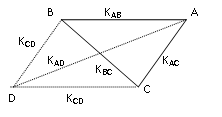
# Chương III

## 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).