# Chương 1: CÁC KHÁI NIỆM CƠ SỞ & HỆ MÃ CỔ ĐIỂN

## 1.1 Mật mã là gì?

Mật mã là một lĩnh vực khoa học chuyên nghiên cứu về các phương pháp và kỹ thuật đảm bảo an toàn và bảo mật trong truyền tin liên lạc với giả thiết sự tồn tại của các thế lực thù địch, những kẻ muốn ăn cắp thông tin để lợi dụng và phá hoại.

Cụ thể hơn, các nhà nghiên cứu lĩnh vực này quan tâm xây dựng hoặc phân tích (để chỉ ra điểm yếu) các giao thức mật mã (cryptographic protocols), tức là các phương thức giao dịch có đảm bảo mục tiêu an toàn cho các bên tham gia (với giả thiết môi trường có kẻ đối địch, phá hoại).

Ngành Mật mã (cryptology) thường được quan niệm như sự kết hợp của 2 lĩnh vực con:

1. Sinh, chế mã mật (cryptography): nghiên cứu các kỹ thuật toán học nhằm cung cấp các công cụ hay dịch vụ đảm bảo an toàn thông tin
2. Phá giải mã (cryptanalysis): nghiên cứu các kỹ thuật toán học phục vụ phân tích phá mật mã và/hoặc tạo ra các đoạn mã giản nhằm đánh lừa bên nhận tin.

Hai lĩnh vực con này tồn tại như hai mặt đối lập, “đấu tranh để cùng phát triển” của một thể thống nhất là ngành khoa học mật mã (cryptology). Tuy nhiên, do lĩnh vực thứ hai (cryptanalysis) ít được phổ biến quảng đại nên dần dần, cách hiểu chung hiện nay là đánh đồng hai thuật ngữ cryptography và cryptology. Theo thói quen chung này, hai thuật ngữ này có thể dùng thay thế nhau. Thậm chí cryptography là thuật ngữ ưa dùng, phổ biến trong mọi sách vở phổ biến khoa học, còn cryptology thì xuất hiện trong một phạm vi hẹp của các nhà nghiên cứu học thuật thuần túy.

## 1.2 Các ứng dụng của mật mã là gì?

* Với các chính phủ: bảo vệ truyền tin mật trong quân sự và ngoại giao, bảo vệ thông tin các lĩnh vực tầm cỡ lợi ích quốc gia.
* Trong các hoạt động kinh tế: bảo vệ các thông tin nhạy cảm trong giao dịch như hồ sơ pháp lý hay y tế, các giao dịch tài chính hay các đánh giá tín dụng …
* Với các cá nhân: bảo vệ các thông tin nhạy cảm, riêng tư trong liên lạc với thế giới qua các giao dịch sử dụng máy tính và/hoặc kết nối mạng.

## 1.3 Những kỷ nguyên quan trọng trong ngành mật mã ?

*Thời kỳ tiền khoa học*: Tính từ thượng cổ cho đến 1949. Trong thời kỳ này, khoa mật mã học được coi là một ngành mang nhiều tính thủ công, nghệ thuật hơn là tính khoa học. Các hệ mật mã được phát minh và sử dụng trong thời kỳ này được gọi là các hệ mật mã cổ điển

*Kỷ nguyên mật mã được coi là ngành khoa học*: được đánh dấu bởi bài báo nổi tiếng của Claude Shannon “Commication theory of secretcy systems” , được công bố năm 1949. Công trình này dựa trên một bài báo trước đó của ông mà trong đó ông cũng đã khai sáng ra ngành khoa học quan trọng khác, *lý thuyết thông tin* (inforrmation theory).

## 1.4 Mô hình truyền tin mật cơ bản như thế nào?



Sender S



Receiv

er

R



Ene

my E



K

ey Z



Y=E

Z

X

(

)



Ke

y Z

‟



X=D

Z

)

(

Y

Người phát S (sender) muốn gửi một thông điệp (message) X tới người nhận R (receiver) qua một kênh truyền tin (communication channel). Kẻ thù E (enenmy) lấy/nghe trộm thông tin X. Thông tin X là ở dạng đọc được, còn gọi là bản rõ (plaintext). Để bảo mật, S sử dụng một phép biến đổi mã hoá (encryption), tác động lên X, để chế biến ra một bản mã Y (cryptogram, hay ciphertext), không thể đọc được. Ta nói bản mã Y đã che giấu nội dung của bản rõ X bản đầu. Giải mã (decryption) là quá trình ngược lại cho phép người nhận thu được bản rõ X từ bản mã Y.

Để bảo mật, các khối biến đối sinh và giải mã là các hàm toán học với tham số khoá (key). Khóa là thông số điều khiển mà sở hữu kiến thức về nó thông thường là hạn chế. Thông thường khoá (Z) chỉ được biết đến bởi các bên tham gia truyền tin S và R.

## 1.5 Hệ thống mật mã đối xứng (Symmetric Key Cryptosystem - SKC) là gì?

Trong mô hình của hệ thống này, khóa của hai thuật toán sinh mã và giải mã là giống nhau và bí mật đối với tất cả những người khác; nói cách khác, hai bên gửi và nhận tin chia sẻ chung một khóa bí mật duy nhật. Vai trò của hai phía tham gia là giống nhau và có thể đánh đổi vai trò, gửi và nhận tin, cho nên hệ thống được gọi là “mã hóa đối xứng”. Chúng ta sẽ sử dụng ký hiệu viết tắt theo tiếng Anh là SKC.

## 1.6 Nhược điểm của hệ thống mật mã đối xứng?

Hệ thống mật mã khóa bí mật đối xứng có những nhược điểm lớn trên phương diện quản lý và lưu trữ, đặc biệt bộc lộ rõ trong thế giới hiện đại khi liên lạc qua Internet đã rất phát triển. Nếu như trong thế giới trước kia liên lạc mật mã chỉ hạn chế trong lĩnh vực quân sự hoặc ngoại giao thì ngày nay các đối tác doanh nghiệp khi giao dịch qua Internet đều mong muốn bảo mật các thông tin quan trọng. Với hệ thống khóa bí mật, số lượng khóa bí mật mà mỗi công ty hay cá nhân cần thiết lập với các đối tác khác có thể khá lớn và do đó rất khó quản lý lưu trữ an toàn các thông tin khóa riêng biệt này.

Một khó khăn đặc thù khác nữa là vấn đề xác lập và phân phối khóa bí mật này giữa hai bên, thường là đang ở xa nhau và chỉ có thể liên lạc với nhau qua một kênh truyền tin thông thường, không đảm bảo tránh được nghe trộm.

## 1.7 Thành phần của hệ thống mật mã khóa công khai hay phi đối xứng (Public Key Cryptosystem – PKC)

* Thành phần khóa công khai, có thể đăng ký phổ biến rộng khắp, dùng để sinh mã hoặc để xác thực chữ ký điện tử
* Thành phần khóa bí mật, chỉ dành riêng cho bản thân, dùng để giải mã hoặc tạo ra chữ ký điện tử.

Chỉ với cặp khóa này, thực thể chủ có thể giao dịch bảo mật với quảng đại xã hội, trong đó việc quản lý và lưu trữ có thể được tổ chức chặt chẽ mà việc phải tự nhớ thông tin mật là tối thiểu (giống như việc chỉ nhớ 1 mật khẩu hay một số PIN tài khoản ngân hàng).

## 1.8 Cách đánh giá tính bảo mật của các hệ mật mã?

Ta có thể kết luận một hệ mã mật là *không an toàn* (insecure), bằng việc chỉ ra cách phá nó trong một mô hình tấn công (khái niệm sẽ giới thiệu sau đây) phổ biến, trong đó ta chỉ rõ được các mục tiêu về ATBM (security) không được đảm bảo đúng. Tuy nhiên để kết luận rằng một hệ mã là an toàn cao thì công việc phức tạp hơn nhiều. Thông thường, người ta phải đánh giá hệ mật mã này trong nhiều mô hình tấn công khác nhau, với tính thách thức tăng dần. Để có thể khẳng định tính an toàn cao, cách làm lý tưởng là đưa ra một chứng mình hình thức (formal proof), trong đó người ta chứng minh bằng công cụ toán học là tính ATBM của hệ mã đang xét là tương đương với một hệ mã kinh điển, mà tính an toàn của nó đã khẳng định rộng rãi từ lâu.

Như đã nói trên, người ta phủ định tính an toàn của một hệ mã mật thông qua việc chỉ ra cách phá cụ thể hệ mã này trên một mô hình tấn công (attack model) cụ thể. Mỗi mô hình tấn công sẽ định nghĩa rõ năng lực của kẻ tấn công, bao gồm năng lực tài nguyên tính toán, loại thông tin mà nó có khả năng tiếp cận để khai thác và khả năng tiếp xúc với *máy mật mã* (thiết bị phần cứng có cài đặt thuật toán sinh và giải mã). Các mô hình tấn công thường được sắp xếp theo thứ tự mạnh dần của năng lực kẻ tấn công. Nếu một hệ mật mã bị phá vỡ trong một mô hình tấn công căn bản (năng lực kẻ tấn công là bình thường) thì sẽ bị đánh giá là hoàn toàn không an toàn. Sau đây là một số mô hình tấn công phổ biến.

## 1.9 Kể tên các mô hình tấn công phổ biến và cách hoạt động?

*Tấn công chỉ-biết-bản-mã* (*ciphertext-only attack*). Ở đây kẻ địch E chỉ là một kẻ hoàn toàn bên ngoài, tìm cách nghe trộm trên đường truyền để lấy được các giá trị Y, bản mã của thông tin gửi đi. Mặc dù kẻ địch E chỉ biết các bản rõ Y, nhưng mục tiêu nó hướng tới là khám phá nội dung một/nhiều bản rõ X hoặc lấy được khóa mật Z (trường hợp phá giải hoàn toàn). Đây là mô hình tấn công căn bản nhất trong đó kẻ địch không có năng lực quan hệ đặc biệt (như một số hình thức tấn công sau), diện thông tin tiếp xúc chỉ là các bản mã. Rõ ràng nếu một hệ mã mà không đứng vững được trong mô hình này thì phải đánh giá là không đáng tin cậy.

*Tấn công biết-bản-rõ* (*known-plaintext attack*). Mặc dù tên gọi hơi dễ hiểu nhầm, thực chất trong mô hình này ta chỉ giả thiết là E có thể biết một số cặp X-Y (bản rõ và bản mật tương ứng) nào đó. Nguyên nhân E thu được có thể hoàn toàn tình cờ hoặc nhờ một vài tay trong là nhân viên thấp cấp trong hệ thống. Tất nhiên mục tiêu của E là khám phá nội dung các bản rõ quan trọng khác và/hoặc lấy được khóa mật. Rõ ràng mô hình tấn công này làm mạnh hơn so với *tấn công chỉ qua bản mã*: Việc biết một số cặp X-Y sẽ làm bổ sung thêm đầu mối phân tích; đặc biệt từ bây giờ E có thể dùng phép thử loại trừ để vét cạn không gian khóa (*exshautive key search*) và tìm ra khóa đúng tức là sao cho Enc (K,X)=Y.

*Tấn công bản-rõ-chọn*-*sẵn* (*chosen-plaintext attack*). Trong mô hình này, không những E thu nhặt được một số cặp X-Y mà một số bản rõ X do bản thân E soạn ra (chosen plaintext). Điều này thoạt nghe có vẻ không khả thi thực tế, tuy nhiên ta có thể tưởng tượng là E có tay trong là một thư ký văn phòng của công ty bị tấn công, ngoài ra do một qui định máy móc nào đó tất cả các văn bản dù quan trọng hay không đều được truyền gửi mật mã khi phân phát giữa các chi nhánh của công ty này. Có thể nhận xét thấy rằng, việc được tự chọn giá trị của một số bản rõ X sẽ thêm nhiều lợi ích cho E trong phân tích quan hệ giữa bản mã và bản rõ để từ đó lần tìm giá trị khóa.

Một cách tương tự, người ta cũng sử dụng mô hình *tấn công bản-mã-chọn-sẵn* (*chosen-ciphertext attack*) trong đó kẻ địch có thể thu nhặt được một số cặp X-Y mà Y là giá trị được thiết kế sẵn. Trong thực tế điều này có thể xảy ra nếu như kẻ địch có thể truy nhập được vào máy mật mã 2 chiều (có thể sử dụng với cả 2 chức năng là sinh mã và giải mã). Tất nhiên cả hai dạng tấn công rất mạnh nói trên kẻ thù đều có thể khôn ngoan sử dụng một chiến thuật thiết kế bản rõ (hay bản mã) chọn sẵn theo kiểu thích nghi (adaptive), tức là các bản rõ chọn sau có thể thiết kế dựa vào kiến thức phân tích dựa vào các cặp X-Y đã thu nhặt từ trước.

## 1.10 Các mô hình đánh giá với các mức độ mạnh đến yếu

*Bảo mật vô điều kiện* (*unconditional security*): Đây là mô hình đánh giá ATBM mức cao nhất, trong đó “vô điều kiện” được hiểu theo ý nghĩa của lý thuyết thông tin (information theory), trong đó các ý niệm về “lượng tin” được hình thức hóa thông qua các phép toán xác suất. Trong mô hình này, kẻ địch được coi là không bị hạn chế về năng lực tính toán, tức là có thể thực hiện bất kỳ khối lượng tính toán cực lớn nào đặt ra trong khoảng thời gian ngắn bất kỳ. Mặc dù có năng lực tính toán siêu nhiên như vậy, mô hình này chỉ giả thiết kẻ tấn công là người ngoài hoàn toàn (tức là ứng với mô hình tấn công chỉ-biết-bản-mã). Một hệ mật mã đạt được mức an toàn vô điều kiện, tức là có thể đứng vững trước sức mạnh của một kẻ địch bên ngoài (chỉ biết bản mã) có khả năng không hạn chế tính toán, được gọi là đạt đến *bí mật tuyệt đối* (*perfect secretcy*).

Một cách khái quát, việc nghe trộm được bản mã đơn giản là chỉ cung cấp một lượng kiến thức zero tuyệt đối, không giúp gì cho việc phá giải mã của kẻ địch. Việc biết bản mã không đem lại chút đầu mối gì cho khả năng lần tìm ra khóa của hệ mã.

*Bảo mật chứng minh được* (*provable security*): Đây cũng là một mô hình đánh giá mức rất cao, lý tưởng trong hầu hết các trường hợp. Một hệ mật mã đạt được mức đánh giá này đối với một mo hình tấn công cụ thể nào đó, nếu ta có thể chứng mình bằng toán học rằng tính an toàn của hệ mật là được qui về tính NP-khó của một bài toán nào đó đã được biết từ lâu (ví dụ bài toán phân tích ra thừa số nguyên tố, bài toán cái túi, bài toán tính logarit rời rạc ...). Nói một cách khác ta phải chứng minh được là kẻ thù muốn phá được hệ mã thì phải thực hiện một khối lượng tính toán tương đương hoặc hơn với việc giải quyết một bài toán NP-khó đã biết.

*Bảo mật tính toán được*, hay *bảo mật thực tiễn* (*computational security* hay *practical security*): Đây là một trong những mức đánh giá thường được áp dụng nhất trong thực tế (khi những mức bảo mật cao hơn được cho là không thể đạt tới). Khi đánh giá ở mức này với một hệ mã cụ thể, người ta lượng hóa khối lượng tính toán đặt ra để có thể phá hệ mã này, sử dụng kiểu tấn công mạnh nhất đã biết (thường kèm theo đó là mô hình tấn công phổ biến mạnh nhất). Từ việc đánh giá được khối lượng tính toán này cùng thời gian thực hiện (với năng lực kẻ địch mạnh nhất có thể trên thực tế), và so sánh với thời gian đòi hỏi đảm bảo tính mật trên thực tế, ta có thể đánh giá hệ mã có đạt an toàn thực tiễn cao hay không. Đôi khi, cơ sở đánh giá cũng dựa vào một bài toán khó nào đó mặc dù không đưa ra được một chứng minh tương đương thực sự

*Bảo mật tự tác* (*ad hoc security*): Một số hệ mật mã riêng được một số công ty hoặc cá nhân tự chế để phục vụ mục đích đặc biệt dùng nội bộ. Tác giả loại hệ mật mã có thể sử dụng những lập luận đánh giá hợp lý nhất định dựa trên việc ước đoán khối lượng tính toán của kẻ địch khi sử dụng những tấn công mạnh nhấn đã biết và lập luận về tính bất khả thi thực tiễn để thực hiện. Mặc dù vậy hệ mật mã này vẫn có thể bị phá bởi những tấn công có thể tồn tại mà chưa được biết tới đến thời điểm đó; vì vây, thực tế bảo mật ở mức này hàm nghĩa không có một chứng minh đảm bảo thực sự, nên không thể coi là tin cậy với đại chúng.

## 1.11 Các loại mật mã

*Mật mã cộng (Additive cipher) - Mật mã Xeda (Ceasar)*

Mật mã cộng (Additive cipher) là một mật mã một bảng thế đặc biệt trong đó, phép biến đổi mã được biểu diễn thông qua phép cộng đồng dư như sau. Giả sử ta gán các giá trị từ A-Z với các số 1-25,0. Thế thì một chữ plaintext X có thể mã thành ciphertext Y theo công thức:

*Y = X*  *Z,*

trong đó Z là giá trị của khoá,  là ký hiệu phép cộng đồng dư modulo 26.

Mật mã nhân tính (multiplicative cipher)

Bảng thế cũng có thể được xây dựng từ phép nhân đồng dư của chữ cái trong bảng gốc với giá trị của khóa:

*Y=X**Z*

Khoảng đầu thiên nhiên kỷ thứ hai, mật mã một bảng thế đã bị phá và các nhà khoa học đã dần nghĩ đến các nguyên tắc thiết kế mã tốt hơn, nhằm tránh bảo tồn các qui luật thống kê từ TIN sang MÃ (bản rõ sang bản mã). Ta sẽ xem xét một số mã như vậy sau đây.

*Mã với bảng thế đồng âm (homophonic substitution ciphers)*

Trong các cipher loại này, ánh xạ chữ cái TIN- MÃ không còn là 1-1 nữa mà là một-nhiều. Tức là mỗi chữ của bảng chữ cái tin sẽ được mã hoá thành 1 chữ trong 1 tập con các chữ mã nào đó. Mỗi chữ mã trong tập con này được gọi là homophone, tạm dịch là đồng âm.

Thông thường người ta bố trí số lượng đồng âm ứng với mỗi chữ tin tỷ lệ với tần xuất xuất hiện của chữ đó trong ngôn ngữ tự nhiên. Vì vậy đồ thị tần xuất của các chữ cái trong bản mã sẽ trở nên bằng phẳng. Mặc dù các cipher loại này là khó phá hơn nhưng chúng lại bị tăng thêm độ dư thừa so với tin gốc.

*Sử dụng nhiều bảng thế (mã đa bảng thế)*

## 1.12 Phương pháp giải Vigenere cipher ?

Trong Vigenere Cipher, người ta dùng tất cả 26 bảng thế là sự thuđược từ bảng gốc chữ cái tiếng Anh mà dịch đi từ 0-25 vị trí. Sự hoà trộn này có quy luật hoàn toàn xác định bởi khoá. Mỗi chữ của khoá sẽ xác định mỗi bảng thế được dùng.

*Phương pháp giải mã Vigenere.*

Ý tưởng của phương pháp này gồm 3 bước như sau:

1. Đi tìm chu kỳ p (độ dài khoá)
2. Chia tách MÃ thành p đoạn phân mã, mỗi đoạn bao gồm các chữ ở vị trí kp+i (k=1,2,3 ... ; i=0,p-1), tức là được mã hoá theo bảng thế với chữ khoá chỉ số *i*.
3. Dùng phương pháp một bảng thế đã biết để giải từng đoạn phân mã (cụ thể là với mã Vigenere chỉ cần một phép dịch đúng)

Người ta sử dụng khái niệm IC (Index of Coincidence) để tính chu kỳ p.

Theo định nghĩa, IC xác định qua công thức:

25 *fi* (*fi -1*)

 i=0

IC = ----------------- n(n-1)

Trong đó *f* là xác xuất của phép thử - nhặt ra 2 con chữ ngẫu nhiên bất kỳ từ trong một đoạn văn bản - để thu được cùng một chữ  cho trước.

Phương pháp thực hành

1. Đặt k=1
2. Kiểm tra xem p có phải nhận giá trị k hay không.
   1. Chia Mã thành k phân mã và tính IC của các phân mã.
   2. Nếu như chúng đều xấp xỉ nhau và đều xấp xỉ 0.068 thì p=k Nếu chúng khác nhau nhiều và nhỏ hơn nhiều so với 0.068 thì p>k
3. Tăng k lên một đơn vị và lặp lại bước 2.

## 1.13 One-time-pad (Vernam cipher) là gì?

Mật mã One-time-pad được đề xuất bởi G. Vernam (1917); sau đó đã được chứng minh là đảm bảo bí mật tuyệt đối (perfect secretcy - 1949). Như tên gọi của nó, trong One-time-pad khóa được viết trên 1 băng (tape) dài, và sử dụng đúng 1 lần. Đồng thời chuỗi khóa là chuỗi văn bản sinh ngẫu nhiên, có độ dài bằng văn bản sử dụng hoặc hơn. Thao tác mã hóa đơn giản là phép dịch theo bảng thế ứng với chữ khóa tương ứng hoặc XOR nếu xử lý theo chuỗi nhị phân.

Sinh mã: Y = X + Z (mod 26)

Giải mã : X = Y - Z (mod 26)

Vì vậy, One-time-pad có thể coi là mã Vigenere với khóa là một chuỗi ngẫu nhiên có độ dài đúng bằng văn bản, như ví dụ sau sẽ cho thấy

Chú ý rằng khóa chỉ được dùng đúng một lần, tức là vứt bỏ sau khi dùng. Nếu dùng lại thì không còn đảm bảo an toàn nữa.

## 1.14 Khái niệm bí mật tuyệt đối

Trong hệ thống *đảm bảo bí mật tuyệt đối*, bản mã bị tiết lộ cho kẻ thù không hề đem lại một ý nghĩa nào cho phân tích tìm khóa phá mã. Sự kiện nghe trộm bản mã (có độ dài bất kỳ) sẽ không làm thay đổi phân phối xác xuất ban đầu của plaintext.

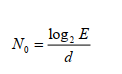
*Hay là, một hệ thống là có bí mật tuyệt đối nếu:*

*P(X) = P(X/Y)*  *TIN X VÀ MÃ Y*

**Định lý Shannon.** Trong hệ thống có BMTĐ, số lượng khoá có thể (độ lớn không gian khoá) phải lớn hơn hoặc bằng số lượng thông báo có thể (độ lớn không gian TIN).

Điều này cho thấy để đạt được BMTĐ thì khoá phải rất dài, do đó việc trao chuyển khoa giữa hai bên truyền tin sẽ làm cho hệ thống trở nên phi thực tế. Như vậy, nhìn chung chúng ta không thể đạt được bí mật tuyệt đối mà chỉ có thể có được các hệ thống với mức an toàn thực tế (Practical security) được cài đặt tuỳ theo giá trị của thông tin cần bảo vệ và thời gian sống của nó.

## 1.15 Đánh giá mức độ bảo mật của một cipher?

Shannon đưa ra một khái niệm, *unicity distance*, để “đo” mức an toàn của một hệ mã: Unicity distance, ký hiệu N0, là độ dài tối thiểu của bản mã nghe trộm được để có thể xác định được khóa đúng duy nhất. Unicity distance có thể được tính theo công thức:

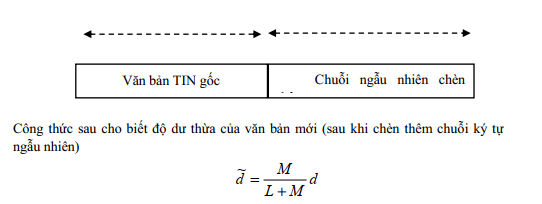
Trong đó d là *độ dư thừa* của ngôn ngữ sử dụng của TIN.

## 1.16 Ta có thể “tăng” tính mật của một hệ mã cho trước hay không?

* Tăng độ lớn không gian khóa
* Giảm tính dư thừa của ngôn ngữ văn bản TIN: tiền xử lý qua 1 bước thuật toán nén

Chú ý: một thuật toán nén lý tưởng có thể đem lại độ dư thừa 0, do đó N0  0

* Có thể chèn thêm một đoạn văn bản ngẫu nhiên để “phẳng hóa“ độ thị tần xuất của văn bản TIN. Ta sẽ xét cụ thể biện pháp này dưới đây



# Chương II: MẬT MÃ KHỐI VÀ MẬT MÃ KHÓA ĐỐI XỨNG

## 2.1 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 đối 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).

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.

## 2.2 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.

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

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).

## 2.3 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:

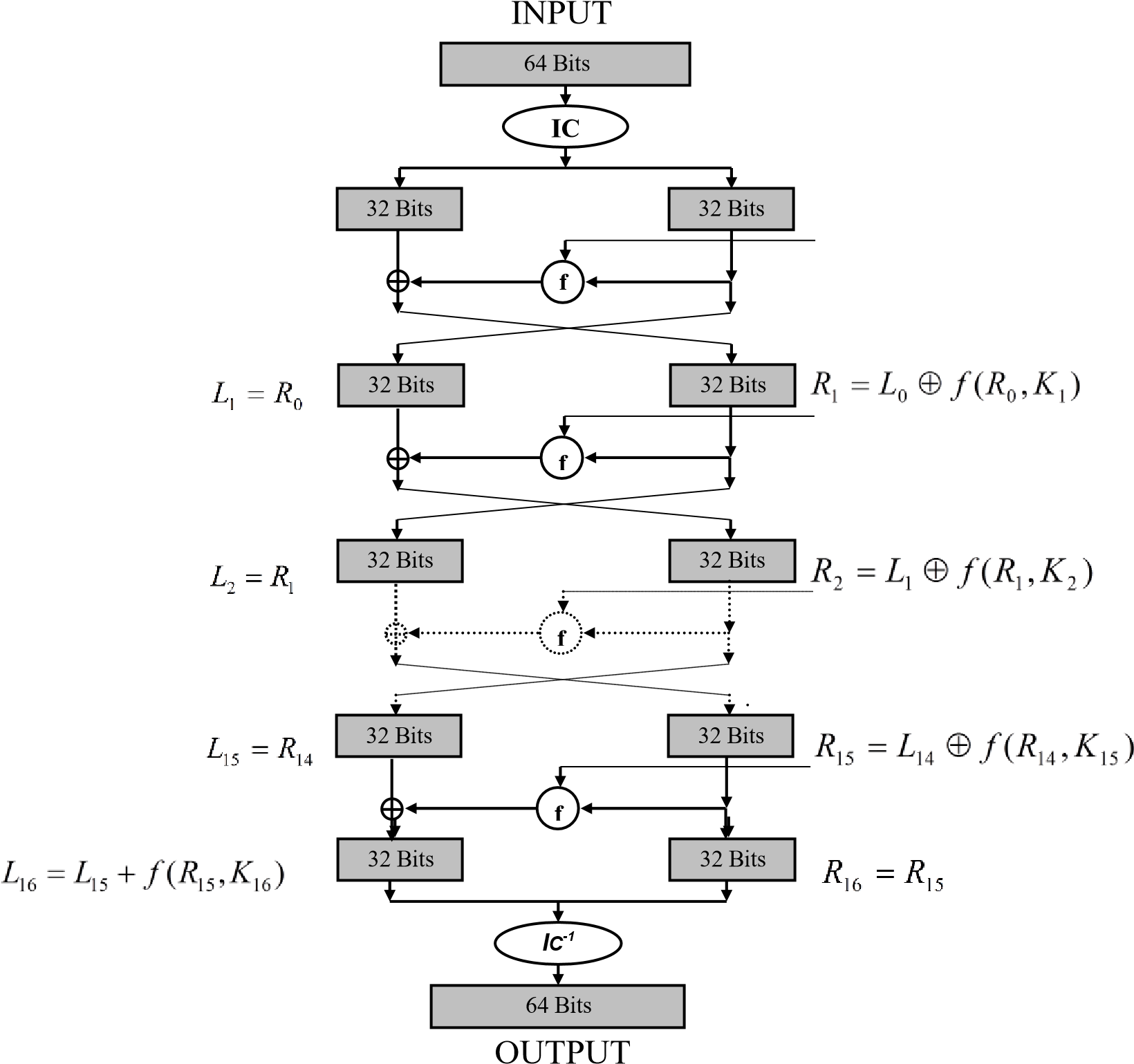
* Sự phát triển của công nghệ thông tin và của nhu cầu an toàn & bảo mật thông tin: sự ra đời của các mạng máy tính tiền thân của Internet đã cho phép khả năng hợp tác và liên lạc số hóa giữa nhiều công ty, tổ chức trong các dự án lớn của chính phủ Mỹ.
* Các thuật toán „cây nhà lá vườn‟ (ad hoc) không 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

## 2.4 Các thuật toán và lưu đồ hoạt động của DES

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.

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

(Li,Ri) = (Ri-1, Li-1  *f* (Ri-1,Ki))

trong đó, (Li,Ri) là 2 nửa trái và phải thu được từ biến đổi của vòng lặp thứ *i*. Ta cũng có thể viết lại

(Li,Ri) = T F (Ri-1,Ki))

Trong đó *F* là phép thay thế Li-1 bằng Li-1  *f* (Ri-1,Ki), 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F16TF15T ... F2TF1 (IP)

*Thuật toán giải mã*

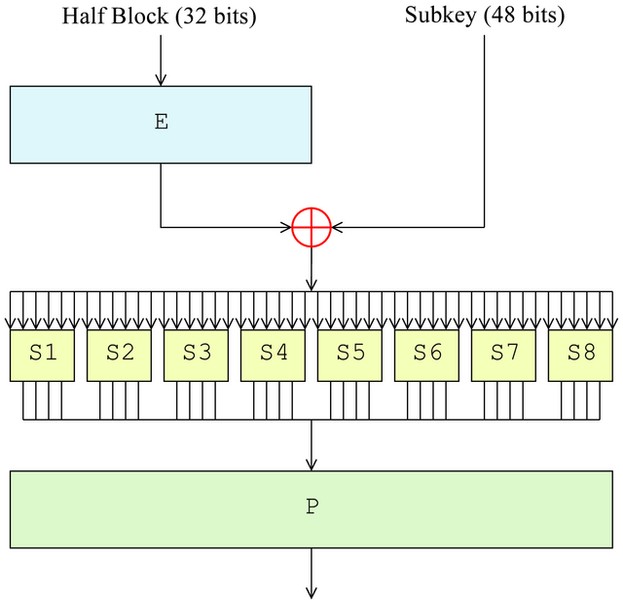
*DES* được xây dựng giống hệt như thuật toán sinh mã nhưng có các khóa con được sử dụng theo thứ tự ngược lại, tức là dùng khóa K16 cho vòng lặp 1, khóa K15 cho vòng lặp 2 ... Vì vậy, thuật toán giải mã có thể được viết lại dưới dạng công thức sau:

DES-1 = (IP)-1F1TF2T ... F15TF16 (IP) -1,

Bây giờ chú ý rằng mỗi hàm T hoặc F đều là các hàm có tính chất đối hợp (f=f hay f(f(x) =x). Do đó nếu ta thực hiện phép tích hàm DES-1DEShay DES 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.

*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 Ri-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 Ki. 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 Fi).



## 2.5 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 S1,S2 ... S8. 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.

**Middle 4 bits of input**

**S5**

**0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111**

**00** 0010 1101 0000 1110 1001

**Outer 01** 1110 1011 0010 1100 0100 0111 1101 0001 0101 0000 1111 1010 0011 1001 1000 0110 **bits 10** 0100 0010 0001 1011 1010 1101 0111 1000 1111 1001 1100 0101 0110 0011 0000 1110

1100

0100

0001

0111

1010

1011

0110

1000

0101

0011

1111

**11** 1011 1000 1100 0111 0001 1110 0010 1101 0110 1111 0000 1001 1010 0100 0101 0011

## 2.6 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 vây, NSA đã tiết lộ 3 thuộc tính của S-boxes, những thuộc tính này bảo đảm tính confusion & diffusion của thuật toán.

1. Các bit 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 256 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.

## 2.7 Các điểm yếu của DES

- Tính bù.

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

y = DESz (x)  y  DESz(x)

Cho nên nếu biết MÃ y được mã hóa từ TIN x với khóa z thì ta suy ra y được mã hóa từ TIN x với khóa 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.

* Khóa yếu

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

Z1 = Z2 = Z3 = ...=Z15 = Z16

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

DESz = DES-1z

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]

1. [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-1z = DESz‟ hay là DES-1z‟ = DESz

## 2.8 Cách tấn công bằng phương pháp vét cạn (hay là brute-force attack)

DES có 256=1017 khóa. Nếu như biết một cặp plaintext-ciphertext thì chúng ta có 17 khả năng này để tìm ra khóa cho kết quả khớp. Giả sử như một phép thể thử tất cả 10

-6 11 thử mất quãng 10 s (trên một máy PC thông thường), thì chúng ta sẽ thử mất 10 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

7 con chip mật mã DES chạy song song thì bây giờ mỗi con chip chỉ một thiết bị với 10

10 phép thử. Chip mã DES ngày nay có thể xử lý phải chịu trách nhiệm tính toán với 10

7 5 phép mã DES trong một tới tốc độ là 4.5 x 10 bits/s tức là có thể làm được hơn 10 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.

## 2.9 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 49 bản rõ chọn trước (chosen plaintext). Để đầy đủ 16 vòng lặp, tấn công này cần tới 2 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, 43 bản rõ chọn trước và cần 2

## 2.10 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,
* Blowfísh,
* 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.

## 2.11 Mật mã AES là gì?

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.

## 2.12 Các chế độ sử dụng mã khối

* *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.

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.

* *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.

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).

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

* *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.

* *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.

* *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 hòan tòan 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.

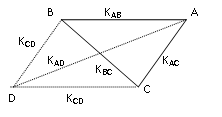
# Chương III

## 3.1 Các cách gọi đối với hệ mật mã?

Các hệ mật mã khóa đối xứng (Symmtric Key Cryptosystems) do vai trò hai bên gửi và nhận tin đều như nhau vì đều sở hữu chung một khoá bí mật. Cũng có nhiều cách gọi khác đối với các hệ mật mã này, sử dụng tùy vào các ngữ cảnh phù hợp:

* Hệ mã với khóa sở hữu riêng (Private Key Cryptosystems)
* Hệ mã với khóa bí mật (Secret Key Cryptosystems)  Hệ mã truyền thống (Conventional Cryptosystems)

Chúng ta sẽ sử dụng ký hiệu viết tắt cho hệ mật mã đối xứng là SKC.



## 3.2 Nhược điểm của hệ mã đối xứng?

* Vấn đề quản lý khoá (tạo, lưu mật, trao chuyển ...)
* Là rất phức tạp khi sử dụng trong môi trường trao đổi tin giữa rất nhiều người dùng.
* Với số lượng NSD là *n* thì số lượng khoá cần tạo lập là *n(n-1)/2*.
* Mỗi người dùng phải tạo và lưu *n-1* khoá bí mật để làm việc với *n-1* người khác trên mạng.
* Như vậy rất khó khăn và không an toàn khi *n* tăng lớn.
* Trên cơ sở mã đối xứng, ta không thể thiết lập được khái niệm chữ ký điện tử (mà thể hiện được các chức năng của chữ ký tay trong thực tế) và cũng do đó không có dịch vụ non-repudiation (không thể phủ nhận được) cho các giao dịch thương mại trên mạng.

## 3.3 Hoạt động của hệ mã đối xứng?

Trong hệ thống mới này, mỗi NSD có hai khoá, một được gọi là khoá bí mật (secret key hay private key) và một được gọi là khoá công khai (public key). Khoá thứ nhất chỉ mình user biết và giữ bí mật, còn khoá thứ hai thì anh ta có thể tự do phổ biến công khai. Khoá thứ nhất thường đi liền với thuật toán giải mã, còn khoá thứ hai thường đi liền với thuật toán sinh mã, tuy nhiên điều đó không phải là bắt buộc. Ta hãy ký hiệu chúng là z (khóa riêng) và Z (khóa công khai)

Hoạt động của chúng là đối xứng

X = D(z, E(Z, X)) (1)

và X = E(Z, D(z, X)) (2)

Trong đó hệ thức (1) biểu tượng cho bài toán truyền tin mật: bất kỳ NSD nào khác như B, C, D ... muốn gửi tin cho A chỉ việc mã hoá thông tin với khoá công khai (ZA) của A rồi gửi đi. Chỉ có A mới có thể khoá riêng để giải mã (zA) và đọc được tin; kẻ nghe trộm Eve không thể giải mã để lấy được tin vì không có khoá zA.

Còn hệ thức (2) sẽ được sử dụng để xây dựng các hệ chữ ký điện tử như sau này ta sẽ nghiên cứu, trong đó thao tác Ký chính là thực hiện E(ZA) còn kiểm định chữ ký là thông qua gọi D(zA).

Hệ mật mã theo nguyên tắc nói trên được gọi là hệ mã với khoá công khai (public key cryptosystems) hay còn được gọi là mã khóa phi đối xứng (asymmetric key cryptosystems). Ta sẽ viết tắt hệ thống kiểu này bằng PKC.

## 3.4 Nguyên tắc cấu tạo một hệ PKC sử dụng cửa bẫy (trapdoor) ?

Một hệ mã PKC có thể được tạo dựng trên cơ sở sử dụng một hàm một chiều (one-way). Một hàm *f* được gọi là một chiều nếu:

* Đối với mọi X tính ra Y = *f*(X) là dễ dàng.
* Khi biết Y rất khó để tính ngược ra X.

Tuy nhiên, chúng ta cần một hàm một chiều đặc biệt có trạng bị một cửa bẫy (trap door) sao cho nếu biết sử dụng nó thì việc tìm nghịch đào của *f* là dễ dàng, còn nếu không (không biết bí mật cửa bẫy) thì vẫn khó như thường.

Một hàm một chiều có cửa bẫy như thế có thể dùng để tạo ra một hệ mã PKC như sau. Lấy *EZ* (hàm sinh mã) là hàm một chiều có cửa bẫy này. Như vậy bí mật cửa bẫy chính là khóa bí mật *z*, mà nếu biết nó thì có thể dễ dàng tính được cái nghịch đảo của *EZ* tức là biết *Dz*, còn nếu không biết thì rất khó (chỉ còn cách thử vét cạn, thực tế sẽ là bất khả thi vì khối lượng tính toán quá lớn).

## 3.5 Bài toán đóng thùng

*Cho 1 tập hợp các số dương ai, 1**i**n và một số T dương. Hãy tìm một tập hợp chỉ số S* *1,2,...,n*  *sao cho:*  *i**S ai = T*

Từ bài toán *Đóng thùng* này chúng ta sẽ khảo sát các khả năng vận dụng để tạo ra thuật toán mã khối PKC. Sơ đồ đầu tiên như sau:

Chọn một vector *a = (a1, a2, ... , an)* - được gọi là vector mang (cargo vector) Với một khối tin *X = (X1,X2,X3 ..., Xn)*, ta thực hiện phép mã hoá như sau:

*T=*  *aiXi* (\*)

*i=1,n*

Việc giải mã là: Cho mã T, vector mang a, tìm các Xi sao cho thoả mãn (\*).

Merkle áp dụng một mẹo dựa trên sử dụng vector mang đặc biệt là vector siêu tăng (super-increasing) như sau. Một vectơ là siêu tăng nếu thành phần *i+1* là lớn hơn tổng giá trị của các thành phần đứng trước nó (1i). Khi sử dụng một vector siêu tăng làm vector mang thì sẽ thấy việc tính ngược, tức là giải bài toán đóng thùng là dễ dàng nhờ một giải thuật thăm ăn đơn giản. Điều này được minh họa qua ví dụ bằng số sau.

Ở bước *i*, tổng đích là *Ti* (tức là phải tìm các *aj* để tổng bằng *Ti*). Ta đem so sánh *Ti* với thành phần lớn nhất trong phần còn lại của vector, nếu lớn hơn thì thành phần này được chọn tức là *Xi* tương ứng bằng 1, còn ngược lại thì *Xi* tương ứng bằng 0. Sau đó tiếp tục chuyển sang bước sau với *Ti+1 = Ti-Xi.*

Mặc dù ta đã thấy sử dụng vector siêu tăng là vector mang cho phép giải mã dễ dàng nhưng, tất nhiên, ta còn phải làm thế nào để cho chỉ có người chủ mới biết được và sử dụng nó còn kẻ thù thì không. Tóm lại, cần tạo ra một bí mật cửa bẫy thông qua việc người chủ phải chủ động “nguỵ trang” vector siêu tăng để chỉ có anh ta mới biết còn người ngoài không thể lần ra được.

## 3.6 Thuật toán Merkle-Hellman

Sơ đồ sau đây sẽ trình bày một cơ chế nguỵ trang như vậy. Vector *a’* là một vector siêu tăng bí mật, sẽ được “ngụy trang”, tức là biến đối thông qua một hàm *g* được chọn sẵn để tạo thành vector *a* không hề có tính siêu tăng (thậm chí là có thể giảm); vector *a* này sẽ được sử dụng làm vector mang. Trong quá trình giải mã, người chủ (Alice) sẽ thực hiện một biến đổi vào dữ liệu, trên cơ sở áp dụng hàm ngược *g*-1, chuyển việc giải mã thành giải một bài toán đóng thùng với vector siêu tăng là vector mang. Phép biến đổi *g* được chọn chính là phép nhân đồng dư với một giá trị khóa bí mật.

*Tạo khoá*:

* Alice chọn một vector siêu tăng:

*a’ = (a1’,a2’,...,an’)*

*a’* được giữ bí mật tức là một thành phần của khoá bí mật

* Sau đó chọn một số nguyên *m >* ∑ *ai’*, gọi là mo-dul đồng dư và một số nguyên ngẫu nhiên , gọi là nhân tử, sao cho nguyên tố cùng nhau với m.

Khoá công khai của Alice sẽ là vector *a* là tích của *a’* với nhân tử : *a = (a1,a2,...,an) ai=**ai’ (mod m); i=1,2,3...n*

Còn khoá bí mật sẽ là bộ ba *(a’, m,* ω*)*

*Sinh mã:*

Khi Bob muốn gửi một thông báo *X* cho Alice, anh ta tính mã theo công thức:

*T=*∑ *aiXi*

## 3.7 Thuật toán tìm giá trị nghịch đảo theo modul đồng dư

Việc xây dựng Knapsack với cửa bẫy đòi hỏi phải tính giá trị nghịch đảo của ω-1 mod m, sao cho x. ω = 1 (mod m) được gọi là theo modul m.

Thuật toán tìm x =  thuật toán GCD mở rộng hay Euclide mở rộng (GCD - Greatest common divior - ước số chung lớn nhất).

Sở dĩ như vậy là vì trong khi đi tìm ước số chung lớn nhất của hai số nguyên n1 và n2, người ta sẽ tính luôn các giá trị a,b sao cho GCD(n1, n2) = a\*n1 + b\*n2.

Từ đó suy ra nếu ta đã biết (n1, n2)=1 thì thuật toán này sẽ cho ta tìm được a, b thoả mãn a\*n1 + b\*n2=1, tức là n1 chính là nghịch đảo của a theo modulo n2 (tức là m)

## 3.8 Mục đích của việc sử dụng hệ thống PKC

Một hệ thống PKC có thể sử dụng vào 2 mục đích cơ bản:

- Bảo mật thông tin và truyền tin

- Chứng thực và chữ ký điện tử. Hai thuật toán đáp ứng các ứng dụng trên thành công nhất là RSA và Elgamal. Nói chung thuật toán PKC là chậm và không thích hợp cho mật mã trên dòng (online) với truyền tin tốc độ cao, vì vậy chỉ thường được sử dụng khi cần đến tính an toàn cao và chấp nhận tốc độ chậm.

Ngoài ra người ta thường sử dụng kết hợp PKC và SKC (symmetric key cryptosystems) với PKC có tác dụng “khởi động mồi” cho SKC: dùng PKC để thiết lập thuật toán tạo ra khoá bí mật thống nhất chung giữa hai bên truyền tin sau đó sử dụng khoá bí mật trên cho pha truyền tin chính bằng SKC sau đó.

## 3.9 Hệ thống khóa công khai RSA là gì?

RSA là hệ mật mã khóa công khai phổ biến và cũng đa năng nhất trong thực tế, phát minh bởi Rivest, Shamir & Adleman (1977). Nó là chuẩn mật mã bất thành văn đối với PKC, cung cấp đảm bảo tính mật, xác thực và chữ ký điện tử.

Cơ sở thuật toán RSA dựa trên tính khó của bài toán phân tích các số lớn ra thừa số nguyên tố: không tồn tại thuật toán thời gian đa thức (theo độ dài của biểu diễn nhị phân của số đó) cho bài toán này. Chẳng hạn, việc phân tích một hợp số là tích của 2 số nguyên tố lớn hàng trăm chữ số sẽ mất hàng ngàn năm tính toán với một máy PC trung bình có CPU khoảng trên 2Ghz.

## 3.10 Thuật toán RSA

Xây dựng: Chọn các tham số

1. *Chọn hai số nguyên tố lớn p và q. Tính n = p x q và m =* *(n) = (p = 1) x (q1).*
2. *Chọn e, 1* *e*  *m -1, sao cho gcd (e, m) = 1.*
3. *Tìm d sao cho e \* d = 1 (mod m), tức là tính d = e-1 (mod m), giải theo thuật toán gcd mở rộng đã trình bày ở phần trước.*

*Khóa công khai (Public key) là (e, n)*

*Khoá dùng riêng (Private key) là d, p, q)*

Giả sử X là một khối tin gốc (plaintext), Y là một khối mã tương ứng của X, và (*zA*,*ZA*)là các thành phần công khai và riêng của khoá của Alice

Sinh Mã. Nếu Bob muốn gửi một thông báo mã hoá cho Alice thì anh ta chỉ việc dùng khoá công khai của Alice để thực hiện:

*Y**EZA* (*X*) *X e* *n*

Giải mã: Khi Alice muốn giải mã Y, cô ta chỉ việc dùng khoá riêng zA = d để thực hiện như sau:

*DzA* (*Y*) *Y d* *n*

Để tiện cho việc giao dịch trên mạng có sử dụng truyền tin mật, người ta có thể thành lập các Public Directory (thư mục khoá công khai), lưu trữ các khoá công khai của các user. Thư mục này được đặt tại một điểm công cộng trên mạng sao cho ai cũng có thể truy nhập tới được để lấy khoá công khai của người cần liên lạc.

## 3.11 Một số ứng dụng cơ bản (của các hệ thống mật mã khóa công khai nói chung)

* *Bảo mật trong truyền tin (Confidentiality)*

A sẽ gửi *EZB* (*X*) cho B.

*B* dễ dàng giải mã bằng khóa bí mật *zB*

* *Chứng thực*

+ Alice ký lên tin cần gửi bằng cách mã hoá với khoá bí mật của cô ta *DzA* (*X*) và gửi (*X*,*S*)  (*X*, *DzA* (*X* )) cho Bob

+ Khi Bob muốn kiểm tra tính tin cậy của tin nhận được, anh ta chỉ việc tính *X* ' *EZA* (*X* )  *EZA* (*DzA* (*X* )) và kiểm tra nếu *X = X’* thì xác thực được tính tin cậy (authenticity) của X.

Chú ý 1: Trong quá trình này cả việc kiểm tra (i) tính toàn vẹn của thông báo và việc (ii) xác thực danh tính của người gửi được thực hiện cùng một lúc. Ta có (i) là vì chỉ cần một bit của tin mà bị thay đổi thì sẽ lập tức bị phát hiện ngay do chữ ký không khớp. Ngoài ra có (ii) vì không ai có thể tạo ra được thông báo đó ngoài Alice, người duy nhất biết *zA*.

Chú ý 2: Alice có thể ký vào giá trị băm (hash) của *X* thay vì ký thẳng lên *X*. Khi đó toàn bộ mã mà Alice sẽ chuyển cho Bob là (*X*, *DzA* (*H*(*X* ))) . *H* là một hàm băm công khai.

Phương pháp này là hiệu quả hơn do tiết kiệm (hàm băm luôn cho ra một xâu độ dài cố định và thông thường ngắn hơn rất nhiều so với xâu đầu vào).

* *Kết hợp tính mật và tin cậy.*

Chúng ta có thể làm như sau để kết hợp cả hai khả năng a và b như trên. A gửi *Y*  *EZB* (*DzA* (*X* )) cho B

B phục hồi *X* như sau: *X*  *EZA* (*DzB* (*Y*))  *EZA* (*DzB* (*EZB* (*DzA* (*X* ))))

Để có bằng chứng nhằm đối phó với việc Alice có thể sau này phủ nhận đã gửi thông báo (*non-repudiation*) thì Bob phải lưu giữ *DzA* (*X*)

## 3.12 Một số vấn đề xung quanh thuật toán RSA

*Vấn đề chọn p và q:*

+ *p* và *q* phải là những số nguyên tố lớn, ít nhất là cỡ *100* chữ số.

+ *p* và *q* phải lớn cỡ xấp xỉ nhau ( về độ dài cùng *100* chữ số chẳng hạn).

*Một vài con số về tốc độ thuật toán trong cài đặt:*

So sánh với DES thì RSA:

+ Có tốc độ chậm hơn rất nhiều. Thường thì, RSA chậm ít nhất là 100 lần khi cài đặt bằng phần mềm, và có thể chậm hơn từ 1000 đến 10,000 lần khi cài đặt bằng phần cứng (còn tùy cách cài đặt)

+ Kích thước của khoá mật lớn hơn rất nhiều.

Nếu như *p* và *q* cần biểu diễn cỡ 300 bits thì *n* cần 600 bits. Phép nâng lên luỹ thừa là khá chậm so với *n* lớn, đặc biệt là nếu sử dụng phần mềm (chương trình).

Người ta thấy rằng thực hiện một phép nhân cỡ m + 7 nhịp Clock khi kích thước *n* là m bit.

Về bài toán phân tích ra thừa số nguyên tố

Giải thuật tốt nhất vẫn là phương pháp sàng số. Một ước lượng về thời gian thực hiện của giải thuật là:

1

9.7 log2 *n*

*L(n)*  10 50

Trong đó *log2n* cho số biết số bit cần để biểu diễn *n*, số cần phân tích ra thừa số nguyên tố. Từ đó rút ra, nếu tăng *n* lên thêm 50 bit (quãng 15 chữ số thập phân) thì thời gian làm phân tích ra thừa số nguyên tố tăng lên 10 lần.

*Vấn đề đi tìm số nguyên tố lớn:*

Một thuật toán để tạo ra tất cả các số nguyên tố là không tồn tại, tuy nhiên có những thuật toán khá hiệu quả để kiểm tra xem một số cho trước có phải là nguyên tố hay không (bài toán kiểm tra tính nguyên tố). Thực tế, việc tìm các số nguyên tố lớn cho RSA là một vòng lặp như sau:

1. Chọn một số ngẫu nhiên p nằm trong một khoảng có độ lớn yêu cầu (tính theo bit)
2. Kiểm tra tính nguyên tố của p, nếu là nguyên tố thì dừng lại, nếu không thì quay lại bước 1.

Những thuật toán tất định để kiểm tra tính nguyên tố là khá tốn thời gian và đòi hỏi được thực hiện trên máy tính có tốc độ cao. Tuy nhiên người ta cũng còn sử dụng các thuật toán xác suất, có khả năng „đoán‟ rất nhanh xem một số có phải nguyên tố không. Các thuật toán xác suất này không đưa ra quyết định đúng tuyệt đối, nhưng cũng gần như tuyệt đối; tức là xác suất báo sai có thể làm nhỏ tùy ý, chỉ phụ thuộc vào thời gian bỏ ra.

*Giải thuật tính luỹ thừa nhanh*

Luỹ thừa có thể được tính như thông thường bằng phép nhân liên tục tuy nhiên tốc độ sẽ chậm. Luỹ thừa trong trường *Zn* (modulo *n*) có thể tính nhanh hơn nhiều bằng giải thuật sau đây. Giải thuật này sử dụng hai phép tính là tính bình phương và nhân.

## 3.13 Điểm yếu của giải thuật RSA

Trong hệ RSA, không phải tất cả các thông tin đều được che giấu tốt, tức là mọi khoá đều tốt và đều làm bản rõ thay đổi hoàn toàn.

Đối với bất kỳ khoá nào tồn tại ít nhất 9 bản rõ bị “phơi mặt”, tuy nhiên đối với n  200 điều đó không còn quan trọng. Mặc dù vậy phải chú ý là nếu e không được chọn cẩn thận thì có thể gần đến 50% bản rõ bị lộ.

Người ta cho rằng có thể tránh được tình huống này nếu số nguyên tố được chọn là AN TOÀN. Một số nguyên tố được gọi là AN TOÀN nếu *p=2p’+1* trong đó *p’* cũng là số nguyên tố.

## **3.14 Đánh giá về an toàn của thuật toán RSA**

*Sự an toàn của thành phần khoá mật (private key) phụ thuộc vào tính khó của việc PTTSNT các số lớn.*

Ký hiệu *Z= (e,n)* là khoá công khai.

Nếu biết PTTSNT của *n là n=p**q* thì sẽ tính được *m=**(n) =(p-1)(q-1)*. Do đó tính được *d=e-1(mod m)* theo thuật toán *GCD* mở rộng.

Tuy nhiên nếu không biết trước *p,q* thì như đã biết không có một thuật toán hiệu quả nào để PTTSNT đối với *n*, tức là tìm được *p,q*, khi *n* lớn. Nghĩa là không thể tìm được *m* và do đó không tính được *d*.

Chú ý: Độ an toàn của RSA chưa chắc hoàn toàn tương đương với tính khó của bài toán PTTSNT, tức là có thể tồn tại phép tấn công phá vỡ được RSA mà không cần phải biết PTTSNT của *n*, chẳng hạn nếu như có kẻ thành công trong các dạng tấn công sau:

* Đi tìm thành phần khóa mật

Kẻ thù biết X và Y với Y=Dz(X). Để tìm d nó phải giải phương trình:

*X = Yd**n*

Hay là tính *d = logYX*

* Đi tìm bản rõ:

Kẻ thù biết *Y* và *e*, để tìm được bản rõ *X* nó phải tìm cách tính căn thức bậc *e* theo đồng dư, để giải phương trình

*Y=Xe*

## 3.15 Một số dạng tấn công có điều kiện quan trọng

*Đối với một số hệ cài đặt rơi vào một số điều kiện đặc biệt có thể trở nên kém an toàn với người sử dụng.*

* *Common modulus attack*: Khi một nhóm user sử dụng các khoá công khai *Z=(e,n)* khác nhau ở thành phần *e* nhưng giống nhau ở modul đồng dư *n*.

Khi đó, nếu kẻ thù tóm được hai đoạn bản mã mà:

+ của cùng một bản rõ được mã hoá bởi khoá PK khác nhau (từ hai user khác nhau)

+ hai thành phần *e* tương ứng là nguyên tố cùng nhau

thì nó sẽ có cách để giải được bản mã. Cụ thể là nếu kẻ thù biết *e1,e2,n,Y1,Y2*





Vì *(e1,e2)=1* nên nó có thể tìm được a và b sao cho: *a\*e1+b\*e2 = 1*

Suy ra kẻ thù có thể tìm được *X* từ:



Tóm lại nên tránh sử dụng chung modul đồng dư (common modulus) giữa những user cùng một nhóm làm việc nào đó.

* *Low exponent attack*:

Tấn công này xảy ra với điều kiện là giá trị *e* đã được chọn nhỏ (*e* mà nhỏ thì thuật toán mã hoá trong truyền tin mật cũng như kiểm định chữ ký sẽ nhanh hơn).

Nếu kẻ thù có thể tìm được *e(e+1)/2* bản mã mà được mã hoá từ những bản rõ phụ thuộc tuyến tính thì hệ thống sẽ bị nguy hiểm. Tuy nhiên nếu các bản rõ này mà không có quan hệ với nhau thì không sao. Vì vậy nên ghép thêm vào các bản rõ những xâu nhị phân ngẫu nhiên để đảm bảo cho chúng là không bị phụ thuộc.

Nếu thành phần khóa mật *d* mà đủ nhỏ thì có thể bị kẻ thù tìm thấy được

## 3.16 Hệ Rabin

Hệ Rabin cũng xây dựng trên việc lấy *n=p**q* làm bí mật. *N* được coi là khoá công khai (PK) còn *(p,q)* là khoá bí mật (SK).

Mã hoá là việc thực hiện:

*Y=X2* (*mod n*)

còn giải mã là việc tính căn bậc hai:

X= *Y* (*mod* *n*) (\*)

Có thể thấy, nếu biết *n=p**q* thì dễ dàng tìm được nghiệm cho phương trình này, còn nếu không thì việc tìm nghiệm là khó tương đương với bài toán PTTSNT số *n*.

Khi biết *N=p**q* thì (\*) được giải ra có bốn nghiệm, do đó để xác định được đâu là bản rõ gốc phải có mẹo để chọn được đúng giá trị cần thiết trong số 4 nghiệm đó Hệ Rabin có một số ưu điểm so với RSA:

 Tính an toàn được chứng minh hoàn toàn tương đương với bài toán PTTSNT, nói cách khác tính ATBM của Rabin là có thể chứng minh được (provable)  Ngoại trừ trường hợp RSA hoạt động với *e* nhỏ còn thuật toán sinh mã của Rabin nhanh hơn nhiều so với RSA là hệ đòi hỏi phải tính luỹ thừa. Thời gian giải mã thì tương đương nhau

## 3.17 Nhược điểm của hệ Rabin

Vì phương trình giải mã cho 4 nghiệm nên làm khó dễ việc giải mã.

Thông thường, bản rõ trước khi được mã hoá cần được nối thêm vào đuôi một chuỗi số xác định để làm dấu vết nhận dạng (chẳng hạn nối thêm 20 số 0 – như vậy trong số 4 nghiệm giải ra, chuỗi nào tận cùng bằng 20 con 0 thì đúng là bản rõ cần nhận).

Vì lý do này nên Rabin thường được dùng chủ yếu cho chứng thực (chữ ký điện tử).

**3.4.2 Hệ El-Gamal**

*Tạo khoá*

Alice chọn một số nguyên tố *p* và hai số nguyên ngẫu nhiên *g* và *u*, cả hai đều nhỏ hơn *p*. Sau đó tính *y =gu (mod p)*

*B*ây giờ khóa công khai của Alice được lấy là *(p,g,y)*, khoá mật là *u*.

*Sinh mã*

* Nếu Bob muốn mã hoá một tin *X* để truyền cho Alice thì trước hết anh ta chọn một số ngẫu nhiên *k* sao cho *(k,p-1) =1*
* Tính *a=gk (mod p) b=ykX (mod p)*

Mã là *Y=(a,b)* và có độ dài gấp đôi bản rõ.

Giải mã: Alice nhận được Y= (a,b) và giải ra X theo công thức sau: *b*

*X* *au* (*mod p*)

# Chương IV CHỮ KÝ ĐIỆN TỬ VÀ HÀM BĂM

## 4.1 Khái niệm chữ ký điện tử?

Hệ thống mật mã đối xứng đã được sử dụng phổ biến trước đó không có tính chất đại diện duy nhất cho một cá nhân. Trong khi một hệ mã hóa khóa công khai (hay còn gọi là phi đối xứng) có thể được xem là được tạo lập để giúp bảo mật truyền tin trong liên lạc giữa 1 cá nhân và phần còn lại của xã hội. Nhờ có mật mã KCK, khái niệm chữ ký điện tử mới được hiện thực hóa và giúp cho giao dịch kinh tế thương mại trong đời sống có thể đi vào số hóa hoàn toàn, qua đó thúc đẩy hoạt động dịch vụ trực tuyến trên Internet phát triển như ngày này.

## 4.2 Sơ đồ chữ ký cơ bản

Nguyên lý tạo chữ ký điện tử là khác hẳn và phức tạp hơn. Đó là, khi có một văn bản ở dạng nhị phân *X*, người ta phải tạo ra một chữ ký ở dạng nhị phân *S* sao cho *S* phụ thuộc hàm vào *X*, tức là *S*=*f* (*X*);

hơn nữa quan hệ hàm này là bí mật (có tham số khóa bí mật) đối với người ngoài. Do đó nếu có kẻ nào thử đánh tráo (tức giả mạo) chữ ký, quan hệ hàm *S*=*f* (*X*) sẽ không còn đúng và bị phát hiện.

Tuy nhiên việc phát hiện xem một văn bản có chữ ký có là chuẩn hay bị giả mạo lại phải là một thao tác mà ai cũng làm được dễ dàng, không cần đến khóa bí mật kia (do người chủ chữ ký nắm giữ). Vì vậy hệ thống chữ ký điện tử được xây dựng trên nguyên tắc sử dụng hai thuật toán riêng rẽ cho việc tạo chữ ký và kiểm định chữ ký, thông qua việc sử dụng cặp 2 hàm toán học đối lập nhau, một cần khóa bí mật còn một thì không. Chính do điều này, mật mã khóa công khai đã được khai thác để giúp hiện thực điểm chốt của cơ chế đặc biệt này.

## 4.3 Các ứng dụng của chữ ký điện tử ?

*Tính chất không chối cãi được (non-repudiation):*

Như vậy nếu Bob đã nhận được văn bản có chữ ký X||S và dùng khóa công khai của Alice để kiểm định thành công, văn bản đó trở thành bằng chứng, ngay cả khi Alice có muốn chối cãi đã tạo ra và ký nó cũng không được. Bởi vì chỉ duy nhất Alice mới sở hữu khóa *dA* bí mật để tạo ra được chữ ký hợp lệ mà thôi.

Ta gọi tính chất này của chữ ký điện tử là *tính không thể chối cãi được* (non-repudiation). Ngay cả khi Alice có khiếu nại bị oan với lý do chữ ký tạo ra bởi một kẻ đã ăn cắp được khóa bí mật của cô ta, thì điều này cũng không thể chứng minh được (tình trạng “*tình ngay lý gian*”).

*Công chứng*

Để có thể đảm bảo phòng tránh được tình trạng chữ ký giả mạo do kẻ gian ăn cắp được khóa bí mật của người bị hại, người ta đã giới thiệu thêm hệ thống *công chứng – public notary*.

Ý tưởng thực hiện: có thêm một bên thứ ba tham gia, vô tư và có thẩm quyền hợp pháp, được gọi là công chứng viên (*public notary*), sẽ được thuê để ký xác nhận thêm vào sau chữ ký của Alice đối với những văn bản quan trọng mà Alice ký.

Văn bản đầy đủ chữ ký sẽ có dạng Y=X||SA||SN trong đó chữ ký của công chứng viên SN là ký trên văn bản X||SA.

*Bằng chứng biên nhận:*

Trong truyền tin liên lạc, chữ ký điện tử có thể sử dụng để đảm bảo tính chính xác của tài liệu (bằng chữ ký của bên gửi A), và bên nhận B có thể gửi lại chữ ký của mình vào tài liệu đã nhận như là bằng chứng để A biết là B đã thực sự nhận được tài liệu đó.

Nếu thủ tục này được thực hiện, sau này A có thể chứng minh được là mình đã gửi tài liệu cho B, ngay cả khi lúc đó B muốn chối cũng không được.

## 4.4 Nhược điểm của hệ chữ ký cơ sở ?

Hệ chữ ký điện tử theo tiếp cận ban đầu nói trên, tức là sử dụng *Dz* để ký và *EZ* để kiểm định, là khá đơn giản và phạm phải nhược điểm lớn:

* Chữ ký quá dài, dài đúng bằng tài liệu: Với văn bản dài, ta cần dùng việc chia khối rồi ký lên nhiều khối; cụ thể là X = X1|| X2|| X3|| ... ||Xt  S= SA(X1) || SA(X2) || SA(X3) || ... || SA(Xt)). Rõ ràng số lượng khối trên văn bản đã ký nhiều gấp đôi ban đầu.
* Không những dài, việc thực hiện nhiều lần thuật toán KCK (ký lên từng khối) sẽ làm thủ tục ký có thể diễn ra rất lâu, thời gian tỷ lệ với độ dài văn bản. Điều này là không chấp nhận được với các giao dịch trực tuyến
* Kẻ tấn công có thể dễ dàng phá hệ thống chữ ký này bằng kiểu tấn công lắp ghép khối (thay đổi thứ tự, thêm hay bớt khối …). Cách làm chi tiết tương tự như trong tấn công vào chế độ mật mã bảng tra điện tử ECB, đã giới thiệu trong chương 2.

Vì vậy hệ thống chữ ký điện tử đơn giản kiểu này đã không được sử dụng. Giải pháp đầy đủ là có thêm sự hộ trợ của hàm băm, tức là “Băm” tài liệu trước khi ký, sẽ được trình bày tiếp theo đây.

## 4.5 Hàm băm là gì?

Một hàm băm *H* sẽ lấy ở đầu vào một thông tin *X* có kích thước bất kỳ và sinh kết quả ra là một chuỗi *hX*=*h*(*X*) có độ dài cố định, thường là nhỏ hơn nhiều so với kích thước của *X*. Chuỗi này thường được gọi là cốt yếu, hay cốt (digest) của thông tin *X*.

Tuy nhiên với hàm băm thiết kế tốt, đụng độ là gần như không thể xảy ra được trên thực tế. Nói cách khác nếu cố đi tìm, khối lượng tính toán phải thực hiện là rất lớn, không khả thi với công cụ tính toán hiện thời.

Hàm băm có ứng dụng chủ chốt trong các hệ chữ ký điện tử được sử dụng hiên nay. Thay vì ký (tức là thực hiện thuật toán DzA) lên văn bản X, Alice cần thực hiện việc ký lên hX; như vậy văn bản đã ký sẽ có dạng X|| DzA(H(X)).

Để đảm bảm an toàn cao, chống được tấn công giả mạo chữ ký, chúng ta cần sử dụng các hàm băm mật mã (cryptographic hash function) với các thuộc tính như sau:

## 4.6 Các thuộc tính của hàm băm?

* Lấy đầu vào là một xâu với độ dài bất kỳ và sinh ra một xâu với độ dài cố định.
* Có tính một chiều (one - way): biết *X*, có thể dễ dàng tính được giá trị băm *hX*, nhưng không thể tính ngược được *X* khi chỉ biết *hX*, với công cụ tính toán hiện nay (bất khả thi về tính toán)
* Có tính phi đụng độ cao (collision free), tức là thực tế không thể tim được hai thông tin X X‟ sao cho *H*(*X*)  *H*(*X*). Tất nhiên, đây là bất khả thi về mặt tính toán .

## 4.7 Tại sao các thuộc tính của hàm băm cầ thiết cho hệ chữ ký điện tử?

Các thuộc tính trên của hàm băm là cần thiết cho hệ chữ ký điện tử (DS), vì:

* Tính chất 1 cần cho việc sinh chữ ký một cách hiệu quả (chữ ký lên (*hX* rõ ràng sẽ ngắn hơn rất nhiều lên trực tiếp X).
* Tính chất 2 và 3 được dùng để chống lại những kẻ giả mạo chữ ký.

## 4.8 “Đụng độ” là gì?

Với không gian giá trị băm nhỏ hơn không gian tin về mặt kích thước thì chắc chắn sẽ tồn tại đụng độ (collision), nghĩa là có hai bản rõ *X**X’* mà giá trị băm của chúng giống nhau nghĩa là *hX=hX’*.

Điều này có thể thấy rõ ràng qua nguyên lý Diricle: “*Nếu có n+1 con thỏ được thả vào n cái chuồng thì phải tồn tại ít nhất một cái chuồng mà trong đó có ít ra là hai con thỏ ở chung*”

Trong thực tế người ta thường chọn không gian băm cỡ khoảng 64bit, 128 bit... Trong khi đó các văn bản thực tế lớn hơn nhiều, cỡ Kb trở lên, cho nên việc tồn tại đụng độ là chắc chắn. Tuy nhiên nếu sử dụng hàm băm mật mã có không gian băm lớn được chế tạo tốt (an toàn) thì việc tìm ra đụng độ đòi hỏi khối lượng tính toán lớn đến mức phi thực tế (infesible computation).

Việc chế tạo các hàm băm phi đụng độ là rất khó. Nhiều hàm băm được phát minh bởi các nhóm có tên tuổi trên thế giới sau một thời gian xuất hiện đã bị những người khác chỉ ra những đụng độ tồn tại và không được công nhận là an toàn nữa.

## 4.8 Birthday attack

Như ta đã biết, có một dạng tấn công gỉa mạo nguy hiểm đối với các hệ chữ ký điện tử có dùng hàm băm là kẻ tấn công tìm cách tạo ra được những văn bản *X* và *X’* có nội dung khác nhau (một có lợi một có hại cho bên A, người sẽ bị lừa để ký vào) mà có giá trị băm giống nhau. Kẻ thù có thể tìm cách tạo ra một số lượng rất lớn các văn bản có nội dung không thay đổi nhưng khác nhau về biểu diễn nhị phân (đơn giản là việc thêm bớt các dấu trắng, dùng nhiều từ đồng nghĩa thay thế nhau ...) sau đó sử dụng một chương trình máy tính để tính giá trị băm của các văn bản đó và đem so sánh với nhau để hi vọng tìm ra một cặp văn bản có đụng độ.

Như đã nêu ở phần trên thì để chắc chắn có thể tìm được một đụng độ như vậy số văn bản cần được tính giá trị băm phải lớn hơn kích thước không gian băm. Chẳng hạn như nếu hàm băm có không gian băm 64 bit thì số lượng văn bản cần được đem ra 64 nạp vào chương trình thử này phải là ít nhất 2 , một con số quá lớn đến mức hàng thế kỷ nữa cũng không thực hiện xong!

Tuy nhiên nếu như kẻ tấn công đem thử với một lượng văn bản ít hơn nhiều, trong phạm vi có thể tính toán được, thì xác xuất để tìm được đụng độ có đáng kể hay không? Và câu trả lời thực đáng ngạc nhiên, xác xuất này có thể vẫn khá lớn, tức là có nhiều hy vọng tìm được đụng độ dù tập văn bản đem thử không lớn lắm. Bản chất của hiện tượng này có thể được minh hoạ rõ qua một phát biểu, thường gọi là *Nghịch lý Ngày sinh nhật* (Birthday Paradox) như sau: *Trong một nhóm có 23 người bất kỳ, xác xuất để có hai người có cùng một ngày sinh nhật là không ít ½*.

Một cách tổng quát, giả sử một hàm băm có *m* giá trị băm khác nhau (tức là kích thước của không gian output của hàm băm là *m*). Nếu chúng ta có *k* giá trị băm từ *k* thông tin được chọn ngẫu nhiên khác nhau, thì xác xuất để có ít nhất một đụng độ là:

*k*(*k*1)

*P*(*m*,*k*) 1*e* 2*m*  (\*)

Với e là hằng số Ơ - le: e  2.7

Ước lượng xác xuất này phụ thuộc vào kích thước của không gian băm (*m*) và số lượng văn bản thông tin được thử đến chứ không phụ thuộc vào hàm băm đều sử dụng. Tức là kích thước của không gian băm xác lập một chặn dưới (lower bound) cho xác xuất trên.

Tìm đụng độ trên không gian văn bản có kích cỡ 2 là một điều hiện thực đối với ngay cả các máy tính PC bình thường. Việc nghiên cứu Birthday Paradox cho ta

32 thấy lượng văn bản cần đưa ra thử có thể là rất nhỏ so với không gian băm (2 so với 264) mà xác suất tìm được đụng độ là khá cao (50%). Điều này cho thấy một mối hiểm hoạ cho các hệ dùng hàm băm có không gian output nhỏ. Phép tấn công này được gọi là *Tấn công ngày sinh nhật* (Birthday attack).

## 4.9 Các hàm băm dựa trên các phép toán số học đồng dư ?

* *QCMDC (Quadratic Congruential Manipulation Detection Code)*

Bản rõ được chia thành các khối *m* bit. H0 là giá trị khởi đầu được chọn ngẫu nhiên và giữ bí mật ( vì thế vẫn được gọi là hàm băm có khóa - keyed hash function).

Các bước xây dựng hàm băm như sau:

M là một số nguyên tố sao cho M  2m-1,

Hi = (Hi-1 + Xi)2 (mod M)

Hn sẽ là giá trị băm Hệ này đã bị phá (Coppersmith).

* *Davies-Price (1985)*

Chia văn bản thành các khối có *m-d* bit:

*X = X1 X2 X3 ... Xn*

*Hi = (Hi-1*  *Xi)2 (mod M), H0=0*

*M* là luỹ thừa của 2.

Hệ này bị chứng minh là không đảm bảo tính một chiều (Girault)

## 4.10 Các hàm băm được chế tạo đặc biệt

* *MD5 (Rivest 1992)*

Đây là một trong các hàm băm có tiếng nhất và được sử dụng thông dụng:

+ Nó lấy vào các khối đầu vào 512 bit và sinh ra các giá trị băm 128 bit.

+ Được tin là phi đụng độ và one-way

+ Thuật toán MD5 được thiết kế cho phép chạy tốt nhất trên các máy tính 32 bit. Nó sử dụng các phép toán đơn giản như phép cộng modulo 32, do đó thích hợp với ciệc mã hoá cho các bộ xử lý 32 bit.

* *SHA (Secure Hash Function)*

Đây là một thuật toán được đề xuất và bảo trợ bởi cơ quan NIST để sử dụng đối với hệ chữ ký DSA (cũng là một dự chuẩn cho chữ ký điện tử). Nó cho giá trị băm là 160 bit và được thiết kế với cùng một tiếp cận như MD5.

*HAVAL*

Một hệ băm của Australia cho phép thay đổi kích thước giá trị băm. Cấu trúc rất giống như MD5.

* *Snefru Mekle (1989)*

+ Là hàm băm có khóa (keyed hash function)

+ Cho phép 1 trong 2 lựa chọn kích thước giá trị băm là 128 bit và 256 bit

+ Eli Biham đã chỉ ra một đụng độ cho trường hợp 128 bit

## 4.11 “El-Gamal” là gì?

Được xây dựng trên *tính khó* của việc tính toán logarit trên không gian *Zp* khi mà *p* là số nguyên tố.

*Thuật toán*

+ Alice chọn một số nguyên tố lớn *p* sao cho *p-1* có một ước số nguyên tố lớn + Giả sử *a* là phần tử cấu trúc (primitive element) của *Zp: {ai, i=0,p-1}*  *Zp\*.*

+ Alice chọn *x* và tính *y=ax* (mod *p*).

+ Khoá công khai là *p,a,y*. Khoá bí mật là *x*.

Để ký một bản rõ *X*:

+ Alice chọn một số nguyên *k, 1*  *k*  *p-1* sao cho *gcd(k,p-1) =1*

+ Cô ta tính *r=ak* (mod *p*).

+ Tính *s=k-1(X-xr)* (mod *p-1*)

+ Tính *s=k-1(X-xr)* (mod *p-1*)

+ TIN với chữ ký là: (*X*||(*r,s*))

Để kiểm định chữ ký, làm như sau:

*a X*  *yr* *r s*

*DSA*

Được đề xướng bởi NIST, công bố năm 1994 Những phê bình chống lại DSA:

+ không dùng được cho mã hoá dữ liệu và phân phối khoá

+ được phát triển bởi NSA (cơ quan an ninh Hoa kỳ) và do đó không đáng tin

+ Kích thước khoá quá nhỏ

## 4.12 Chữ ký mù (Blind signature)

Là một dạng của chữ ký điện tử , trong đó nội dung của một thông điệp được ngụy trang (mù) trước khi nó được ký kết. Chữ ký mù kết quả có thể được xác minh công khai dựa trên thông điệp ban đầu, không bị ràng buộc theo cách của một chữ ký điện tử thông thường. Chữ ký mù thường được sử dụng trong các giao thức liên quan đến quyền riêng tư trong đó người ký và tác giả thư là các bên khác nhau. Ví dụ bao gồm các hệ thống bầu cử mật mã và các chương trình tiền kỹ thuật số .

## 4.13 Group signature

Chữ ký này chỉ nói lên người ký nằm trong nhóm đó thôi chứ không nói lên đích xác đó là người nào nên giữ được tính bí mật riêng tư của người ký.

Tuy nhiên hệ chữ ký đặc biệt này còn có một tính chất đặc biệt nữa là:

Nếu như cần thiết, một người thẩm quyển có thể “mở” được một chữ ký ra để xem ai cụ thể trong nhóm đã ký.

Ứng dụng của nó là nếu như chương trình quản máy in cho thấy có người đã quá lạm dụng thì trưởng phòng có thể sử dụng quyền hạn của mình để “phanh phui” những chữ ký lên các yêu cầu in tốn kém đó, sau đó có biện pháp phạt người lạm dụng như nộp tiền phạt.

Khả năng này làm cho tất cả mọi người phải biết điều đối với máy in của công nếu không muốn bị “bêu tên”.

Hệ chữ ký trên do đó được gọi là hệ chữ ký nhóm (group signature)

## 4.14 Undeniable signature

Đây là chữ ký mà thuật toán kiểm định đòi hỏi phải có sự tham gia của người ký. Thực chất đây là chữ ký có tính chất không thể chuyển giao được (untransferable):

Chỉ có ý nghĩa đối với người nhận là người có trao đổi làm ăn với người ký, khi chuyển nó cho một người khác thì không có tác dụng nữa (không thể kiểm định được chữ ký nữa).

Các văn bản có chữ ký này không nhằm vào mục đích đem đi công bố ở nơi khác mà chỉ có tính chất giấy phép. Vì thế nếu sao chép là mất ý nghĩa.

Chữ ký này được dùng trong việc bán các sản phẩm phần mềm:

các hãng phần mềm sẽ bán các sản phẩm của mình có chữ ký chứng tỏ tính bản quyền.

Việc kiểm định đòi hỏi phải liên lạc với hãng này.

Nếu như có việc một con buôn nào đó bán phần mềm sao chép thì lúc người mua đòi kiểm định sẽ bị lộ ngay vì không thực hiện được.

## 4.15 Multisignature (Đồng ký)

Ở đây, chữ ký không phải là của một người mà của một nhóm người.

Muốn tạo được chữ ký, tất cả những người này cùng phải tham gia vào protocol.

Tuy nhiên chữ ký có thể được kiểm định bởi bất kỳ ai.

Đây là trường hợp dành cho thực tế của việc đưa ra những quyết định do nhiều người.

## 4.16 Proxy signature (chữ ký uỷ nhiệm)

Hệ chữ ký này dành cho các trường hợp mà người chủ chữ ký bị ốm không có khả năng làm việc hay là đi vắng đến một nơi không có phương tiện mạng máy tính cần thiết để ký. Vì vậy chữ ký uỷ nhiệm được tạo ra để người chủ có thể uỷ nhiệm cho một người nào đó ký thay. Tất nhiên chữ ký uỷ nhiệm phải có các thuộc tính riêng thêm vào:

+ Chữ ký uỷ nhiệm là phân biệt với chữ ký thường, và người được uỷ nhiệm không thể tạo được chữ ký chủ (chữ ký thường của người chủ).

+ Chữ ký uỷ nhiệm cũng có chức năng chứng thực như chữ ký chủ, chỉ có người chủ và người được uỷ nhiệm mới có thể tạo ra được chữ ký này. Người nhận được văn bản có thể hoàn toàn tin tưởng vào chữ ký đó như chữ ký chủ.

+ Người chủ có thể xác định được danh tính người ký từ một chữ ký uỷ nhiệm

+ Người được uỷ nhiệm ký không thể chối cãi được nếu đã ký một văn bản uỷ nhiệm hợp lệ (Tức là anh ta không thể chối đổ cho ai khác hay chính người chủ đã ký mà lại nói là anh ta ký).