CHƯƠNG 2

Mật mã khối và mật mã khóa đối xứng

1. Các khái niệm và nguyên lý thiết kế cơ sở

Các hệ mật mã cổ điển được giới thiệu trong chương trước đều thuộc loại mật mã dòng (stream cipher), trong đó phép biến đổi mật mã thực hiện trên từng ký tự độc lập. Tuy nhiên ngày nay được ưa chuộng sử dụng hơn là một kiểu mật mã khác – mật mã khối (block cipher) -- trong đó từng khối nhiều ký tự được mã hóa cùng một lúc. Trong mật mã khối, các tham số quan trọng là kích thước (độ dài khối) và kích thước khóa. Các khái niệm này được minh họa qua ví dụ sau đây.

Ví dụ 2.1 Bảng sau đây biểu diễn một thuật toán mã hóa theo khối

key	000	001	010	011	100	101	110	111
0	001	111	110	000	100	010	101	011
1	001	110	111	100	011	010	000	101
2	001	000	100	101	110	111	010	011
3	100	101	110	111	000	001	010	011
4	101	110	100	010	011	001	011	111

Theo bảng này, dữ liệu plaintext 010100110111 sẽ được mã hóa thành:

010 100 110 111 ➔ 111 011 000 101 theo key=1

010 100 110 111 ➔ 100 011 011 111 theo key=4

Ở đây số lượng khóa là 5, do $2^2 < 5 < 2^3$ nên cần 3 bit để biểu diễn và lưu giữ khóa, tức là kich thước khóa là 3. Đồng thời kích thước khối cũng là 3.

Cũng qua ví dụ đơn giản này (chỉ có tính chất minh họa), ta thấy rằng nếu các tham số kích thước khối và khóa qua nhỏ thì mật mã rất dễ bị phá bằng các tấn công thông qua phân tích thống kê. Chẳng hạn trong ví dụ trên, nếu kẻ thù nhận được một khối mã ciphertext 001 thì nó có thể dễ dàng suy ra plaintext tương ứng chỉ có thể là 000 hoặc 101 (nhờ thống kê trên bảng biến đổi mã).

Vì vậy***, các điều kiện cần cho mật mã khối an toàn*** là:

- Kích thước khối phải đủ lớn để chống lại các loại tấn công phá hoại bằng phương pháp thống kê. Tuy nhiên cần lưu ý rằng kích thước khối lớn sẽ làm thời gian trễ lớn.
- Không gian khóa phải đủ lớn (tức là chiều dài khóa phải đủ lớn) để chống lại tìm kiếm vét cạn.Tuy nhiên mặt khác, khóa cần phải đủ ngắn để việc làm khóa, phân phối và lưu trữ được hiệu quả.

Về các nguyên lý thiết kế mật mã khối, người ta đã ghi nhận 2 nguyên tắc cơ sở sau để có bảo mật cao, đó là việc tạo ra confusion (tính hỗn loạn, rắc rối) và diffusion (tính khuếch tán).

Confusion. (Hỗn loạn, rắc rối) Sự phụ thuộc của bản mã đối với bản rõ phải thực phức tạp để gây rắc rối, cảm giác hỗn loạn đối với kẻ thù có ý định phân tích tìm qui luật để phá mã. Quan hệ hàm số của mã-tin là phi tuyến (non-linear).

Diffusion. (Khuếch tán) Làm khuếch tán những mẫu văn bản mang đặc tính thống kê (gây ra do dư thừa của ngôn ngữ) lẫn vào toàn bộ văn bản. Nhờ đó tạo ra khó khăn cho kẻ thù trong việc dò phá mã trên cơ sở thống kê các mẫu lặp lại cao. Sự thay đổi của một bit trong một khối bản rõ phải dẫn tới sự thay đổi hoàn toàn trong khối mã tạo ra.

Một cách đơn giản nhất, *confusion* có thể được thực hiện bằng phép thay thế (substitution) trong khi *diffusion* được tạo ra bằng các phép chuyển đổi chỗ (transposition/permutation) hay hoán vị. Toàn bộ sơ đồ biến đổi mật mã sẽ là một lưới các biến đổi thay thế-hoán vị (substitution-permutation network).

Ví du 2.2: Phép hoán vị cột: Để mã hóa "computer security", ta viết lại thành nhiều hàng 5 cột

c o m p u

t e r s e

c u r i t

y.

Mã tạo ra bằng cách viết lại theo cột: C T C Y O E U M R R P S I U E T

Bên cạnh các nguyên tắc tạo tính bảo mật nói trên, việc thiết kế mật mã khối cũng đề cao các nguyên tắc cài đặt hiệu quả.:

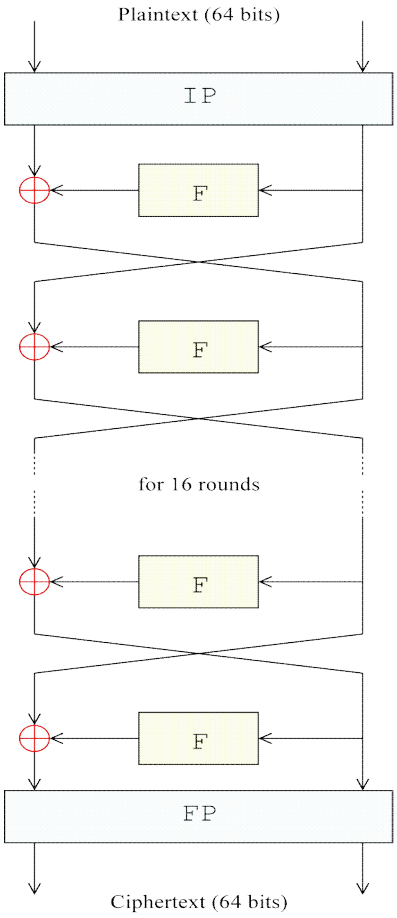
- Cài đặt cho phần mềm cần đảm bảo tính mềm dẻo và giá thành thấp.
- Cài đặt cho phần cứng cần đảm bảo tốc độ cao và tính kinh tế.

Để đáp ứng tốt các nguyên lý thiết kế đã nêu trên, các thuật toán mật mã khối thường được tổ chức như một cấu trúc nhiều vòng lặp.

Khái niệm vòng lặp

Một cách phổ biến, các hệ mã khối thường được thiết kế theo cấu trúc nhiều vòng lặp với mỗi vòng lặp lại gọi thực hiện một hàm *f* cơ sở (nhưng với các tham số khác nhau). Theo đó, đầu vào của một vòng lặp là đầu ra của vòng lặp trước và một khóa con phát sinh từ khóa đầy đủ dựa trên một thuật toán lập lịch khóa (key scheduler), hay cũng gọi là thuật toán sinh khóa con.

Giải mã sẽ là một quá trình ngược, trong đó các khóa con sử dụng tại mỗi vòng lặp sẽ được lập lịch để sử dụng theo thứ tự ngược.



Hình 2.1 *Sơ đồ minh họa một cấu trúc 16 vòng lặp, với đầu vào và ra đều có kích thước 64 bits (Nguồn: Wikipedia). Có hai khối hoán vị đầu và cuối (IP và FP). Hàm F cơ sở chỉ nhận đầu vào 32 bits, nhưng tác động của nó sẽ rộng khắp qua chỉ 2 vòng nhờ sự hoán vị 2 nửa trái và phải.*

Thông thường, hàm cơ sở vòng lặp *f* được thiết kế có một tính chất đặc biệt là tính đối hợp hàm (involution), tức là nó bằng hàm ngược của nó: $f = f^{-1}$ hay là $f(f(x)) = x$

Ví dụ 2.3 Ta xét phép biến đổi *f* với miền xác định: $x \in$ {tập các chuỗi nhị phân độ dài 3}

$f = \begin{bmatrix} 123 \\ 213 \end{bmatrix}$ (bit thứ nhất và thứ hai đổi chỗ cho nhau, bit thứ ba giữ nguyên).

Như thế ta có f là một hàm có tính đối hợp, chẳng hạn cụ thể là: $f(101) = 011$; từ đó $f(f(101)) = 101$

Chúng ta sẽ tìm hiểu chi tiết một hệ mã khối điển hình, đó là chuẩn mật mã DES (Data Encryption Standard); chuẩn này ra đời vào năm 1977 và đã thống trị ứng dụng mật mã suốt 2 thập kỷ sau đó. Tuy nhiên chuẩn mật mã này đã trở nên lạc hâu, kém an toàn và được thay thế bởi chuẩn mới AES (Advanced Encryption Standard).

2. Chuẩn mật mã DES

Lịch sử của DES

Vào những năm đầu thập kỷ 70, nhu cầu có một chuẩn chung về thuật toán mật mã đã trở nên rõ ràng. Các lý do chính là:

- Sự phát triển của công nghệ thông tin và của nhu cầu an toàn & bảo mật thông tin: sự ra đời của các mạng máy tính tiền thân của Internet đã cho phép khả năng hợp tác và liên lạc số hóa giữa nhiều công ty, tổ chức trong các dự án lớn của chính phủ Mỹ.
- Các thuật toán 'cây nhà lá vườn' (ad hoc) không thể đảm bảo được tính tin cậy đòi hỏi cao.
- Các thiết bị khác nhau đòi hỏi sự trao đổi thông tin mật mã thống nhất, chuẩn.

Một chuẩn chung cần thiết phải có với các thuộc tính như:

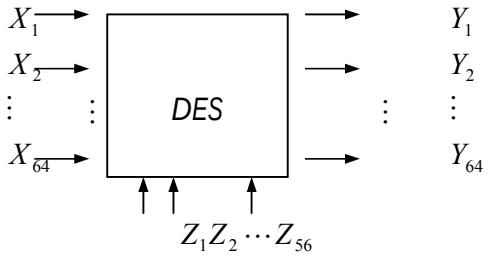
1. Bảo mật ở mức cao
2. Thuật toán được đặc tả và công khai hoàn toàn, tức là tính bảo mật không được phép dựa trên những phần che giấu đặc biệt của thuật toán.
3. Việc cài đặt phải dễ dàng để đem lại tính kinh tế
4. Phải mềm dẻo để áp dụng được cho muôn vàn nhu cầu ứng dụng

Năm 1973, Cục quản lý các chuẩn quốc gia của Mỹ đã có văn bản cổ động cho việc tạo lập các hệ mật mã chuẩn ở cơ quan đăng ký liên bang của Mỹ. Điều này đã dẫn đến sự công bố vào năm 1977 của cục An ninh Quốc gia Mỹ (NSA) về Data Encryption Standard, viết tắt là DES. Thực chất, DES được phát triển bởi IBM như là sự sửa đổi của một hệ mã trước kia được biết với cái tên Lucipher. Trong khoảng 2 thập kỷ tiếp theo, DES là hệ mã được dùng rộng rãi nhất và cũng là gây ra nhiều nghi ngờ, tranh cãi trong lĩnh vực này: xung quanh các nguyên tắc thiết kế đảm bảo tính mật, chiều dài khóa tương đối ngắn và khả năng NSA còn che giấu cửa sau

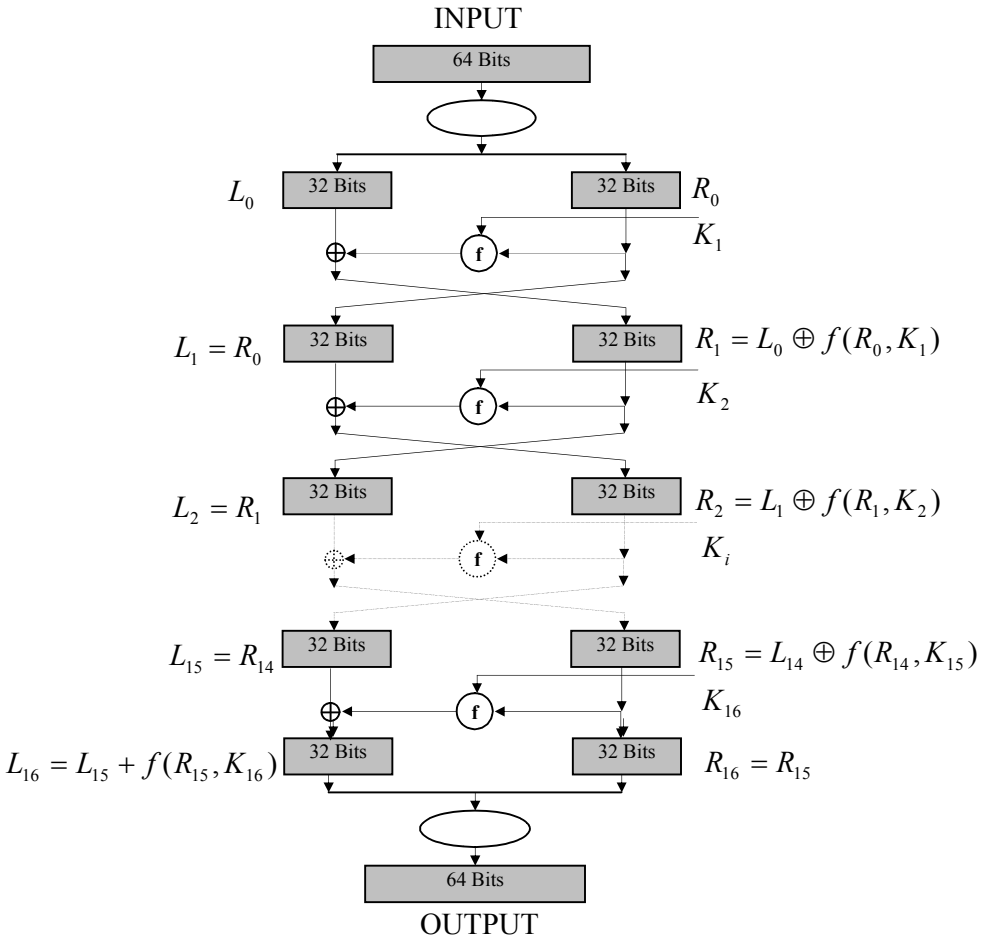
(backdoor) để có thể bẻ khóa, phá mã ít tốn kém hơn thông thường.

Thuật toán và lưu đồ hoạt động của DES

Các hình vẽ sau cung cấp sơ đồ khái quát và chi tiết của thuật toán sinh mã trong DES.



Hình 2.2 *Sơ đồ cơ bản của DES: đầu vào của DES là khối độ dài 64 bits, đầu ra 64 bits và khóa là 56 bits.*



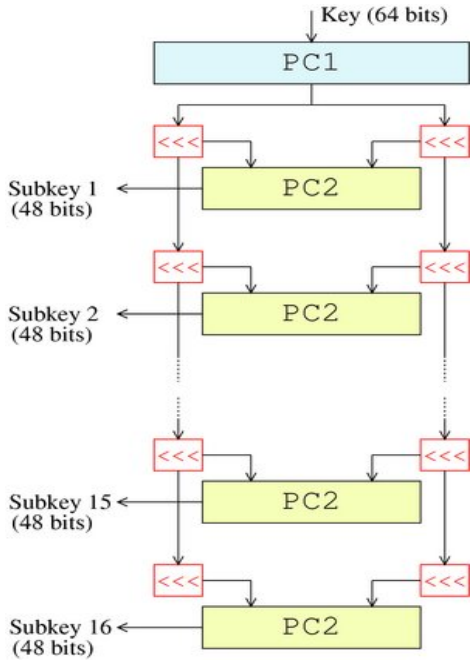
Hình 2.3 *Sơ đồ giải thuật sinh mã DES với cấu trúc 16 vòng lặp*

Sơ đồ hình vẽ 2.3 cho thấy DES được cấu tạo bởi 16 bước lặp với bước lặp cơ sở gọi hàm

chuyển đổi phi tuyến *f*; 16 bước lặp này được kẹp vào giữa hai tác tử giao hoán IP và IP^{-1}. Hai tác từ này không có ý nghĩa gì về mặt bảo mật mà hoàn toàn nhằm tạo điều kiện cho việc cài đặt phần cứng, 'chip hóa' thuật toán DES. Hàm cơ sở *f* là nguồn gốc của sức mạnh bảo mật trong thuật toán DES này. Sự lặp lại nhiều lần các bước lặp với tác dụng của *f* là nhằm tăng cường tính confusion và diffusion đã có trong *f*.

Thuật toán sinh khóa con

16 vòng lặp của DES cùng gọi thực hiện *f* nhưng với các tham số khóa khác nhau. Tất cả 16 khóa khác nhau này, được gọi là khóa con, cùng sinh ra từ khóa chính của DES bằng một thuật toán sinh khóa con. Trong thuật toán sinh khóa con này (lập lịch khóa), khóa chính K, 64 bit, đi qua 16 bước biến đổi, tại mỗi bước này một khóa con được sinh ra với độ dài 48 bit.



Hình 2.4 *Sơ đồ thuật toán sinh khóa con (Key Scheduler) – Nguồn: Wikipedia*

Qua sơ đồ thuật toán sinh khóa con có thể thấy rằng thực sự chỉ có 56 bit của khóa chính được sử dụng, 8 bit còn lại là mã kiểm tra chẵn lẻ (parity bits) và bị lọc ra ở biến đổi PC1. Các bộ biến đổi PC1 và PC2 chỉ đơn giản là các bộ vừa chọn lọc vừa hoán vị (PC = permuted choice = lựa chọn có hoán vị). Các biến đổi R1 và R2 (left rotate 1 bit và 2 bit) tương ứng là các phép đẩy bit trái 1 và 2 vị trí.

Cấu trúc vòng lặp DES

Mỗi vòng lặp của DES thực hiện trên cơ sở công thức sau:

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

trong đó, (L_i,R_i) là 2 nửa trái và phải thu được từ biến đổi của vòng lặp thứ *i*.

Ta cũng có thể viết lại

$$(L_i,R_i) = T\circ F(R_{i-1},K_i))$$

Trong đó *F* là phép thay thế L_{i-1} bằng $L_{i-1} \oplus f(R_{i-1},K_i)$, còn T là phép đổi chỗ hai thành phần L và R. Tức là mỗi biến đổi vòng lặp của DES có thể coi là một tích hàm số của F và T (trừ vòng cuối cùng không có T).

Ta có thể viết lại toàn bộ **thuật toán sinh mã DES** dưới dạng công thức tích hàm số như sau:

$$DES = (IP)^{-1}\circ F_{16}\circ T\circ F_{15}\circ T\circ \ldots \circ F_2\circ T\circ F_1\circ (IP)$$

Thuật toán giải mã DES được xây dựng giống hệt như thuật toán sinh mã nhưng có các khóa con được sử dụng theo thứ tự ngược lại, tức là dùng khóa K16 cho vòng lặp 1, khóa K15 cho vòng lặp 2 ... Vì vậy, thuật toán giải mã có thể được viết lại dưới dạng công thức sau:

$$DES^{-1} = (IP)^{-1}\circ F_1\circ T\circ F_2\circ T\circ \ldots \circ F_{15}\circ T\circ F_{16}\circ (IP)$$

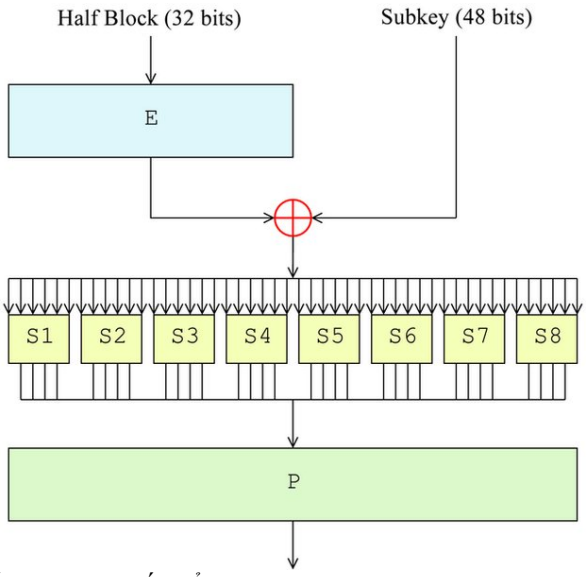
Bây giờ chú ý rằng mỗi hàm T hoặc F đều là các hàm có tính chất đối hợp ($f=f^{-1}$, hay f(f(x) =x).

Do đó nếu ta thực hiện phép tích hàm $DES^{-1}\circ DES$ hay $DES \circ DES^{-1}$ thì sẽ thu được phép đồng nhất. Điều đó giải thích tại sao thuật toán giải mã lại giống hệt như sinh mã chỉ có khác về thứ từ trong chuỗi khóa con.

Bài tập. Bạn đọc hãy tự chứng minh tính đối hợp của T và F đồng thời chỉ rõ tại sao x= DES (DES-1 (x) với mọi x là chuỗi nhị phân 64 bit.

Cấu trúc cụ thể hàm f

Sơ đồ biến đổi cụ thể của hàm *f* được minh họa trong hình 2.5. Trước hết, 32 bit của thành phần R_{i-1} được mở rộng thành 48 bit thông qua biến đổi E (expansion: mở rộng với sự lặp lại một số bit) rồi đem XOR với 48 bit của khóa K_i. Tiếp theo, 48 bit kết quả sẽ được phân thành 8 nhóm 6 bit. Mỗi nhóm này sẽ đi vào một biến đổi đặc biệt gọi là biến đổi S-box (có 8 S-box khác nhau ứng với mỗi nhóm 6 bit) và cho ra kết quả là 8 nhóm 4 bit. Từ đó, 32 bit hợp thành (sau khi qua 8 S-box khác nhau) sẽ được hoán vị lại theo hàm hoán vị P để đưa ra kết quả cuối cùng của hàm *f* (tức nhân của F_i).



Hình 2.5 *Cấu trúc của biến đổi hàm f, bước lặp cơ sở của DES. Nguồn: Wikipedia*

Cấu trúc của các S-Box

Như ta biết mỗi một trong 8 nhóm 6 bit sẽ đi vào mỗi trong 8 bộ biến đổi S_1,S_2 ... S_8.

Mỗi S-box bao gồm 4 bảng biến đổi dòng, thực chất là một biến đổi hoán vị cho 16 tổ hợp của 4 bits. Trong 6 bits đầu vào thì hai bit ngoài cùng (bit 1 và 6) được dùng để chỉ định 1 trong 4 bảng biến đổi dòng này; vì thế chúng được gọi là các bit điều khiển trái và phải (CL và CR). Còn lại 4 bit chính (các bit 2-5) của nhóm 6 bit đầu vào sẽ là tổ hợp 4 bits bị biến đổi.

S_5		**Middle 4 bits of input**															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Hình 2.6 *Bảng biến đổi S5: đầu vào 6 bits 011011 sẽ được biến đổi thành 1001 (ô vàng)*

Các thuộc tính của S-Box

Các nguyên tắc thiết kế của 8 S-box được đưa vào lớp thông tin mật 'Classified information' ở Mỹ. Mặc dù vậy, NSA đã tiết lộ 3 thuộc tính của S-boxes, những thuộc tính này bảo đảm tính confusion & diffusion của thuật toán.

1. Các bít vào (output bit) luôn phụ thuộc không tuyến tính vào các bít ra (input bit).
2. Sửa đổi ở một bit vào làm thay đổi ít nhất là hai bit ra.
3. Khi một bit vào được giữ cố định và 5 bit con lại cho thay đổi thì S-boxes thể hiện một tính

chất được gọi là 'phân bố đồng nhất ' (uniform distribution): so sánh số lượng bit số 0 và 1 ở các đầu ra luôn ở mức cân bằng. Tính chất này khiến cho việc áp dụng phân tích theo lý thuyết thông kê để tìm cách phá S-boxes là vô ích.

Rõ ràng, 3 tính chất này đảm bảo tốt confusion & diffusion. Thực tế, sau 8 vòng lặp tất cả các bit ra của DES sẽ chịu ảnh hưởng của tất cả các bit vào và tất cả các bit của khóa. Hơn nữa sự phụ thuộc này là rất phức tạp. Tuy nhiên sau này một số tấn công mới đã được đề xuất và cho thấy 8 vòng lặp này là chưa đủ để bảo mật (điều này cho thấy NSA đã biết trước các dạng tấn công này nên mới qui định số vòng lặp là 16 ngay từ đầu).

Chính cấu tạo của S-box đã gây tranh luận mạnh mẽ trong các thập kỷ 70-90 về khả năng cơ quan NSA (National Security Agency), Mỹ, vẫn còn che dấu các một số đặc tính của S-box hay cài bên trong những cửa bẫy (trapdoor) mà qua đó họ có thể dễ dàng phá giải mã hơn người bình thường (biết các bí mật này có thể giản lược không gian khóa 2^{56} để tìm kiếm vét cạn nhanh hơn). Sự phát hiện sau đó của các tấn công mới, rất mạnh như tấn công vi phân, đã củng cố sự nghi ngờ của giới khoa học.

Các điểm yếu của DES

1.Tính bù.

Ký hiệu $\bar{u}$ là phần bù của u (ví dụ 0100101 và 1011010 là bù của nhau) thì DES có tính chất sau:

$$y = DES_z(x) \Rightarrow \bar{y} = DES_{\bar{z}}(\bar{x})$$

Cho nên nếu biết MÃ y được mã hóa từ TIN x với khóa z thì ta suy ra $\bar{y}$ được mã hóa từ TIN $\bar{x}$ với khóa $\bar{z}$. Tính chất này chính là một điểm yếu của DES bởi vì nhờ đó kẻ địch có thể loại trừ một nửa số khóa cần phải thử khi tiến hành phép thử-giải mã theo kiểu tìm kiếm vét cạn không gian khóa.

2. Khóa yếu

Các khóa yếu là các khóa mà theo thuật toán sinh khóa con thì tất cả 16 khóa con đều như nhau

$$Z_1 = Z_2 = Z_3 = ... = Z_{15} = Z_{16}$$

điều đó khiến cho phép sinh mã và giải mã đối với các khóa yếu này là giống hệt nhau

$$DES_z = DES^{-1}{}_z$$

Có tất cả 4 khóa yếu như sau:

1) [00000001 00000001 ... ... 00000001]

2) [11111110 11111110 ... ... 11111110]

3) [11100000 11100000 11100000 11100000
11110001 11110001 11110001 11110001]

4) [00011111 00011111 00011111 00011111
00001110 00001110 00001110 00001110]

Đồng thời có 10 khóa yếu với thuộc tính là tồn tại Z, Z' sao cho

$DES^{-1}_{z} = DES_{z'}$ hay là $DES^{-1}_{z'} = DES_{z}$

Tấn công bằng phương pháp vét cạn (hay là brute-force attack)

DES có $2^{56}=10^{17}$ khóa. Nếu như biết một cặp plaintext-ciphertext thì chúng ta có thể thử tất cả 10^{17} khả năng này để tìm ra khóa cho kết quả khớp. Giả sử như một phép thử mất quãng 10^{-6}s (trên một máy PC thông thường), thì chúng ta sẽ thử mất 10^{11}s tức là 7300 năm!

Nhưng nhớ rằng đấy mới chỉ là sử dụng các máy tính thông thường, còn có các máy tính được chế tạo theo nguyên lý xử lý song song. Chẳng hạn nếu như làm được một thiết bị với 10^7 con chip mật mã DES chạy song song thì bây giờ mỗi con chip chỉ phải chịu trách nhiệm tính toán với 10^{10} phép thử. Chip mã DES ngày nay có thể xử lý tới tốc độ là 4.5 x 10^7bits/s tức là có thể làm được hơn 10^5 phép mã DES trong một giây.

Diffie và Hellman (1977) đã ước lượng rằng có thể chế được một máy tính chuyên dụng để vét cạn không gian khóa DES trong1/2 ngày với cái giá cho chiếc máy này là 20 triệu đô la. Cái giá này được tính toán lại và giảm xuống $200,000 vào năm 1987. Vì vậy DES đã bị phê bình ngay từ khi ra đời vì có kích thước khóa quá ngắn!

Hiện nay đã có những thiết kế cụ thể cho loại máy tính chuyên dụng phá khóa này dựa trên kỹ thuật xử lý song song tiên tiến và cho biết một thiết bị kiểu này có giá khoảng $10,000 có thể cho kết quả trong 1 ngày.

Sau đây là một đoạn trích, tham khảo từ nguồn Wikipedia (theo từ khóa DES):

In academia, various proposals for a DES-cracking machine were advanced. In 1977, Diffie and Hellman proposed a machine costing an estimated US$20 million which could find a DES key in a single day. By 1993, Wiener had proposed a key-search machine costing US$1 million which would find a key within 7 hours. However, none of these early proposals were ever implemented—or, at least, no implementations were publicly acknowledged. The vulnerability of DES was practically demonstrated in the late 1990s. In 1997, RSA Security sponsored a series of contests, offering a $10,000 prize to the first team that broke a message encrypted with DES for the contest. That contest was won by the DESCHALL Project, led by Rocke Verser, Matt Curtin, and Justin Dolske, using idle cycles of thousands of computers across the Internet. The feasibility of cracking DES quickly was demonstrated in 1998 when a custom DES-cracker was built by theElectronic Frontier Foundation (EFF), a cyberspace civil rights group, at the cost of approximately US$250,000 (see EFF DES cracker). Their motivation was to show that DES was breakable in practice as well as in theory: "*There are many people who will not believe a truth until they can see it with their own eyes. Showing them a physical machine that can crack DES in a few days is the only way to convince some people that they really cannot trust their security to DES.*" The machine brute-forced a key in a little more than 2 days search.

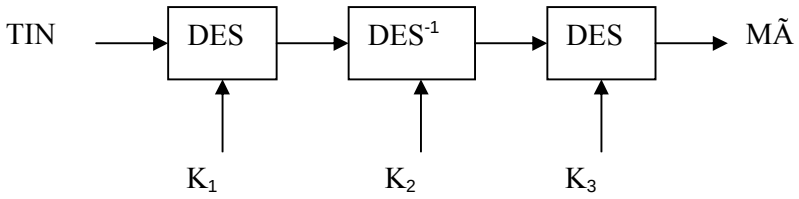
Tăng kích thước khóa của DES

Nếu như ta dùng nhiều khối DES nối tiếp thì có thể làm tăng kích thước của khóa. Tuy nhiên chú ý rằng nếu nối hai khối DES với hai khóa khác nhau thì không vì thế kích thước khóa của

cả hệ thống được tăng gấp đôi thành 56 *2 =112 bits mà chỉ là 57 bit.

Bài tập. Hãy giải thích tại sao.

Sơ đồ 3-DES dưới đây, trái lại, thực sự cung cấp một hệ mã với độ dài khóa là 112 bits



Hình 2.7 *Sơ đồ 3-DES (Triple-DES)*

Các dạng tấn công khác

Differential Cryptanalysis. Được công bố lần đầu bởi E. Biham và A. Shamir vào cuối những năm 80 (thế kỷ trước), tuy nhiên thực tế đã được biết đến từ lâu nhưng không công bố bởi IBM và NSA (Cục An ninh Quốc gia Mỹ). Để phá được DES với đầy đủ 16 vòng lặp, tấn công này cần tới 2^{49} bản rõ chọn trước (chosen plaintext). Để có được khối lượng bản rõ này là không thể xảy ra trên thực tế, điều đó cũng cho thấy là DES đã được thiết kế ban đầu để tránh được tấn công này.

Linear Cryptanalysis. Tấn công này được phát hiện bởi Matsui vào năm 1994, và cần 2^{43} bản rõ chọn trước.

3. Các hệ mật mã khối khác

Các mật mã khối khác (Cho đến năm 1999)

Qua thời gian, có nhiều thuật toán mật mã khối khác nhau được đề xuất bởi cộng đồng khoa học mật mã như FEAL (-4, -8, -N, -NX), NewDES, LOKI91, Blowfish, RC2, MMB, IDEA ... Tuy nhiên, khá nhiều trong số đó đã bị phá giải hoặc chỉ ra có những điểm yếu nhất định. Điều đó chứng tỏ đề xuất thuật toán mã khối tốt có thể thay thế được DES không phải là đơn giản.

Trong số nói trên IDEA (1990) có thể được xem là thuật toán có độ an toàn cao nhất, cho đến giờ vẫn chưa có một công bố nào nói lên một điểm yếu đáng kể nào của DES, mặc dù kể từ năm 1990 đã có nhiều loại tấn công rất mạnh được sử dụng để thử phá giải. IDEA chính là một trong các thuật toán được dùng trong PGP (Pretty Good Privacy) - một giải pháp bảo mật không thương mại gần như duy nhất cho phép các người dùng trên Internet sử dụng cho các nhu cầu thỏa mãn bí mật riêng như e-mail.

IDEA làm việc với dữ liệu khối 64 bit, nhưng với khóa128 bit nên việc thay thế sử dụng IDEA cho DES là một khó khăn lớn.

Mật mã AES

Vào năm 2000, cơ quan quản lý về chuẩn và công nghệ của Mỹ, NIST (National Institute of Standard and Technology), đã tổ chức một cuộc thi để chọn một hệ mật mã mới thay thế cho DES. Hệ mã Rijndael đã được chọn và được công bố (2002) như là chuẩn mật mã mới thay thế cho DES, với tên gọi là Advanced Encryption Standard (AES). Vào đến vòng trong còn có các ứng viên khác là RC6, Serpent, MARS và Twofish. Hệ mã này được phát triển bởi 2 nhà khoa học Bỉ, Joan Daemen và Vincent Rijnmen (vì vậy tên gọi Rijndael được tạo ra từ việc ghép tiền tố tên họ 2 ông này)

AES được xây dựng trên nguyên lý thiết kế *lưới giao hoán – thay thế* (substitution-permutation network). Đây là một hệ mã có tốc độ tốt trong cả cài đặt phần mềm cũng như phần cứng. Khác với DES, AES không theo mẫu thiết kế mạng Feistel. Thay vào đó các thao tác cơ bản được thực hiện trên các khối ma trận dữ liệu 4*4 (bytes), được gọi là các trạng thái (state). Số vòng lặp của AES là một tham số xác định trên cơ sở kích thước khóa: 10 vòng lặp cho khóa 128bit, 12 cho 192 bit, 14 cho 256bit.

Giáo trình này sẽ không đi sâu tìm hiểu về AES. Sinh viên được khuyến khích tìm đọc thêm từ các tài liệu tham khảo về AES.

4. Các chế độ sử dụng Mã khối

Thuật toán mã khối có đầu vào và đầu ra là các khối có độ dài xác định (như ở DES là 64bit). Để mã hóa một dữ liệu có độ dài tùy ý thì ta phải cắt dữ liệu thành nhiều khối đơn vị và áp dụng thuật toán mã nhiều lần, rồi sau sẽ kết hợp các khối dữ liệu thu được theo một sơ đồ nào đó. Có nhiều loại sơ đồ, hay còn gọi là chế độ mật mã khác nhau, với ưu nhược điểm khác nhau và được áp dụng cho các nhu cầu khác nhau. Sau đây là một số chế độ hay dùng.

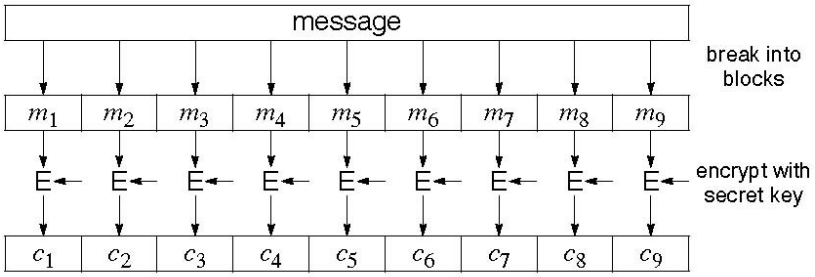
Chế độ bảng tra mã điện tử (Electronic code book - ECB)

Trong chế độ này, các khối được tạo mật mã riêng biệt, độc lập. Do đó, những khối tin giống nhau sẽ được mã hóa thành những khối mã giống nhau.

Điều này trở nên nguy hiểm, tạo miếng đất màu mỡ cho kẻ địch vận dụng tấn công replay cũng như thao tác biên tập theo khối. Kẻ thù có thể nghe trộm và tìm cách thu thập các mẫu tin-mã phổ biến, sau đó cắt ghép và trộn lẫn để tạo ra các bản mã giả mã bên nhận không phát hiện được. Ví dụ: Nếu ECB được sử dụng trong truyền tin mật trong giao dịch ngân hàng, kẻ địch có thể tấn công làm giả thông báo, lệnh chuyển tài khoản.

Nhược điểm nói trên khiến cho việc truyền tin mật theo chế độ mã này là không có lợi, tuy nhiên chế độ này thường được dùng trong mã hóa thông tin lưu trữ, ví dụ như các cơ sở dữ liệu vì nó cho phép từng đơn vị dữ liệu được mã hóa độc lập và do đó có thể cập nhật thay đổi dễ

dàng từng phần mà không động chạm đến các phần khác của cơ sở dữ liệu.



Hình 2.8 *Sơ đồ chế độ mật mã ECB*

Chế độ mã móc xích (Cipher Block Chaining - CBC)

Trong chế độ này, mỗi khối tin trước khi được mã hóa thì được XOR với khối mã sinh ra từ bước trước đó.

$$X_1 = X_1' \oplus IV$$

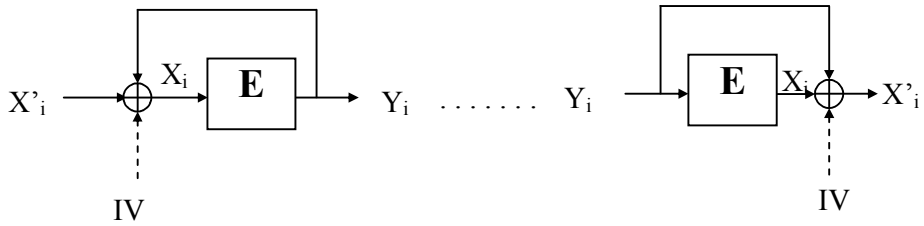
$$X_2 = X_2' \oplus Y_1$$

...

$$X_i = X_i' \oplus Y_{i-1}$$

Như vậy các khối mã đều phụ thuộc rất chặt vào nhau theo kiểu "*móc xích*". Cũng qua đó có thể thấy rằng CBC sẽ tạo ra các khối bản mã khác nhau khi các khối tin đưa vào là giống nhau tức là che giấu được các mẫu tin-mã phổ biến khỏi sự theo dõi của kẻ thù, chặn đứng khả năng phá hoại bằng tấn công replay và biên tập nói trên.

Tại bước đầu tiên, khi chưa có khối mã sinh ra từ bước trước, khối tin đầu sẽ được XOR với một vecto khỏi đầu, chọn ngẫu nhiên, ký hiệu là IV (initial vector).



Hình 2.9 *Sơ đồ chế độ mật mã CBC*

Tính chất phụ thuộc lẫn nhau của các khối bản mã còn đem lại một ưu thế nữa là ngăn chặn kẻ thù sửa đổi cắt xén mã truyền tin, vì dù chỉ thay đổi 1 bit trên mã cùng làm ảnh hưởng đến toàn bộ thông tin mà được giải mã từ đó, đến mức người nhận có thể phát hiện được dễ dàng do đoạn thông tin giải mã sẽ bị hoàn toàn vô nghĩa.

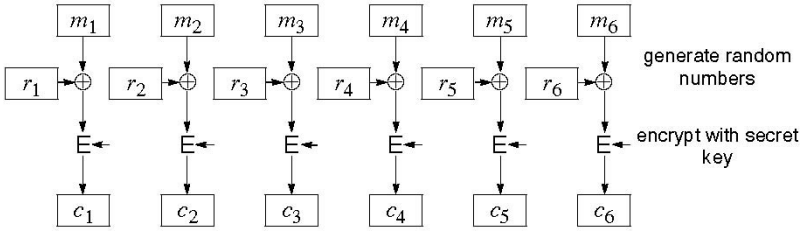
Tuy nhiên tính chất đó cũng đem lại một mối hại là nếu như mã truyền đi bị sai 1 ít do nhiễu thì giải mã sẽ bị ảnh hưởng lan truyền nhiều, dẫn đến phải phát lại. Ngoài ra chế độ CBC mặc định sự xử lý tuần tự, do đó không thể thực hiện tính toán song song, tức là không thể cải tiến được tốc độ cho hệ máy tính song song.

Liệu có tồn tại một cơ chế tấn công khác, thông minh hơn loại đã áp dụng cho ECB, để phá mã hoặc lợi dụng CBC? Lý luận về sự phụ thuộc móc xích mới chỉ cho ta một cảm giác an toàn chứ chưa phải là một chứng minh chặt chẽ. Tuy nhiên tính an toàn trong truyền tin mật của chế CBC đã được chứng minh chặt chẽ bằng phương pháp toán học

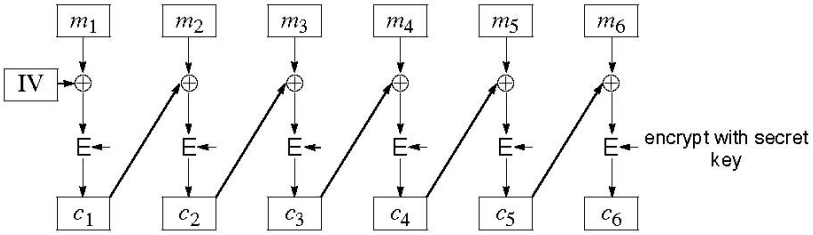
---

Bài tập. Hãy so sánh 2 dạng sơ đồ mật mã dưới đây từ đó liên hệ giữa CBC với mật mã one-time-pad

Sơ đồ A: Sử dụng một chuỗi ngẫu nhiên làm khóa chung



Sơ đồ B: biểu diễn lại CBC



Chế độ Mã phản hồi k-bit (k-bit Cipher Feedback Mode - CFB)

Với một số ứng dụng thời gian thực yêu cầu dòng dữ liệu truyền đến phải liên tục hơn là gián đoạn (như là chuỗi ký tự truyền giữa host và terminal phải tạo thành dòng ký tự liên tục). Do đó các chế độ mật mã khối xử lý và truyền theo từng khối một trở nên không thích hợp; các mã stream cipher với đơn vị xử lý là ký tự - khối 8 bit sẽ là thích hợp hơn với dạng ứng dụng này.

Chế độ CFB là một cải tiến cho phép tạo ra khả năng truyền khối nhỏ k-bit (với k tùy ý) trong khi vẫn dùng thuật toán mã khối.

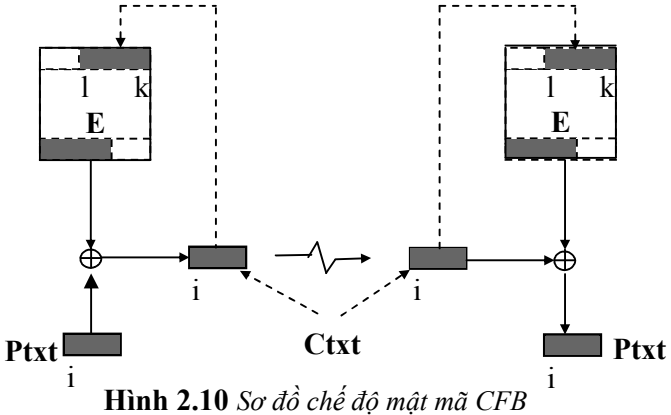
Dòng tin đi vào được 'múc' bằng từng 'gầu' với dung lượng *k* bit mà *k* là tham số thay đổi được. Thuật toán mật mã khối **E** chạy liên tục như một lò nấu: ở mỗi bước người ta lấy k bit (bên trái nhất) của vector đầu ra từ **E** để bỏ vào 'gầu' k bit tin, chúng được XOR với nhau. Kết

quả k bit vừa được đem truyền đi, vừa được bỏ lại vào đầu vào của thuật toán mã khối: vecto đầu vào được dịch trái k vị trí và k bit phải nhất sẽ được thay thế bởi k bit lấy từ gầu tin.

Như vậy có thể thấy rằng thuật toán mã khối được thực hiện như một hàm sinh các số giả ngẫu nhiên k-bit, các gía trị này lại được XOR với các phần tử k-bit tin lấy vào để tạo ra mã truyền đi.

Qua trình giải mã thì được tiến hành theo nguyên tắc đối xứng.

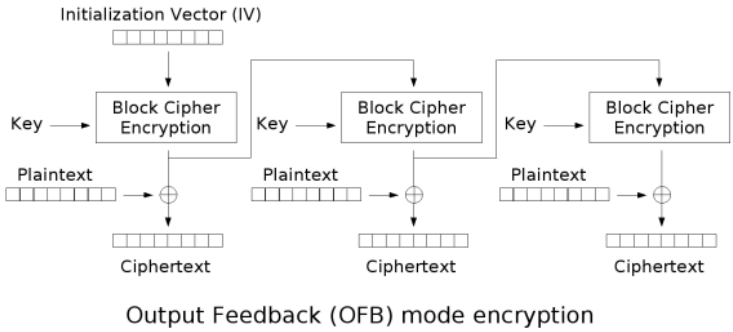
Rõ ràng chế độ này cũng cung cấp các khả năng như của chế độ CBC, thêm vào đó nó cho phép truyền tin với khối ngắn tùy ý, đảm bảo các ứng dụng về truyền-xử lý liên tục.



Hình 2.10 *Sơ đồ chế độ mật mã CFB*

Chế độ mật mã kết quả phản hồi (Output Feedback Mode – OFB)

Chế độ này cũng khá gần với hai chế độ trên đây, nhưng các phép XOR để tạo ra khối ciphertext là độc lập riêng rẽ, chứ không có sự phụ thuộc (móc xích) như trước. Các khối plaintext được XOR với các đầu ra – output – của các hàm sinh mã (thuật toán mật mã khối) mà riêng các phần tử output của hàm mã hóa này là vẫn phụ thuộc móc xích (nên được gọi là output feedback). Tuy nhiên chuỗi móc xích này có thể được thực hiện off-line thông qua tiền xử lý, trước khi thực sự có thông tin văn bản cần gửi đi. Chính vì vậy khả năng thời gian tính toán có thể được rút ngắn nhiều. Ngoài ra, chế độ này cũng cho phép mã khối nhỏ, như stream cipher, giống như với chế độ CFB vậy.



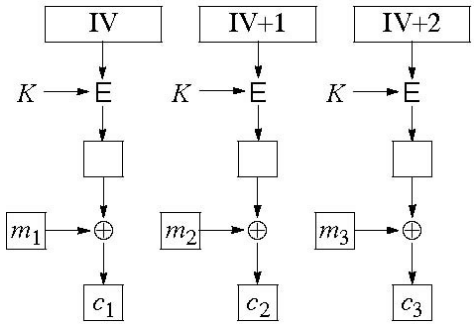
Hình 2.11 *Sơ đồ chế độ mật mã OFB*

Chế độ mật mã con đếm (Counter mode – CTR)

Đây là chế độ mật mã mới được phát minh không lâu lắm (2000) và được cho là ưu tú nhất. Sơ đồ của nó đơn giản một cách đáng ngạc nhiên! Sự móc xích (feedback) giữa các khối đã được loại trừ hoàn toàn, làm cho CTR có những hiệu năng tính toán cao đáng mong ước

- Có thể xử lý song song dễ dàng vì các khối tính toán hoàn toàn độc lập; ngoài ra cũng cho phép tiền xử lý để tính toán trước chuỗi phần tử output của hàm sinh mã (chẳng qua là chuỗi mã hóa của dãy số tự nhiên liên tiếp từ giá trị IV ban đầu).
- Không có sự phụ thuộc lẫn nhau nên có thể dùng vào mã hóa dữ liệu lưu trữ giống như với ECB: cho phép truy nhập ngẫu nhiên (random access) thay vì truy nhập tuần tự như với CBC chẳng hạn.

Mặc dù có sơn đồ tính toán rất đơn giản, tính an toàn của chế độ này đã được chứng minh đầy đủ bằng công cụ toán học hình thức, trên cơ sở thông qua so sánh với mật mã one-time-pad (đạt bí mật tuyệt đối.



Hình 2.12 *Sơ đồ chế độ mật mã CTR*