**3 MÃ BÍ MẬT**

**MẬT MÃ KHÓA BÍ MẬT**

**Nguyên lý thiết kế**

***Nguyên lý***

Như ta đã biết, hệ mã khóa bí mật (*secrete key cryptosystem*) hay còn gọi mã đối xứng (*symmetric cryptosystem*) là hệ mã mà các đối tác tham gia trao đổi thông tin mật sử dụng chung một khóa cho mã hóa lẫn giải mã (tính đối xứng – *symmetric*), vì thế, để thông tin không bị rò rỉ, tất cả đối tác phải giữ bí mật khóa (tính bí mật – *secrete*).

Tóm lại, an toàn của các hệ mã bí mật, trước hết, phụ thuộc vào việc bảo quản tính bí mật của khóa. Nghĩa là, không chỉ được giữ bí mật, khóa phải đảm bảo khó đoán.

Nhắc lại, hệ mã bí mật là một ánh xạ “khả nghịch”

và sao cho .

Để bảo mật, hệ mã phải kháng được các tấn công phục hồi bản rõ m từ bản mã, và suy diễn khóa k. Đó là nguyên lý cơ bản để thiết kế một hệ mã bí mật.

. Các không gian bản rõ – M (plaintext/message space), không gian bản mã – C (ciphertext space), và không gian khóa – K (key space) phải lớn để không thể vét cạn.

. Các phép biến đổi trong hàm E cũng như hàm D phải đủ phức tạp để không thể bị các tấn công thám mã, bao gồm cả tấn công phân tích tần suất và thám mã đại số hay vi sai.

***Tấn công phân tích tần suất***

Tấn công phân tích tần suất (*frequency analysis attack*) là các tấn công dựa vào tần suất xuất hiện của các ký hiệu đặc biệt.

Khi một hệ mã có thể bị phá chỉ bằng cách nghiên cứu bản mã, gọi là tấn công chỉ cần bản mã – **OCA** (*Only Ciphertext Attack*).

. Mọi hệ mã bị phá bằng tấn công chỉ cần bản mã đều không an toàn.

. Vét cạn là kỹ thuật cơ bản nhất để cài đặt OCA bằng cách thử qua tất cả các khả năng có thể có của khóa.

Tấn công khi biết bản rõ là một dạng tấn công, thường để phục hồi khóa bí mật, khi phân tích cặp bản rõ-bản mã.

. Thiết lập hệ phương trình từ các cặp bản rõ-bản mã và giải nó là kỹ thuật chung được dùng để phân tích khóa.

**Phương pháp thiết kế**

Hệ mã bí mật, mã công khai cũng vậy, là biến đổi thông điệp (*message*) đầu vào thành bản mã (*ciphertext*) đầu ra dựa trên khóa (*key*) sao cho chỉ người có khóa tương ứng mới có thể biến đổi ngược lại để đọc được thông điệp đã mã hóa. Hai phương pháp biến đổi cơ bản được dùng phổ biến trong các hệ mã là phép thay thế (*replace*) và phép hoán vị (*permutation*).

***Phép thay thế***

Trong mật mã, phép thay thế (replace) là thay thế một ký hiệu này bằng ký hiệu khác trong không gian các ký hiệu được dùng để xây dựng các không gian bản mã, bản rõ. Tiến trình, hay quy tắc thay thế thường phụ thuộc vào khóa.

Ví dụ thay các ký tự trong thông điệp bằng ký tự đứng sau nó k vị trí trong bảng chữ cái (*alphabet*), nếu vị trí vượt ra khỏi bảng chữ cái thì thứ tự tiếp cho chữ cái đầu bảng. Chẳng hạn, trong bảng chữ cái tiếng Anh, với k bằng 13, chữ ‘a’ được thay bẳng ‘n’, và chữ ‘n’ thay bằng ‘b’.

Thay thế cũng có thể được thực hiện cho các khối ký hiệu (*block*) như các hệ mã khối (*block cipher*) cài đặt.

***Phép hoán vị***

Trong mật mã, phép hoán vị có thể là

. Thay một ký hiệu bằng một ký hiệu là kết quả của phép hoán vị được định nghĩa trên tập ký hiệu nền (alphabet hay 0-9, chẳng hạn); hoặc

. Thay đổi vị trí trong các khối đầu vào.

Ví dụ, là hàm hoán vị có thể được áp dụng trên các không gian xây dựng trên tập ký tự (*alphabet*) là các ký số thập phân sẽ mã thông điệp n ký số m = m1…mn thành c = c1…cn với ci = σ(mi), i = 1, 2, …, n. Hay có thể áp dụng cho các khối có 9 ký hiệu trong bảng ký tự bất kỳ. Trong trường hợp này, mã hóa có thể xem là chuyển vị, thay đổi vị trí các ký hiệu. Chẳng hạn m = m0m1…m9 đượcmã hóa thành c = mσ(0)mσ(1)…mσ(9).

Một hệ mã hiện đại là sự kết hợp của các phép biến đổi thay thế và hoán vị, và được thực hiện trong nhiều vòng (*round*).

**MỘT SỐ HỆ MÃ BÍ MẬT**

Chúng tôi giới thiệu sau một số hệ mã bí mật đã từng được sử dụng trong thực tế, nhằm minh họa các nguyên lý thiết kế hệ mã đồng thời qua đó cũng minh họa một số kỹ thuật thám mã cơ bản.

**Mã tuyến tính**

Mã tuyến tính (*Affine cipher*) là hệ mã mà bản mã (*ciphertext*) C là tổ hợp tuyến tính của bản rõ (*plaintext*) P với khóa (*key*) k, và được thực hiện trên tập có n ký tự (|Alphabet| = n).

***Mã hóa và giải mã***

**Mã hóa**:

trong đó k là số nguyên nguyên tố cùng nhau với n, gcd(k, n) = 1.

Do gcd(k, n) = 1, nên tồn tại k-1 (mod n).- Vậy, để giải mã, tính

**Giải mã**:

***Ví dụ***

Mã hóa thông điệp M = WAR = 22 00 17 với khóa (k = 7, m = 10):

C1 = 7\*22 + 10 ≡ 8 (mod 26).

C2 = 7\*0 + 10 ≡ 10 (mod 26).

C3 = 7\*17 + 10 ≡ 25 (mod 26).

Được C = 08 10 25 = IKZ

Để giải mã, trước hết, sử dụng thuật toán Euclide mở rộng (Bezout) để tính 7-1 (mod 26) = 15. Vậy,

P1 = 15\*(8 – 10) ≡ 22 (mod 26).

P2 = 15\*(10 – 10) ≡ 0 (mod 26).

P3 = 15\*(25 – 10) ≡ 17 (mod 26).

Ta được lại đúng M = 22 00 17 = WAR.

Lưu ý,

. b có thể là một thành phần của khóa bí mật.

. Hệ mã Caesar là hệ mã cổ điển sử dụng bộ chữ cái của ngôn ngữ tự nhiên.

***Phân tích***

Hệ mã tuyến tính có thể được phân tích bằng tấn công chỉ dùng bản mã. Có 2 kỹ thuật tấn công: phân tích tần số và vét cạn.

*Tấn công phân tích tần số* – **frequency analysis attack**

(1) Giả sử bảng chữ cái là tiếng Anh và thứ tự A = 00, B = 01,…, Z = 25.

(2) Trong tiếng Anh, người ta thống kê thấy chữ cái E (04) có tần suất xuất hiện nhiều nhất, kế đến là chữ T (19).

(3) Biết thứ tự của E và T, từ nhiều bản mã, có thể suy ra khóa (k và b).

Chẳng hạn, chữ cái xuất hiện nhiều nhất trong bản mã là V (21), tiếp đến là E (04) và T (19). Ta thiết lập hệ phương trình

21 ≡ 4\*k + b (mod 26)

4 ≡ 19\*k + b (mod 26)

Giải hệ trên được k = 11 và b = 3

Thử lại, nếu đọc được bản mã thì (11, 3) chính là khóa đã được dùng trong quá trình mã hóa. Nếu không thể đọc được, ta thử với các khả năng khác.

*Tấn công vét cạn* – **brute-force attack**

Tấn công vét cạn là thử qua mọi khả năng một khóa có thể nhận, khi biết thuật toán mã hóa/giải mã Ek. Quá trình tấn công đơn giản là lần lượt thử từng khóa k trong không gian khóa K để giải mã bản mã C: P = Dk(M). Nếu M có thể ‘hiểu’ thì k là khóa cần tìm. Qua trình này có thể kết hợp với các công cụ của Trí tuệ Nhân tạo – **AI** (*Artificial Intelligence*) như các công cụ của Xử lý Ngôn ngữ Tự nhiên – **NLP** (*Natural Language Processing*). Quá trình tự động hóa sẽ đơn giản hơn nếu biết một cặp (bản rõ, bản mã) = (P, C). Khi đó có thể thiết lập điều kiện tìm kiếm khóa k trong K:

for k in K:

if Dk(C) == P: return k

Tấn công vét cạn khóa sẽ thành công khi không gian khóa K không lớn. Trong thực tế, tấn công vét cạn thường được cài đặt với một số heuristic, chẳng hạn chỉ vét cạn trong tập các khóa yếu, là những khóa hay được đa số người dùng đã sử dụng; hoặc các khóa được đặt theo tên người, theo ngày sinh,…

**Mã ma-trận**

***Mã hóa và giải mã***

Mã ma-trận (*matrix cipher*) là một mở rộng của mã truyên tính để mỗi lần mã hóa một khối các ký hiệu trong bảng chữ cái, nên mã ma-trận còn được gọi là mã khối (*block cipher*).

*Mã hóa* – **encryption**

Mã ma-trận sử dụng phép biến đổi

**Mã hóa**: C = KP + B (mod n),

với K là ma-trận vuông cấp m, khả nghịch (mod n), P là khối thông điệp m ký hiệu được biểu diễn như một vec-tơ cột, và B là vec-tơ cột m thành phần.

*Giải mã* – **decryption**

Do K khả nghịch (mod n) nên K-1 (mod n) tồn tại. Quá trình giải mã được thực hiện theo

P = K-1(C – B) (mod n).

Thực vậy,

K-1(C – B) ≡ K-1(KP + B – B) ≡ K-1KP ≡ IP ≡ P (mod n).

***Phân tích***

Rõ ràng, sử dụng các phương pháp tấn công như đã dùng trong tấn công mã tuyến tính sẽ khó khả hơn khi tấn công mã ma-trận, do các mẫu đa dạng hơn trong các không gian lớn hơn. Tuy nhiên, khi tin tặc thu thập được m cặp (bản rõ, bản mã), thì có thể sử dụng phương pháp tấn công biết bản rõ. Thực vậy, không mật tính tổng quát,

Giả sử là k = m2 + m cặp (bản rõ, bản mã) thu thập được.

Đặt

,

Ta có

,

thì giải hệ phương trình tuyến tính (mod n) trên, có thể phục hồi được K và B.

***Ví dụ***

*Ví dụ* 1 *–* **encryption and decryption**

Với . Để mã hóa bản rõ P = “TH” , thì

= “LB”.

Để giải mã, trước hết, tính

,

và

= “TH”.

*Ví dụ* 2 – **cryptanalysis**

Giả sử biết “BTGTHM” là văn bản mã của “ATTACK”.

Đặt .

. Từ cặp (BT ↔ AT), ta có

0\*a + 19\*b + s ≡ 1 (mod 26)

0\*c + 19\*d + t ≡ 19 (mod 26)

. (TA ↔ GT),

19\*a + 0\*b + s ≡ 6 (mod 26)

19\*c + 0\*d + t ≡ 19 (mod 26)

. (CK ↔ HM)

2\*a + 10\*b + s ≡ 7 (mod 26)

2\*c + 10\*b + t ≡ 12 (mod 26)

Giải hệ 6 phương trình, 6 ẩn số, trên ta được,

.

**Mã hoán vị**

***Ma-trận hoán vị***

Mã khối hoán vị (*block* *permutation cipher*) đơn giản là hoán vị các ký hiệu trong một khối.

Mã hoán vị có thể được cài đặt bằng mã ma-trận với khóa là một ma-trận hoán vị, và B là vec-tơ cột O, vec-tơ có tất cả các phần tử bằng 0.

Một ma-trận vuông A là ma-trận hoán vị nếu A là mỗi cột của A chỉ có 1 số 1 và tất cả các cột của A khác nhau.

Ví dụ là một ma-trận hoán vị.

***Ví dụ***

Cho ,

và văn bản rõ là “THIS IS NIRVANA”.

Chia văn bản rõ thành các khối 5 ký tự, khối cuối thiếu 2 ký tự ta thêm XX:

THIS IS NIRVANA → THISI SNIRV ANAXX

Mã hóa bằng phương pháp mã ma-trận với K như trên và B = O, ta được

IISHT IVRNS AXXNA.

**MẬT MÃ HIỆN ĐẠI**

**Mã hoán vị – thay thế**

Như ta thấy, thay thế và hoán vị là 2 kỹ thuật căn bản để tạo các hệ mã. Như vậy, để thiết kế một hệ mã, người thiết kế chỉ cần xây dựng được các phép biến đổi để hoán vị và thay thế đủ tốt và kết hợp chúng theo cách riêng là có thể có được một hệ mã bí mật an toàn. Như ta cũng biết, độ bảo mật của các hệ mã bí mật phụ thuộc chính vào phương pháp bảo quản khóa bí mật.

***Mã dựa trên ma-trận***

Bằng cách kết hợp mã ma-trận, một dạng mã thay thế (*replace cipher*), với mã hoán vị (*permutation cipher*) ta được hệ mã an toàn hơn, nhất là khi chiều dài khối thay đổi. Chẳng hạn,

. sử dụng ma-trận khóa K để ánh xạ khối n ký hiệu của bản rõ P thành bản mã lần một C’

. sau đó ghép nhóm lại thành các nhóm m ký hiệu, và áp dụng phép hoán vị trên các nhóm này để được bản mã lần hai C’’

. cuối cùng, áp dụng khóa ma-trận lần nữa để được bản mã cuối cùng C.

Quá trình hoán vị-thay thế có thể được thực hiện nhiều vòng, mỗi vòng là một cặp hoán vị-thay thế (*permutation-replace*) như trên là cách làm của nhiều hệ mã hiện đại.

Ví dụ, với

và ,

và văn bản rõ “SCOOBY DOO WHERE ARE YOU”.

. Để thực hiện mã hóa, trước hiết, ta gom nhóm-2 bản gốc thành

SC OO BY DO OW HE RE AR EY OU ↔ 1802 1414 0124 0314 1422 0704 0017 0424 1220

. Dùng khóa K để mã hóa được

C’ = 2800 0108 0202 2416 0724 0412 0200 0823 1714 2520

. Ghép nhóm-5

2800010802 0224160724 0412020008 2317142520

. Dùng khóa hoán vị T,

C’’ = 0828020001 0702242416 0004081202 2523201714

. Gom nhóm-2 và dùng khóa K lần nữa,

C = 0219 1510 2507 0706 0700 2110 1506 2421 1804 1620

↔ CT PK ZH HG HA VK PG YV SE QU

***Mã dựa trên tác động nhóm***

Như ta biết, mọi phần tử trong nhóm (G, \*) đều khả nghịch, nghĩa là . Như vậy, có thể chọn khóa và mã hóa và giả mã theo cách:

.

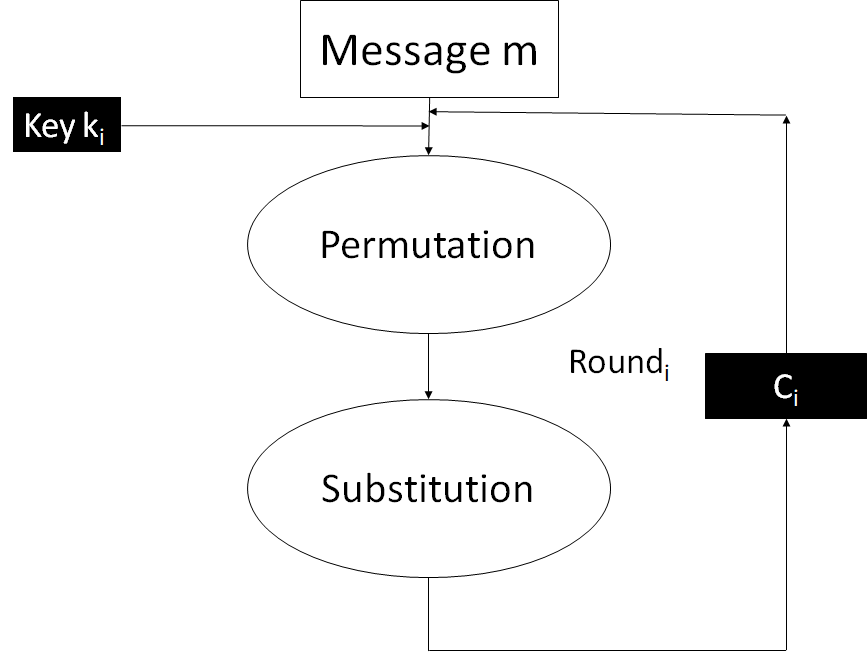
Tuy nhiên, trong thực tế, (G, \*) thường được chọn là một nhóm đơn sinh, G = < x >, và:

.

Nếu , với p nguyên tố, thì nếu là phần tử sinh của G thì rõ rang là một hoán vị và là thế *m* bằng .

**Mã hiện đại**

Mã hiện đại – **AES** (*Advanced Encryption System*) là các hệ mã xây dựng dửa trên khái niệm mạng hoán vị-thay thế của Feitel, gọi là Feitel-network, và có cấu trúc tổng quát như hình vẽ.



Và được mô tả hình thức là một hàm hợp gồm một phép biến đổi tuyến tính và một phép thay thế:

,

Trong đó A là một phép biến đổi tuyến tính và

.

***Hệ mã DES***

DES – Data Encryption Standard, là hệ mã bí mật được IBM đề xuất và được Viện Chuẩn và Công Nghệ Quốc Gia Hoa Kỳ (NIST – *National Institute of Standard and Technology*) phê chuẩn năm 1977.

DES sử dụng một phép hoán vị-thay thế và một số toán tử phi tuyến khác. Các toán tử phi tuyến này được áp dụng nhiều vòng (16 vòng) vào từng khối của thông điệp 32 bit như sau: Bản rõ, trước hết được chia thành các khối thông điệp 64 bit. Khóa sử dụng 56 bit nhận được từ khóa 64 bit nhập vào, có 8 bit kiểm tra chẵn lẻ. Thuật toán giải mã được thực hiện theo chiều ngược lại với cùng khóa bí mật được dùng khi mã hóa. Cụ thể

DES nhận vào thông điệp 64-bit M, khóa 56-bit K, và trả về bản mã 64-bit C:

(**Bước 1**), áp dụng phép hoán vị bit khởi tạo – **IP** (*Initiation Permutation*) vào M, tạ ra M’ = IP(M).

(**Bước 2**), chia M’ thành 2 nửa: nửa trái 32 bit L0 và nửa phải 32 bit R0.

(**Bước 3**), thi hành 16 vòng các phép toán sau:

Li = Ri – 1

Ri = Li-1 ⊕ f(Ri-1, Ki)

trong đó, *f* là hàm nhận nửa 32 bit và một khóa vòng 48 bit, sinh ra một kết xuất 32 bit. Mỗi khóa vòng Ki chứa một tập các khóa 56 bit.

(**Bước 4**), sau 16 vòng, ta được C’ = (R16, L16). C’ sau đó được hoán vị ngược lại để được bản mã cuối cùng C = IP-1(C’).

Giải mã được thực thi theo trình tự ngược lại: một phép hoán vị, 16 vòng XoR sử dụng khóa vòng theo thứ tự ngược lại, và phép hoán vị sau cùng để phục hồi bản rõ.

Hiện nay, DES 56-bit đã bị phá bằng thám mã vi sai và/hay thám mã đại số. Vì thê, saun cuộc thi, năm 1999, NIST công bố 5 chuẩn mã hóa hiện đại mới, gọi chung là AES gồm

. MARS, do IBM phát triển.

. RC6, do RSA Lab phát triển.

. Rijndael, Joan Daemen và Rijmen (Catholic De Nouvel) phát triển.

. Serpent, do Ross Anderson (Anh), Eli Bihan (Do Thái), và Lars Knudsen (Na Uy) phát triển.

. Twofish, do Bruce Scheineir, John Kelsey, Doug Whiting, David Wagner Chris Hall, và Niels Ferguson (Mỹ) phát triển.

Hiện, AES của ĐH Công Giáo Bỉ được dùng phổ biến nhất. Tuy vậy, có thể dùng Triple-DES (3DES) cũng cho kết quả bảo mật tương đương.

***Hệ mã 3DES***

Đặt E(k, M) và D(k, C) ký hiệu mã hóa và giải mã DES của M và C với khóa k. Mỗi phép mã hóa và giải mã của 3DES là hợp của 3 phép DES. Cụ thể

. 3DES mã hóa biến đổi khối 64 bit M thành khối 64 bit C:

C = E(K3, D(K2, E(K1, M))).

. 3DES giải mã biến đổi khối 64 bit C thành khối 64 bit M:

M = D(K1, E(K2, D(K3, C))).

Có 3 cách khác nhau để tổ hợp khóa cho 3DES: K1, K2, K3.

(**C1**) K1, K2, K3 là 3 khóa khác nhau.

(**C2**) K1, K2 khác nhau, K3 = K1.

(**C3**) K1 = K2 = K3.

Ví dụ, nếu chọn (C2), là cách phổ biến nhất, thì 3DES được gọi như sau:

C = E(K1, D(K2, E(K1, M))) ↔ M = D(K1, E(K2, D(K1, C))).

**PHƯƠNG THỨC HOẠT ĐỘNG CỦA MÃ BÍ MẬT**

Theo định nghĩa,

,

với M và C là các tập (dù rất lớn nhưng) hữu hạn. Cụ thể, mỗi phần tử trong M và C được biểu diễn bằng n bit cố định. Nhưng trong thực tế, văn bản thường lớn hơn n bit rất nhiều. Vì thế, văn bản được cắt thành các chuỗi bit phù hợp để có thể áp dụng hàm E. Cắt chia văn bản lớn như vậy dẫn đến 2 khái niệm: mã khối (*block cipher*) và mã dòng (*stream cipher*).

**Mã khối – block cipher**

Văn bản được cắt thành các khối có chiều dài n lớn hơn đơn vị (đơn vị có thể là từ (word), byte, hay chuẩn nhất là bit). Mã hóa văn bản, tài liệu,… thường sử dụng phương pháp này.

**Mã dòng – stream cipher**

Không phải chỉ áp dụng trên các ‘khối’ có độ dài đơn vị, mà bản chất nó nằm ở chữ ‘dòng – stream’, nghĩa là dữ liệu vào liên tục, ồ ạt như dòng chảy, và không biết điểm dừng. Chẳng hạn, mã hóa các cuộc điện thoại, hay truyền hình trả tiền,… là những ứng dụng sử dụng mã dòng.

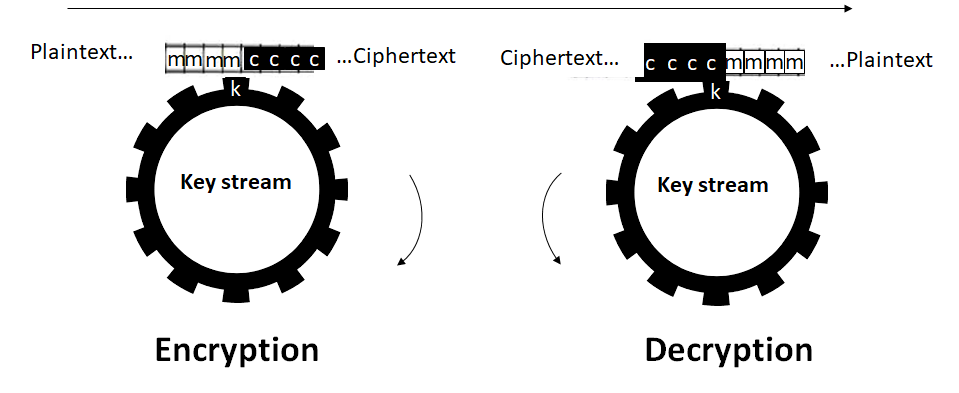
Nguyên tắc cơ bản của mã dòng là

. sinh ra một dòng khóa (key stream) thật dài và ngẫu nhiên.

. từng đơn vị dữ liệu sẽ được XoR với từng đơn vị dòng khóa.

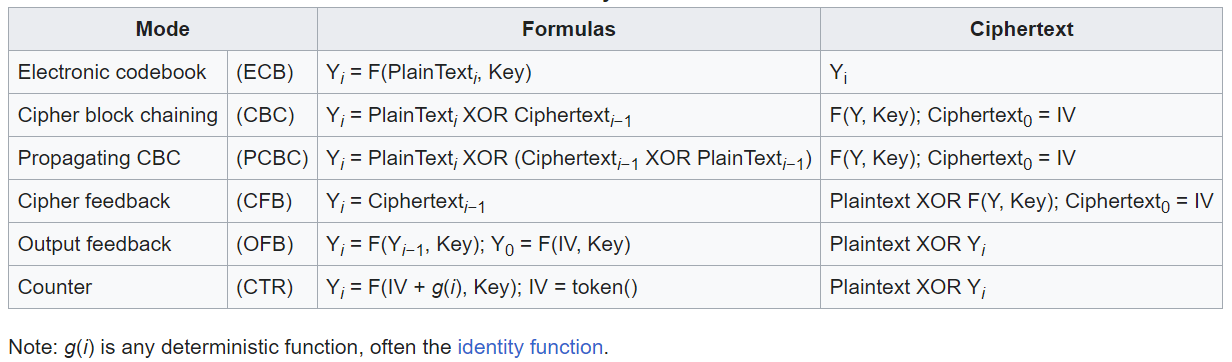
(Giải mã là quá trình tương tụ do ki ⊕ mj ⊕ ki = mj).

Hình sau minh họa hoạt động của hệ mã dòng.



**Các phương thức**

Bên cạnh cách chia cắt văn bản, việc kết hợp các khối trong tiến trình mã hóa toàn bộ văn bản cũng tạo nên các kết quả khác nhau. Bảng sau mô tả tóm tắt 6 phương thức thực hiện mã hóa trên văn bản lớn.



Giả sử văn bản rõ (*plaintext*) M được cắt thành n khối rời nhau cùng chiều dài bit:

M = m1m2…mi-1mi…mn,

và văn bản mã (*ciphertext*) được ghép từ n bản mã đều nhau:

C = c1c2…ci-1ci…cn.

Ký hiệu thông điệp rõ thứ i: mi = Plaintexti, i = 1, 2, …, n,

bản mã thứ j: cj = Yj = Ciphertextj, j = 2, 3, …, n, và

hàm mã/giải mã: F(X, k) mã hóa/giải mã X với khóa bí mật key.

Các thức hoạt động của hệ mã trên văn bản lớn như sau:

***Thức từ điển***

Trong thức từ điển – **ECB** (*Electronic Code Book mode*), các khối được mã hóa/giải mã độc lập nhau, lần lượt từ khối đầu tiên cho đến khối cuối cùng:

Yj = F(Plaintexti, key).

Gọi là **từ điển** vì mã hóa/giải mã có thể xem như tra trong từ điển có 2b hạng mục để biểu diễn số nguyên b-bit: ECB(i, F(I, k)), i = 0, 1, …, 2b - 1.

Sử dụng ECB chỉ có thể mã nội dung dữ liệu, hình thức được bảo toàn, vì 2 điểm dữ liệu có cùng giá trị sẽ cho cùng giá trị mã: Plaintexti = Plaintextj ↔ Ciphertexti = Ciphertextj.

**Thức chuỗi khối mã**

Trong thức chuỗi khối mã – **CBC** (*Cipher Block Chain mode*), bản mã hiện tại chính là mã của bản rõ hiện tại được trộn (XoR) với bản mã liền trước. Với bản mã đầu tiên là một giá trị khởi tạo ngẫu nhiên – **IV** (*Init Value*).

Ciphertext0 = IV #Initation Value

∀i = 1, 2, …, n:

Yi = Plaintexti ⊕ Ciphertexti-1

Ciphertexti = F(Yi, key)

Bằng cách liên kết thành chuỗi như vậy, thức CBC phá vỡ cấu trúc của văn bản dữ dữ liệu. Tuy nhiên, vì một lý do nào đó, nếu một khối mã bị ‘hư hỏng’, tiến trình giải mã sẽ gặp vấn đề.

***Thức lan truyền khối mã***

Thức lan truyền khối mã – **PCBC** (*Propagating CBC mode*) cũng tương tự thức CBC ngoại trừ bản mã được lan truyền cho khối sau: bản mã trước được trộn với bản rõ trước trước khi được trộn chung với bản rõ hiện hành.

Ciphertext0 = IV

∀i = 1, 2, …, n:

Yi = Plaintexti ⊕ (Ciphertexti-1 ⊕ Plaintexti-1)

Ciphertexti = F(Yi, key).

Thức PCBC này cũng phá vỡ được cấu trúc nhưng chưa giải quyết được vấn đề hư hỏng một vài khối mã.

***Thức hồi tác bản mã***

Thay vì liên kết bản rõ, thức hồi tác bản mã – **CFB** (*Ciphertext Feed Back mode*) liên kết các bản mã lại thành chuỗi, thay vì liên kết các bản rõ.

Ciphertext0 = IV

∀i = 1, 2, …, n:

Yi = Ciphertexti-1

Ciphertexti = Plaintexti ⊕ F(Yi, key)

Cũng như 2 thức chuỗi liên kết trước, thức CFB cũng chưa giải quyết được vấn đề bản mã hỏng.

***Thức hồi tác kết xuất***

Trong thức hồi tác kết xuất – **OFB** (*Output Feed Back mode*) này, các kết xuất được xâu kết thành chuỗi.

Y0 = F(IV, key)

∀i = 1, 2, …, n:

Yi = F(Yi-1, key)

Ciphertexti = Plaintexti ⊕ Yi

Thức OFB gợi ý cho cách giải quyết vấn đề hư hỏng của bản mã trong thức sau.

***Thức đếm***

Bằng cách hồi tác kết xuất của biến đếm, thức hồi tác đếm – **Counter** (*Counter mode*) khắc phục được vấn đề hư hỏng của các khối mã bằng cách ‘bỏ qua’ khối bị hư và đếm ‘nhảy cóc’ bằng hàm sinh số giả ngẫu nhiên cho khối i.

IV = token() #cần điểm xuất phát để có thể đếm khi có vấn đề hư hỏng bản mã xảy ra

∀i = 1, 2, …, n:

Yi = F(IV + g(i), key)

Ciphertexti = Plaintexti ⊕ Yi

**MỘT HỆ MÃ NHANH**

Phần cuối này giới thiệu một hệ mã nhanh dựa trên mã ma-trận (*matrix cipher*) và thực hiện trên .

Như ta đã thấy ở những phần trước, bằng cách thiết kế thích hợp, mã ma-trận có thể được thiết kế là một mã hiện đại gồm trong đó 2 phép biến đổi hoán vị và thay thế. Nhắc lại, mã hoán vị-thay thế dựa trên ma-trận, gồm các vòng:

. **Thay thế** (*substitution*). Sử dụng một ma-trận khóa K cấp n1 để mã hóa các phần tử trong khối n1 ký hiệu bằng cách thay thế nó với các ký hiệu trong tập chữ cái (*alphabet*).

. **Hoán vị** (*permutation*). Sử dụng ma-trận hoán vị cấp n2 (thường n2 ≠ n1) để đổi chỗ các ký hiệu nhận được sau bước mã thay thế.

Như ta cũng thấy, vấn đề chính của mã ma-trận là chọn ra được ma-trận khóa cấp n trên . Ta chọn p là số nguyên tố, thay vì số bất kỳ cho không gian khóa lớn nhất. Thực vậy, nhóm tuyến tính tổng quát (*General Linear group*), ký hiệu hay , là nhóm các ma-trận khả nghịch cấp n xác định trên . Số phần tử của nhóm GL(n, p) , chính là cấp của nó – O(GL(n, p)), được xác định theo công thức sau:

.

Chẳng hạn,

. *p* = 3, *n* = 2, số phần tử của GL(2, 3) bằng (32 – 30)(32 – 31) = 48.

. *p* = 2, số phân tử của GL(n, 2) bằng (2n – 20)(2n – 21) = (2n – 1)21(2n-1 – 1) = 2(2n – 1)(2n-1 – 1).

Nhận xét,

.

là một số rất lớn nếu n là số hàng vạn bit. Ta sẽ sử dụng kết quả này để tạo các ma-trận mã hóa nhị phân. Vấn đề của ta là sẽ sinh nhanh các ma-trận khả nghịch cấp n rất lớn.

**Tạo nhanh ma-trận khả nghịch**

Ta sẽ sử dụng một số kết quả của đại số tuyến tính để thiết kế thuật toán sinh nhanh các ma-trận khả nghịch cấp n.

**Định nghĩa** (ma-trận tam giác).

. Ma-trận tam giác dưới cấp *n*, ký hiệu L, là ma-trận có dạng:

.

. Ma-trận tam giác trên cấp *n*, ký hiệu U, là ma-trận có dạng:

.

Bây giờ, ta khảo sát tính khả nghịch của ma-trận.

**Mệnh đề** (về tính khả nghịch của ma-trận của ma-trận vuông cấp n).

. A khả nghịch nếu chỉ nếu định thức của A khác 0: A khả nghịch ⇔ det(A) ≠ 0.

Từ kết quả trên, ta có

.

Để tính định thức ma-trận, ta sử dụng kết quả sau:

**Mệnh đề** (về tính định thức của ma-trận vuông cấp n).

. Định thức của ma-trân tam giác bằng tích các phần tử trên đường chéo của nó:

,

.

. Định thức của ma-trận tích bằng tích các định thức ma-trận thừa số:

.

Từ kết quả trên, ta có

. .

. . (\*)

. .

. . (\*\*)

Và

. .

. . (\*\*\*)

Như vậy, để sinh nhanh một ma-trận nhị phân cấp n khả nghịch, ta sinh ngẫu nhiên các ma-trận tam giác L và U có các phần tử trên đường chéo đều bằng 1.

**Tìm ma-trận nghịch đảo**

***Trường hợp tổng quát***

Cho ma-trận khả nghịch A, theo định nghĩa, nếu A-1 là nghịch đảo của A thì .

Đặt , với , là vec-tơ đơn vị thứ i, dạng cột, nghĩa là chỉ có phần tử thứ i bằng 1, tất cả các phần tử khác bằng 0.

Giải n hệ phương trình tuyến tính

,

ta sẽ được ma-trận nghịch đảo A-1 của ma-trận khả nghịch A.

***Trường hợp tích các ma-trận tam giác***

Thuật toán phát sinh nhanh các ma-trận khả nghịch như trình bày trong phần trước luôn tra về ma-trận khóa K là tích 2 ma-trận tam giác K = LU hay K = UL. Trong trường hợp này, giả sử K = LU, ta có cách tính nhanh ma-trận nghịch đảo K-1.

Đặt , và giải n hệ phương trình tuyến tính tìm K-1.

Do nên ta có cách giải nhanh các hệ phương trình này như sau:

.

Như vậy, ta sẽ giải ‘cuốn chiếu’ 2 hệ phương trình tuyến tính đặc biệt và , như sau.

. Giải LX = B, với các phần tử của L, B, và X thuộc {0, 1}.

.

. Giải UX = B, với các phần tử của U, B, và X thuộc {0, 1}.

.