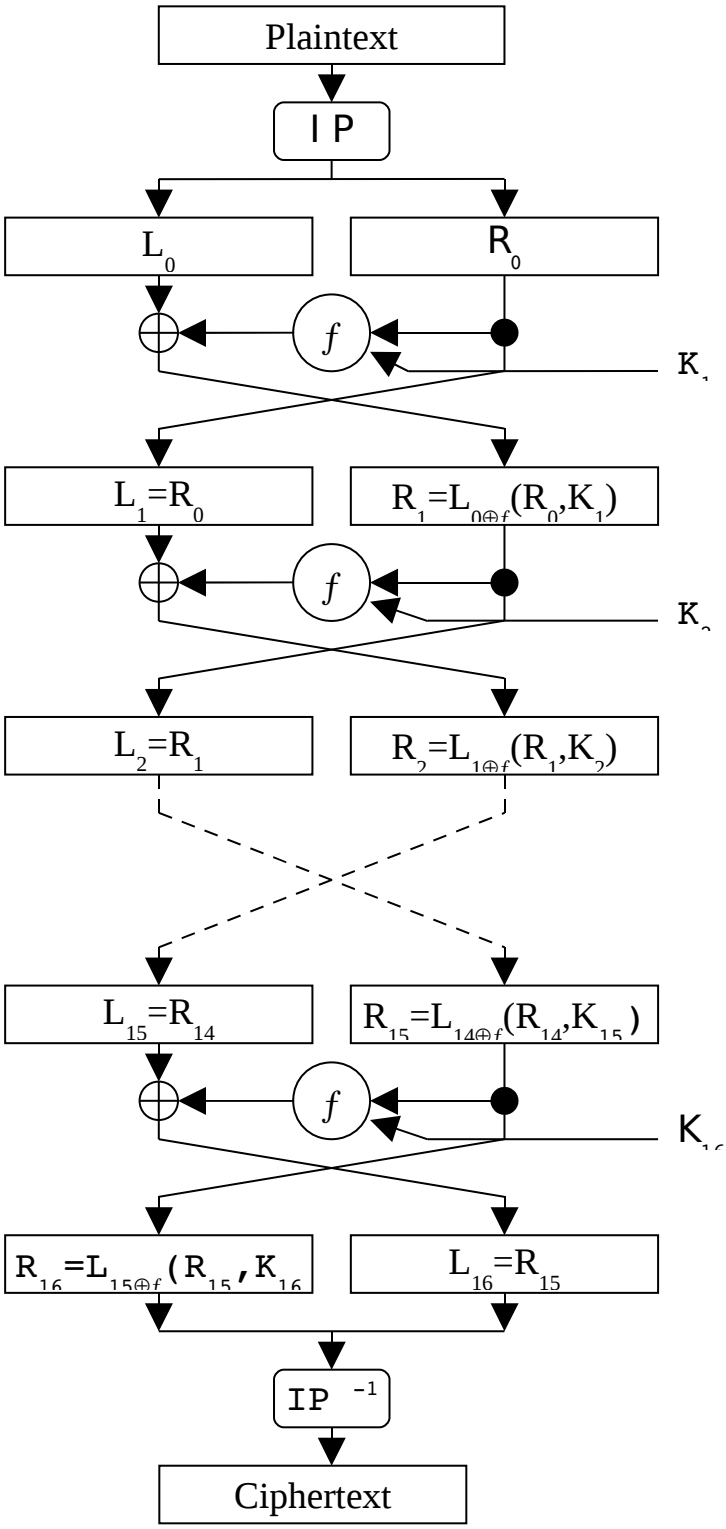
4.1. CHUẨN MÃ HOÁ DỮ LIỆU DES

DES là thuật toán mã hoá khối (block algrithm), nó mã hoá một khối dữ liệu 64 bíts. Một khối bản rõ 64 bít đưa vào vào thực hiện, sau khi mã hoá dữ liệu ra là một khối bản mã 64 bít. Cả mã hoá và giải mã đều sử dụng cùng một thuật toán và khoá.

Khoá mã có độ dài 64 bít, trong đó có 8 bít chẵn lẻ được sử dụng để kiểm soát lỗi. Các bít chẵn lẻ nằm ở các vị trí 8, 16, 24, ... , 64. Tức là cứ 8 bít thì có 1 bít kiểm soát lỗi, bít này qui định số bít có giá trị “1” của khối 8 bít đó là chẵn hay lẻ.

Nền tảng để xây dựng khối của DES là sự kết hợp đơn giản của các kỹ thuật thay thế và hoán vị bản rõ dựa trên khoá, đó là các vòng lặp. DES sử dụng 16 vòng lặp áp dụng cùng một kiểu kết hợp các kỹ thuật trên khối bản rõ (Hình 1).

Thuật toán chỉ sử dụng các phép toán số học và logic thông thường trên các số 64 bít, vì vậy nó dễ dàng thực hiện vào những năm 1970 trong điều kiện về công nghệ phần cứng lúc bấy giờ. Ban đầu, sự thực hiện các phần mềm kiểu này rất thô sơ, nhưng hiện tại thì việc đó đã tốt hơn, và với đặc tính lặp đi lặp lại của thuật toán đã tạo nên ý tưởng sử dụng chíp với mục đích đặc biệt này.


Plaintext
IP
L0
R0
f
K1
L1=R0
R1=L0⊕f(R0,K1)
f
K2
L2=R1
R2=L1⊕f(R1,K2)
L15=R14
R15=L14⊕f(R14,K15)
f
K16
R16=L15⊕f(R15,K16)
L16=R15
IP -1
Ciphertext

Hình 2. DES

4.1.1. MÔ TẢ THUẬT TOÁN

DES thực hiện trên từng khối 64 bíts của bản rõ. Sau khi thực hiện hoán vị khởi đầu, khối dữ liệu được chia làm hai nửa trái và phải, mỗi nửa 32 bít. Tiếp đó, có 16 vòng lặp giống hệt nhau được thực hiện, được gọi là các hàm f, trong đó dữ liệu được kết hợp với khoá. Sau 16 vòng lặp, hai nửa trái và phải được kết hợp và hoán vị cuối cùng (hoán vị ngược) sẽ kết thúc thuật toán.

Mỗi vòng lặp của DES thực hiện theo các bước sau (Hình 2):

1. Khối bản rõ 64 bít được hoán vị (hoán vị khởi đầu) để thay đổi thứ tự của các bít.

2. Tiếp theo, bản rõ được chia làm hai nửa trái và phải, mỗi nửa 32 bít.

3. Khoá được chia làm hai nửa, mỗi nửa 28 bít.

4. Các nửa của khoá được dịch trái, số bít được dịch 1 hay hai tuỳ thuộc vào vòng đó. Sau các nửa đó được ghép lại, hoán vị và lựa chọn ra 48 bít.

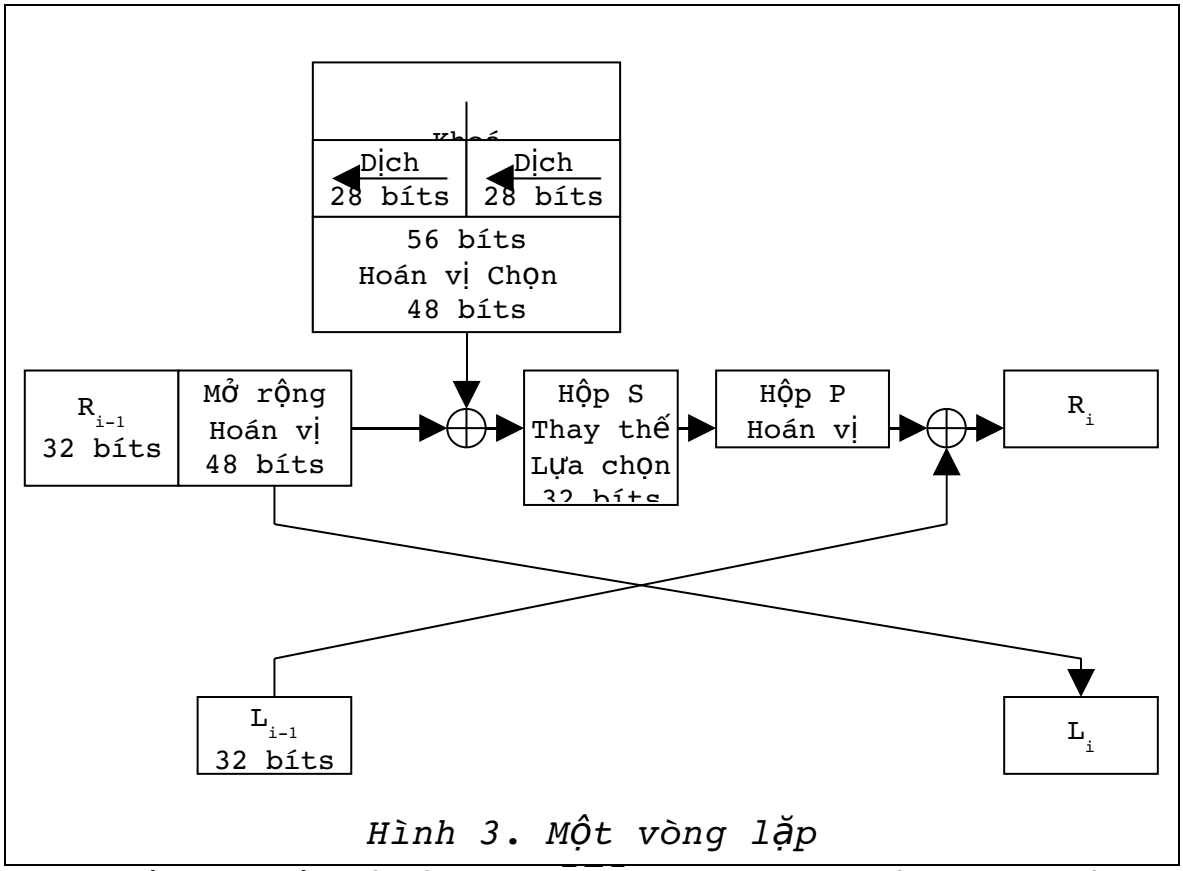
5. Khối 32 bít bản rõ bên phải được mở rộng thành khối 48 bít để nó có thể XOR được với 48 bít khoá. Một phép hoán vị khác cũng được thực hiện trong bước này.

6. Kết quả của bước 3 và 5 (bản rõ và khoá) được XOR với nhau.

7. Kết quả của bước 6 được chuyển thành 32 bít bằng cách sử dụng một hàm thay thế và lựa chọn.

8. Kết quả của bước 7 được XOR với nửa trái 32 bít của khối bản rõ 64 bít đã được tạo ra ở bước 2.

9. Kết quả của bước 8 trở thành nửa phải mới và nửa phải cũ được tạo ở bước 2 trở thành nửa trái mới.



Hình 3. Một vòng lặp

Nếu B_i là kết quả của vòng thứ i, L_i và R_i là hai nửa trái và phải của B_i, K_i là khoá 48 bíts của vòng thứ i, và f là hàm thực hiện thay thế, hoán vị và XOR với khoá, ta có biểu diễn của một vòng sẽ như sau:

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

$$R_i=L_{i-1} \text{ XOR } f(R_{i-1},K_i)$$

4.1.2. HOÁN VỊ KHỞI ĐẦU (THE INITIAL PERMUTATION)

Hoán vị khởi đầu đổi chỗ khối dữ liệu vào, sự hoán được mô tả trong Bảng 1. Bảng này, và tất cả các bảng khác sau này, được đọc từ trái qua phải, từ trên xuống dưới. Ví dụ, hoán vị khởi đầu chuyển bít 1 thành bít 58, bít 2 thành bít 50, bít 3 thành bít 42, ...

Bảng 1. Hoán vị khởi đầu.

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

Hoán vị khởi đầu và tương ứng là hoán vị cuối cùng không làm ảnh hưởng đến sự an toàn của DES.

4.1.3. KHOÁ CHUYỂN ĐỔI (THE KEY TRANSFORMATION)

Đầu tiên, khoá 64 bít được giảm xuống thành một khoá 56 bít bằng cách bỏ qua 8 bít chẵn lẻ. Sự loại bỏ được thực hiện theo Bảng 2.

Bảng 2. Khoá chuyển đổi

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

Các bít chẵn lẻ này có thể được sử dụng để đảm bảo rằng không có lỗi nào xảy ra khi đưa khoá vào. Sau khi khoá 56 bít được trích ra, một khoá khác 48 bít được sinh ra cho mỗi vòng của DES. Những khoá này, k_i, được xác định bằng cách:

Đầu tiên, khoá 56 bít được chia làm hai phần mỗi phần 28 bít. Sau đó, các phần này được dịch trái một hoặc hai bít, phụ thuộc vào vòng đó. Số bít được dịch được cho trong Bảng 3.

Bảng 3.

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

Sau khi được dịch, 48 bít được lựa chọn ra từ 56 bít. Bởi vì sự thực hiện này đổi chỗ thứ tự các bít như là sự lựa chọn một tập con các bít, nó được gọi là hoán vị nén (compression permutation), hoặc hoán vị lựa chọn (permuted choice). Sự thực hiện này cung cấp một tập hợp các bít cùng cỡ với đầu ra của hoán vị mở rộng. Bảng 4 định nghĩa hoán vị nén (cũng gọi là hoán vị lựa chọn). Ví dụ, bít ở vị trí 33 của khoá dịch được chuyển tới vị trí 35 của đầu ra, và bít ở vị trí 18 của khoá dịch bị bỏ qua.

Bảng 4. Hoán vị nén

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

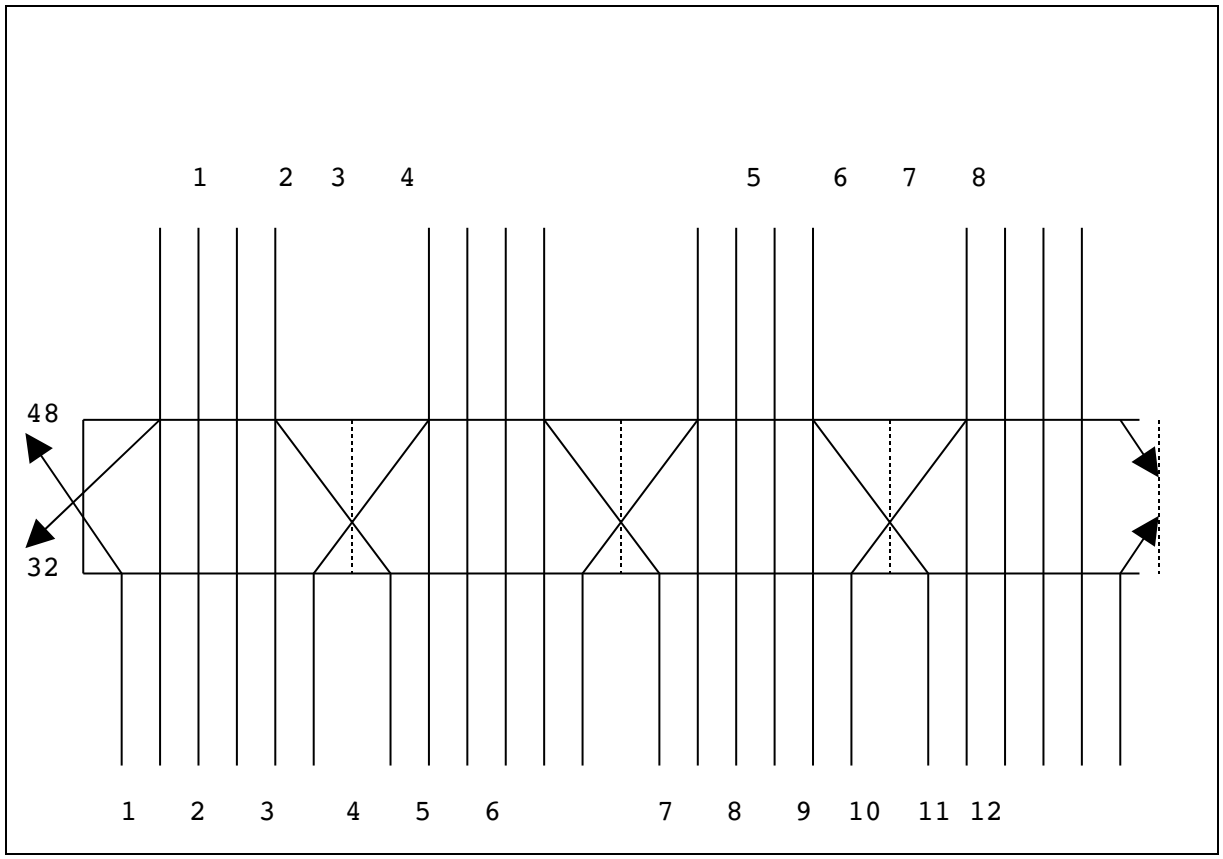
4.1.4. HOÁN VỊ MỞ RỘNG (EXPANSION PERMUTATION)

Ở thao tác này, nửa phải của dữ liệu, R_i, được mở rộng từ 32 bíts thành 48 bíts. Bởi vì sự thực hiện này thay đổi thứ tự của các bít bằng cách lặp lại một bít nào đó, nó được hiểu như là một sự hoán vị mở rộng.

Hình 3 định nghĩa hoán vị mở rộng - hộp E. Với mỗi bộ 4 bít của khối dữ liệu vào, bít đầu tiên và bít thứ tư tương ứng với 2 bít của khối dữ liệu ra, trong khi bít thứ hai và bít thứ ba tương ứng với một bít của khối dữ liệu ra. Bảng 5 mô tả vị trí của các bít trong khối dữ liệu ra theo khối dữ liệu vào. Ví dụ, bít ở vị trí thứ 3 của khối dữ liệu vào được chuyển tới vị trí thứ 4 trong khối dữ liệu ra. Và bít ở vị trí 21 của khối dữ liệu vào được chuyển tới vị trí 30 và 32 trong khối dữ liệu ra.

Bảng 5. Hộp E

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	12	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



Hình 3. Hoán vị mở

Mặc dù khối dữ liệu ra rộng hơn khối dữ liệu vào, nhưng một khối dữ liệu vào chỉ có duy nhất một khối dữ liệu ra.

4.1.5. HỘP THAY THẾ S (S-BOX SUBSTITUTION)

Sau khi khoá được nén, nó được XOR với khối mở rộng, 48 bít kết quả được chuyển sang giai đoạn thay thế. Sự thay thế được thực hiện bởi 8 hộp thay thế (substitution boxes, S-boxes). Khối 48 bít được chia thành 8 khối 6 bít. Mỗi khối được thực hiện trên một hộp S riêng biệt (separate S-box): khối 1 được thực hiện trên hộp S 1, khối 2 được thực hiện trên hộp S 2, v.v...

Mỗi hộp S là một bảng gồm 4 hàng và 16 cột. Mỗi phần tử của hộp là một số 4 bít. Sáu bít vào hộp S sẽ xác định số hàng và số cột để tìm kết quả ra. Bảng 6 biểu diễn 8 hộp S.

Những bít vào xác định một phần tử trong hộp S một cách riêng biệt. Sáu bít vào của hộp được ký hiệu là b1, b2, b3, b4, b5 và b6. Bít b1 và b6 được kết hợp thành một số 2 bít, nhận giá trị từ 0 đến 3, tương ứng với một hàng trong bảng. Bốn bít ở giữa, từ b2 tới b5, được kết hợp thành một số 4 bít, nhận giá trị từ 0 đến 15, tương ứng với một cột trong bảng.

Ví dụ, giả sử ta đưa giữ liệu vào hộp S thứ 6 (bít 31 tới bít 36 của hàm XOR) là 110010. Bít đầu tiên và bít cuối cùng kết hợp thành 10, tương ứng với hàng thứ 3 của hộp S thứ 6. Bốn bít giữa kết hợp thành 1001, tương ứng với cột thứ 10 của hộp S thứ 6. Phần tử hàng 3 cột 9 của hộp S thứ 6 là 0. Giá trị 0000 được thay thế cho 110010.

Đây là một bước khó hiểu trong thuật toán. Tất cả các bước khác đều đơn giản và dễ phân tích. Các hộp S thì không, chúng đem lại cho DES tất cả sự an toàn.

Kết quả của sự thay thế là 8 khối 4 bít được sinh ra, và chúng được kết hợp lại thành một khối 32 bít. Khối này được chuyển tới bước tiếp theo: hộp hoán vị P (P-box permutation).

Bảng 6. Hộp S

Hộp S thứ nhất

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

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

4.1.6. HỘP HOÁN VỊ P (THE P-BOX PERMUTATION)

Khối dữ liệu 32 bít ra của hộp thay thế S được hoán vị tiếp trong hộp P. Sự hoán vị này ánh xạ mỗi bít dữ liệu vào tới một vị trí trong khối dữ liệu ra; không bít nào được sử dụng hai lần và cũng không bít nào bị bỏ qua. Nó được gọi là hoán vị trực tiếp (straight permutation). Bảng 7 cho ta vị trí của mỗi bít cần chuyển. Ví dụ, bít 4 chuyển tới bít 21, trong khi bít 32 chuyển tới bít 4.

Bảng 7. Hộp hoán vị 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

Cuối cùng, kết quả của hộp hoá vị P được XOR với nửa trái của khối 64 bít khởi đầu. Sau đó, nửa trái và phải được chuyển đổi cho nhau và một vòng mới được tiếp tục.

4.1.7. HOÁN VỊ CUỐI CÙNG

Hoán vị cuối cùng là nghịch đảo của hoán vị khởi đầu, và nó được mô tả trong Bảng 8. Chú ý rằng nửa trái và nửa phải không được tráo đổi sau vòng cuối cùng của DES; thay vào đó khối nối $R_{16}L_{16}$ được sử dụng như khối dữ liệu ra của hoán vị cuối cùng. Không có gì đưa ra ở đây; tráo đổi các nửa và dịch vòng hoán vị sẽ cho chính xác như kết quả trước; điều đó có nghĩa là thuật toán có thể được sử dụng cho cả mã hoá và giải mã.

Bảng 8. Hoán vị cuối cùng

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

4.1.8. GIẢI MÃ DES

Sau khi thay đổi, hoán vị, XOR, và dịch vòng, bạn có thể nghĩ rằng thuật toán giải mã hoàn toàn khác và phức tạp, khó hiểu như thuật toán mã hoá. Trái lại, sự hoạt động được lựa chọn để đưa ra một đặc tính hữu ích: cùng thuật toán làm việc cho cả mã hoá và giải mã.

Với DES, có thể sử dụng cùng chức năng để giải mã hoặc mã hoá một khối. Chỉ có sự khác nhau đó là các khoá phải được sử dụng theo thứ tự ngược lại. Nghĩa là, nếu các khoá mã hoá cho mỗi vòng là $k_1, k_2, k_3, ..., k_{15}, k_{16}$ thì các khoá giải là $k_{16}, k_{15}, ..., k_3, k_2, k_1$. Thuật toán dùng để sinh khoá được sử dụng cho mỗi vòng theo kiểu vòng quanh. Khoá được dịch phải, và số ở những vị trí được dịch được tính từ cuối của bảng lên, thay vì từ trên xuống.

4.1.9. PHẦN CỨNG VÀ PHẦN MỀM THỰC HIỆN DES

Một phần mềm DES trên máy tính lớn IBM 3090 có thể thực hiện 32.000 phép tính mã hoá trong một giây. Với máy vi tính thì tốc độ thấp hơn. Bảng 9 đưa ra kết quả thực tế và sự đánh giá cho bộ xử lý của Intel và Motorola.

Bảng 9. Tốc độ của DES trên các bộ vi xử lý khác nhau

Bộ vi xử lý	Tốc độ (Mhz)	BUS (bíts)	Khối DES (/giây)
8088	4.7	8	370
68000	7.6	16	900
80286	6.0	16	1.1000
68020	16.0	32	3.500
68030	16.0	32	3.900
80286	25.0	16	5.000
68030	50.0	32	9.600
68040	25.0	32	16.000
68040	40.0	32	23..200
80486	33.0	32	40.600

(Chú ý : Phần mềm này được viết trên C và Assembler, và có thể mua được từ Utimaco-Belgium, Interleuvenlaan 62A, B-300 leuven, Belgium. Cỡ mã xấp xỉ 64K. ANSI C thực hiện chậm hơn khoảng 20%.)

4.2. THUẬT TOÁN MÃ HOÁ RSA (PUBLIC-KEY ALGORITHM)

4.2.1. KHÁI NIỆM HỆ MẬT MÃ RSA

Khái niệm hệ mật mã RSA đã được ra đời năm 1976 bởi các tác giả R.Rivets, A.Shamir, và L.Adleman. Hệ mã hoá này dựa trên cơ sở của hai bài toán :

+ Bài toán Logarithm rời rạc (Discrete logarith)

+ Bài toán phân tích thành thừa số.

Trong hệ mã hoá RSA các bản rõ, các bản mã và các khoá (public key và private key) là thuộc tập số nguyên $Z_N = \{1, \ldots, N-1\}$. Trong đó tập Z_N với $N=p\times q$ là các số nguyên tố khác nhau cùng với phép cộng và phép nhân Modulo N tạo ra modulo số học N.

Khoá mã hoá E_{KB} là cặp số nguyên (N,K_B) và khoá giải mã D_{kb} là cặp số nguyên (N,k_B), các số là rất lớn, số N có thể lên tới hàng trăm chữ số.

Các phương pháp mã hoá và giải mã là rất dễ dàng.

Công việc mã hoá là sự biến đổi bản rõ P (Plaintext) thành bản mã C (Ciphertext) dựa trên cặp khoá công khai K_B và bản rõ P theo công thức sau đây :

$$C = E_{KB}(P) = E_B(P) = P^{KB} \pmod N . \quad (1)$$

Công việc giải mã là sự biến đổi ngược lại bản mã C thành bản rõ P dựa trên cặp khoá bí mật k_B, modulo N theo công thức sau :

$$P = D_{kB}(C) = D_B(C) = C^{kB} \pmod N . \quad (2)$$

Dễ thấy rằng, bản rõ ban đầu cần được biến đổi một cách thích hợp thành bản mã, sau đó để có thể tái tạo lại bản rõ ban đầu từ chính bản mã đó :

$$P = D_B(E_B(P)) \qquad (3)$$

Thay thế (1) vào (2) ta có :

$$(P^{KB})^{kB} = P \pmod N \qquad (4)$$

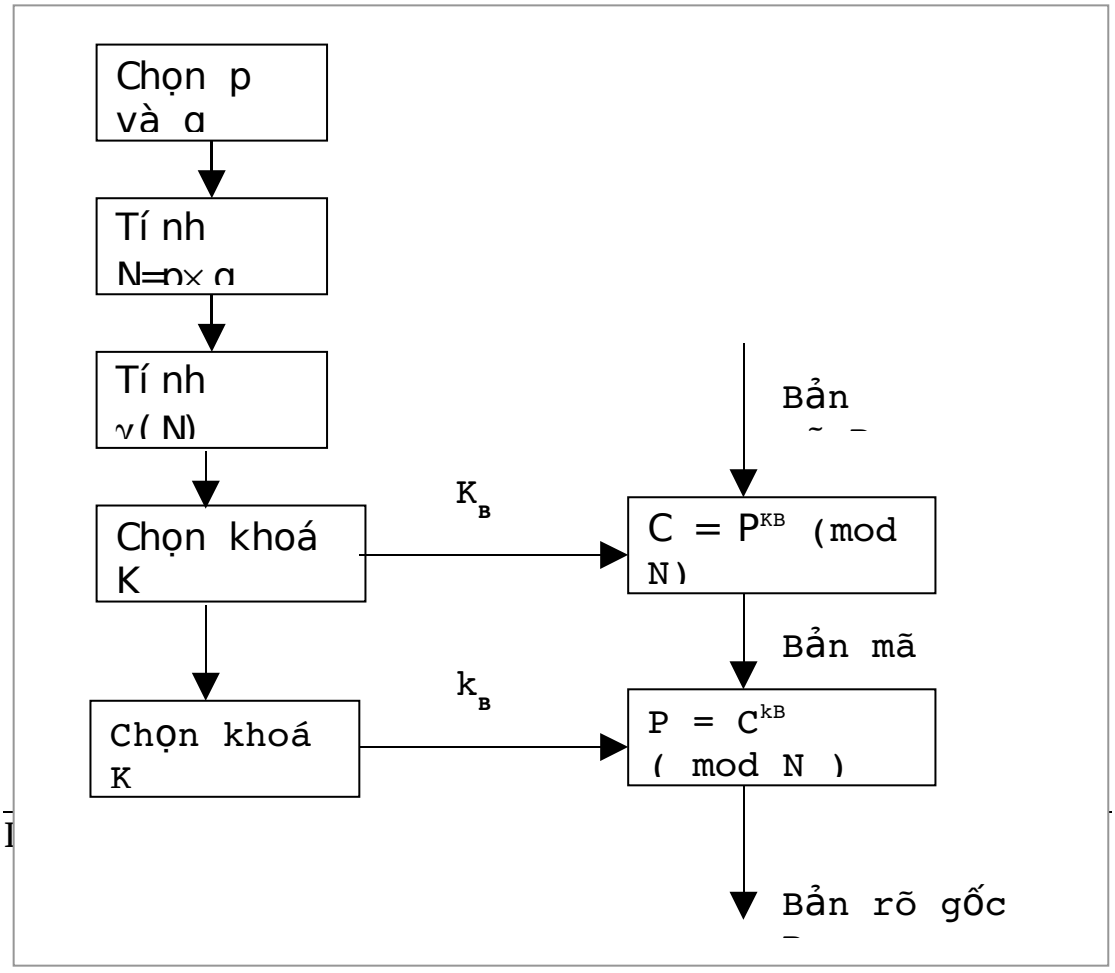
Trong toán học đã chứng minh được rằng, nếu N là số nguyên tố thì công thức (4) sẽ có lời giải khi và chỉ khi $K_B.k_B = 1 \pmod{N-1}$, áp dụng thuật toán ta thấy $N=p\times q$ với p, q là số nguyên tố, do vậy (4) sẽ có lời giải khi và chỉ khi :

$$K_B.k_B \equiv 1 \pmod{\gamma(N)} \qquad (5)$$

trong đó $\gamma(N) = LCM(p-1,q-1)$.

LCM (Lest Common Multiple) là bội số chung nhỏ nhất.

Nói một cách khác, đầu tiên người nhận B lựa chọn một khoá công khai K_B một cách ngẫu nhiên. Khi đó khoá bí mật k_B được tính ra bằng công thức (5). Điều này hoàn toàn tính được vì khi B biết được cặp số nguyên tố (p,q) thì sẽ tính được $\gamma(N)$.



Sơ đồ các bước thực hiện mã hoá theo thuật toán RSA

4.2.2. ĐỘ AN TOÀN CỦA HỆ RSA

Một nhận định chung là tất cả các cuộc tấn công giải mã đều mang mục đích không tốt. Trong phần độ an toàn của hệ mã hoá RSA sẽ đề cập đến một vài phương thức tấn công điển hình của kẻ địch nhằm giải mã trong thuật toán này.

Chúng ta xét đến trường hợp khi kẻ địch nào đó biết được modulo N, khoá công khai K_B và bản tin mã hoá C, khi đó kẻ địch sẽ tìm ra bản tin gốc (Plaintext) như thế nào. Để làm được điều đó kẻ địch thường tấn vào hệ thóng mật mã bằng hai phương thức sau đây:

- Phương thức thứ nhất :

Trước tiên dựa vào phân tích thừa số modulo N. Tiếp theo sau chúng sẽ tìm cách tính toán ra hai số nguyên tố p và q, và có khả năng thành công khi đó sẽ tính được $\lambda(N)$ và khoá bí mật k_B. Ta thấy N cần phải là tích của hai số nguyên tố, vì nếu N là tích của hai số nguyên tố thì thuật toán phân tích thừa số đơn giản cần tối đa $\sqrt{N}$ bước, bởi vì có một số nguyên tố nhỏ hơn $\sqrt{N}$. Mặt khác, nếu N là tích của n số nguyên tố, thì thuật toán phân tích thừa số đơn giản cần tối đa $N^{1/n}$ bước.

Một thuật toán phân tích thừa số có thể thành phức tạp hơn, cho phép phân tích một số N ra thành thừa số trong $O(\sqrt{P})$ bước, trong đó p là số chia nhỏ nhất của N, việc chọn hai số nguyên tố là cho thuật toán tăng hiệu quả.

- Phương thức thứ hai :

Phương thức tấn công thứ hai vào hệ mã hoá RSA là có thể khởi đầu bằng cách giải quyết trường hợp thích hợp của bài toán logarit rời rạc. Trường hợp này kẻ địch đã có trong tay bản mã C và khoá công khai K_B tức là có cặp (K_B, C)

Cả hai phương thức tấn công đều cần một sô bước cơ bản, đó là :

$O(\exp \sqrt{\ln N \ln(\ln N)})$, trong đó N là số modulo.

4.2.3. MỘT SỐ TÍNH CHẤT CỦA HỆ RSA

- *Trong các hệ mật mã RSA, một bản tin có thể được mã hoá trong thời gian tuyến tính.*

Đối với các bản tin dài, độ dài của các số được dùng cho các khoá có thể được coi như là hằng. Tương tự như vậy, nâng một số lên luỹ thừa được thực hiện trong thời gian hằng, các số không được phép dài hơn một độ dài hằng. Thực ra tham số này che dấu nhiều chi tiết cài đặt có liên quan đến việc tính toán với các con số dài, chi phí của các phép toán thực sự là một yếu tố ngăn cản sự phổ biến ứng dụng của phương pháp này. Phần quan trọng nhất của việc tính toán có liên quan đến việc mã hoá bản tin. Nhưng chắc chắn là sẽ không có hệ mã hoá nào hết nếu không tính ra được các khoá của chúng là các số lớn.

- *Các khoá cho hệ mã hoá RSA có thể được tạo ra mà không phải tính toán quá nhiều.*

Một lần nữa, ta lại nói đến các phương pháp kiểm tra số nguyên tố. Mỗi số nguyên tố lớn có thể được phát sinh bằng cách đầu tiên tạo ra một số ngẫu nhiên lớn, sau đó kiểm tra các số kế tiếp cho tới khi tìm được một số nguyên tố. Một phương pháp đơn giản thực hiện một phép tính trên một con số ngẫu nhiên, với xác suất 1/2 sẽ chứng minh rằng số được kiểm tra không phải nguyên tố. Bước cuối cùng là tính p dựa vào thuật toán Euclid.

Như phần trên đã trình bày trong hệ mã hoá công khai thì khoá giải mã (private key) k_B và các thừa số p,q là được giữ bí mật và sự thành công của phương pháp là tuỳ thuộc vào kẻ địch có khả năng tìm ra được giá trị của k_B hay không nếu cho trước N và K_B. Rất khó có thể tìm ra được k_B từ K_B cần biết về p và q, như vậy cần phân tích N ra thành thừa số để tính p và q. Nhưng việc phân tích ra thừa số là một việc làm tốn rất nhiều thời gian, với kỹ thuật hiện đại ngày nay thì cần tới hàng triệu năm để phân tích một số có 200 chữ số ra thừa số.

Độ an toàn của thuật toán RSA dựa trên cơ sở những khó khăn của việc xác định các thừa số nguyên tố của một số lớn. Bảng dưới đây cho biết các thời gian dự đoán, giả sử rằng mỗi phép toán thực hiện trong một micro giây.

Số các chữ số trong số được phân tích	Thời gian phân tích

50	4	giờ
75	104	giờ
100	74	năm
200	4.000.000	năm
300	5×10^{15}	năm
500	4×10^{25}	năm

4.3. THUẬT TOÁN MÃ HOÁ BLOWFISH

Blowfish là hệ mật mã khối 64-bit, khoá có độ dài có thể thay đổi được. Thuật toán bao gồm hai phần: phần phát triển khoá và phần mã hoá dữ liệu. Phát triển khoá chuyển một khoá có độ dài lớn nhất 448 bit thành một số mảng khoá con tổng cộng 4168 byte.

Mã hoá dữ liệu thực hiện thông qua mạng Feistel 16 vòng. Mỗi vòng bao gồm một hoán vị dựa vào khoá, thay thế dựa vào khoá và dựa vào dữ liệu. Tất cả các phép toán được dùng là phép XOR và phép cộng trên các từ 32-bit. Các thao tác thêm vào duy nhất là 4 mảng chỉ số để chỉ đến dữ liệu mỗi vòng.

4.3.1. KHOÁ PHỤ

Blowfish sử dụng một số lượng lớn các khoá phụ. Các khoá phụ này phải được tính toán trước khi mã hay giải mã dữ liệu.

- Mảng P bao gồm 18 khoá phụ 32-bit:

  P1,P2,P3,...,P18.

- Có 4 hộp S 32-bit với 256 đầu vào mỗi hộp:

  S1,0, S1,1, ..., S1,255;

  S2,0, S2,1, ..., S2,255;

  S3,0, S3,1, ..., S3,255;

  S4,0, S4,1, ..., S4,255.

Phương pháp chính xác được sử dụng để tính các khoá phụ này sẽ được mô tả ở phần sau.

4.3.2. MÃ HOÁ DỮ LIỆU

Blowfish là một mạng Feistel bao gồm 16 vòng. Đầu vào là x, một phần tử dữ liệu 64-bit.

- Chia x thành 2 phần 32-bit: xL, xR.
- For i=1 to 16:

  xL=xL XOR Pi

  xR=F(xL) XOR xR

  Đổi chỗ xL và xR
- Đổi chỗ xL và xR (tức là không đổi chỗ vòng cuối)
- xR=xR XOR P17
- xL=xL XOR P18
- Kết hợp xL và xR lại

  Hàm F :

  Chia xL thành 4 phần 8-bit: a, b, c và d

  $F(xL) = ((S1,a + S2,b \bmod 2^{32}) \text{ XOR } S3,c) + S4,d \bmod 2^{32}$

  Giải mã hoàn toàn giống như mã hoá trừ việc P1,P2,...,P18 được sử dụng theo trật tự ngược lại.

4.3.3. TÍNH TOÁN CÁC KHOÁ PHỤ

Các khoá phụ được tính sử dụng thuật toán Blowfish. Phương pháp chính xác như sau:

- Khởi tạo mảng P đầu tiên và sau đó là 4 hộp S theo thứ tự với một chuỗi cố định. Chuỗi này bao gồm các số hexa(hệ 16) của pi. Ví dụ:

P1=0x243f6a88

P2=0x85a308d3

P3=0x13198a2e

P4=0x03707344

- XOR P1 với 32 bit đầu của khoá, XOR P2 với 32 bit thứ hai của khoá tiếp tục cho tất cả các bit của khoá (có thể lên tới P14). Lặp lại theo vòng các bit khoá cho đến khi toàn bộ mảng P được XOR với các bít khoá. (Đối với các khoá ngắn có ít nhất một khoá dài tương ứng; ví dụ : nếu A là một khoá 64-bit thì AA,AAA, v.v.., là các khoá tương ứng)
- Mã hoá một chuỗi toàn 0 bằng thuật toán Blowfish sử dụng các khoá phụ đã mô tả trong bước (1) và bước (2).
- Thay thế P1 và P2 bằng đầu ra của bước (3).

- Mã hoá đầu ra của bước (3) dùng thuật toán Blowfish với các khoá phụ đã thay đổi.
- Thay thế P3 và P4 bằng đầu ra của bước (5).
- Tiếp tục xử lý, thay thế tất cả đầu vào của mảng P, và sau đó là 4 hộp S theo thứ tự, với đầu ra thay đổi liên tiếp của thuật toán Blowfish.

Tổng cộng cần có 521 lần lặp để sinh ra tất cả các khoá phụ.