HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

GIÁO TRÌNH

CƠ SỞ MẬT MÃ HỌC

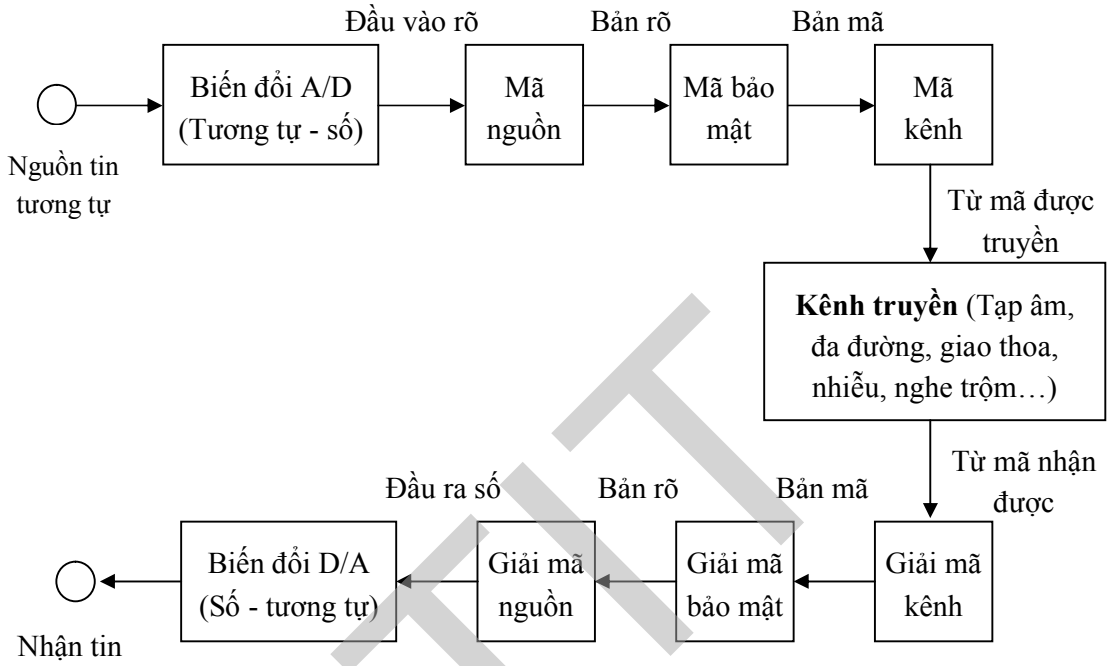
Chủ biên: GS.TS Nguyễn Bình
Cộng tác viên: TS. Ngô Đức Thiện
Khoa KTĐT1 - Học viện CNBCVT

Hà Nội - 2013

MỤC LỤC

CHƯƠNG 1. NHẬP MÔN MẬT MÃ HỌC

1.1. SƠ ĐỒ KHỐI ĐƠN GIẢN CỦA MỘT HỆ THỐNG THÔNG TIN SỐ



Hình 1.1. Sơ đồ hệ thống thông tin số

Trường hợp nguồn tin đầu vào là nguồn tin số thì không cần bộ biến đổi A/D ở đầu vào và bộ biến đổi D/A ở đầu ra

Trong hệ thống này khối mã bảo mật có chức năng bảo vệ cho thông tin không bị khai thác bất hợp pháp, chống lại các tấn công sau:

- Thám mã thụ động: bao gồm các hoạt động:
  + Thu chặn.
  + Dò tìm.
  + So sánh tương quan.
  + Suy diễn.
- Thám mã tích cực: bao gồm các hoạt động:
  + Giả mạo.
  + Ngụy trang.
  + Sử dụng lại.
  + Sửa đổi.

1.2. SƠ LƯỢC VỀ MẬT MÃ HỌC

Khoa học về mật mã (cryptology) bao gồm:

-Mật mã học (cryptography).

-Phân tích mật mã (cryptanalysis).

Mật mã học là khoa học nghiên cứu cách ghi bí mật thông tin nhằm biến bản tin rõ thành các bản mã.

Phân tích mã là khoa học nghiên cứu cách phá các hệ mật nhằm phục hồi bản rõ ban đầu từ bản mã. Việc tìm hiểu các thông tin về khóa và các phương pháp biến đổi thông tin cũng là một nhiệm vụ quan trọng của phân tích mật mã.

Có ba phương pháp tấn công cơ bản của thám mã:

- Tìm khóa vét cạn.
- Phân tích thống kê.
- Phân tích toán học.

Việc tấn công của thám mã có thể được thực hiện với các giả định:

- Tấn công chỉ với bản mã.
- Tấn công với bản rõ đã biết.
- Tấn công với các bản rõ được chọn.
- Tấn công với các bản mã được chọn.

Chú ý:

- Một hệ mật có thể bị phá chỉ với bản mã thường là hệ mật có độ an toàn thấp.
- Một hệ mật là an toàn với kiểu tấn công có các bản rõ được chọn thường là một hệ mật có độ an toàn cao.

Có 4 loại hệ mật mã sau:

- Hệ mật mã dòng.
- Hệ mật mã khối đối xứng.
- Hệ mật mã có hồi tiếp mật mã.
- Hệ mật mã khóa công khai (Bất đối xứng).

Ta sẽ nghiên cứu các loại hệ mật trên ở các chương sau.

Khi xây dựng một hệ mật người ta thường xem xét tới các tiêu chuẩn sau:

- Độ mật cần thiết.
- Kích thước không gian khóa.
- Tính đơn giản và tốc độ mã hóa và giải mã.
- Tính lan truyền sai.
- Tính mở rộng bản tin.

1.3. THUẬT TOÁN VÀ ĐỘ PHỨC TẠP

1.3.1. Khái niệm về thuật toán

1.3.1.1. Định nghĩa thuật toán

Có thể định nghĩa thuật toán theo nhiều cách khác nhau. Ở đây ta không có ý định trình bày chặt chẽ về thuật toán mà sẽ hiểu khái niệm thuật toán theo một cách thông thường nhất.

Định nghĩa 1.1:

Thuật toán là một quy tắc để với những dữ liệu ban đầu đã cho, tìm được lời giải của bài toán được xét sau một khoảng thời gian hữu hạn.

Để minh họa cách ghi một thuật toán cũng như tìm hiểu các yêu cầu đề ra cho thuật toán, ta xét trên các ví dụ cụ thể sau đây:

Cho n số $X[1],X[2],...,X[n]$ ta cần tìm m và j sao cho:

$$m = X[j] = \max_{1 \le k \le n} X[k]$$

Và j là lớn nhất có thể. Điều đó có nghĩa là cần tìm cực đại của các số đã cho và chỉ số lớn nhất trong các số cực đại.

Với mục tiêu tìm số cực đại với chỉ số lớn nhất, ta xuất phát từ giá trị $X[n]$. Bước thứ nhất, vì mới chỉ có một số ta có thể tạm thời xem $m = X[n]$ và $j = n$. Tiếp theo ta so sánh $X[n]$ với $X[n-1]$. Nếu $X[n]$ không nhỏ hơn $X[n-1]$ thì ta giữ nguyên, trong trường hợp ngược lại, $X[n-1]$ chính là số cực đại trong hai số đã xét và ta phải thay đổi m và j. Đặt $m = X[n-1]$, $j = 1,...,n-1$. Với cách làm như trên, ở mỗi bước ta luôn nhận được số cực đại trong số những số đã xét. Bước tiếp theo là so sánh nó với những số đứng trước hoặc kết thúc thuật toán trong trường hợp không còn số nào đứng trước nó.

1.3.1.2. Thuật toán tìm cực đại

M1: [Bước xuất phát] đặt $j \leftarrow n$, $k \leftarrow n-1$, $m \leftarrow X[n]$

M2: [Đã kiểm tra xong?]. Nếu $k = 0$, thuật toán kết thúc.

M3: [So sánh]. Nếu $X[k] \le m$, chuyển sang M5

M4: [Thay đổi m]. Đặt $j \leftarrow k$, $m \leftarrow X[k]$ (Tạm thời m đang là cực đại).

M5: [Giảm k]. Đặt $k \leftarrow k-1$ quay về M2.

Dấu "$\leftarrow$" dùng để chỉ một phép toán quan trọng là phép thay chỗ (replacement).

Trên đây ta ghi thuật toán bằng ngôn ngữ thông thường. Trong trường hợp thuật toán được viết bằng ngôn ngữ của máy tính, ta có một chương trình.

Trong thuật toán có những số liệu ban đầu được cho trước khi thuật toán bắt đầu làm việc được gọi là các đầu vào (input). Trong thuật toán trên đầu vào là các số $X[1],X[2],...,X[n]$.

Một thuật toán có thể có một hoặc nhiều đầu ra (output). Trong thuật toán trên các đầu ra là m và j.

Có thể thấy rằng thuật toán vừa mô tả thỏa mãn các yêu cầu của một thuật toán nói chung, đó là:

- *Tính hữu hạn:* Thuật toán cần phải kết thúc sau một số hữu hạn bước. Khi thuật toán ngừng làm việc ta phải thu được câu trả lời cho vấn đề đặt ra. Thuật toán m rõ ràng thỏa mãn điều kiện này, vì ở mỗi bước ta luôn chỉ từ việc xem xét một số sang số đứng trước nó và số các số là hữu hạn.
- *Tính xác định:* Ở mỗi bước thuật toán cần phải xác định, nghĩa là chỉ rõ việc cần làm. Nếu đối với người đọc thuật toán trên chưa thỏa mãn được điều kiện này thì đó là lỗi của người viết.

Ngoài những yếu tố kể trên, ta còn phải xét đến tính hiệu quả của thuật toán. Có rất nhiều thuật toán về mặt lý thuyết là hữu hạn bước, tuy nhiên thời gian "hữu hạn" đó vượt quá khả năng làm việc của chúng ta. Những thuật toán đó sẽ không được xét đến ở đây, vì chúng ta chỉ quan tâm những thuật toán có thể sử dụng thực sự trên máy tính.

Cũng do mục tiêu trên, ta còn phải chú ý đến độ phức tạp của các thuật toán. Độ phức tạp của một thuật toán có thể được đo bằng không gian tức là dung lượng bộ nhớ của máy tính cần thiết để thực hiện thuật toán và bằng thời gian, tức là thời gian máy tính làm việc. Ở đây khi nói đến độ phức tạp của thuật toán ta luôn hiểu là độ phức tạp của thời gian.

1.3.2. Độ phức tạp của thuật toán

Thời gian làm việc của máy tính khi chạy một thuật toán nào đó không chỉ phụ thuộc vào thuật toán mà còn phụ thuộc vào máy tính được sử dụng. Vì thế, để có một tiêu chuẩn chung, ta sẽ đo độ phức tạp của một thuật toán bằng số các phép tính phải làm khi thực hiện thuật toán. Khi tiến hành cùng một thuật toán, số các phép tính phải thực hiện còn phụ thuộc vào cỡ của bài toán, tức là độ lớn của đầu vào. Vì thế độ phức tạp của thuật toán sẽ là một hàm số của độ lớn đầu vào. Trong những ứng dụng thực tiễn, chúng ta không cần biết chính xác hàm này mà chỉ cần biết "cỡ" của chúng, tức là cần có một ước lượng đủ tốt của chúng.

Trong khi làm việc, máy tính thường ghi các chữ số bằng bóng đèn "sáng, tắt", bóng đèn sáng chỉ số 1, bóng đèn tắt chỉ số 0. Vì thế để thuận tiện nhất là dùng hệ đếm cơ số 2, trong đó để biểu diễn một số, ta chỉ cần dùng hai ký hiệu 0 và 1. Một ký hiệu 0 hoặc 1 được gọi là 1bit "viết tắt của binary digit". Một số nguyên n biểu diễn bởi k chữ số 1 và 0 được gọi là một số $k-bit$.

Độ phức tạp của một thuật toán được đo bằng số các phép tính bit. Phép tính bit là một phép tính logic hay số học thực hiện trên các số một bit 0 và 1.

Để ước lượng độ phức tạp của thuật toán ta dùng khái niệm bậc O lớn.

Định nghĩa 1.2:

Giả sử $f[n]$ và $g[n]$ là hai hàm xác định trên tập hợp các số nguyên dương. Ta nói $f[n]$ có bậc O-lớn của $g[n]$ và viết $f[n]=O(g[n])$, nếu tồn tại một số $C>0$ sao cho với n đủ lớn. Các hàm $f[n]$ và $g[n]$ đều dương thì $f[n]<C\,g[n]$.

Ví dụ 1.1. :

1) Giả sử $f[n]$ là đa thức: $f[n]=a_d n^d+a_{d-1}n^{d-1}+...+a_1 n+a_0$ trong đó $a_d>0$. Dễ chứng minh $f[n]=O(n^d)$.

2) Nếu $f[n]=O(g[n])$, $f_2[n]=O(g[n])$ thì $f_1+f_2=O(g)$.

3) Nếu $f_1=O(g_1)$, $f_2=O(g_2)$ thì $f_1 f_2=O(g_1 g_2)$.

4) Nếu tồn tại giới hạn hữu hạn:

$$\lim_{n\to\infty}\frac{f[n]}{g[n]}$$

thì $f=O(g)$

5) Với mọi số $\varepsilon>0$, $\log n=O(n^{\varepsilon})$

Định nghĩa 1.3:

Một thuật toán được gọi là có độ phức tạp đa thức hoặc có thời gian đa thức, nếu số các phép tính cần thiết để thực hiện thuật toán không vượt quá $O(\log^d n)$, trong đó n là độ lớn của đầu vào và d là số nguyên dương nào đó.

Nói cách khác nếu đầu vào là các số k bit thì thời gian thực hiện thuật toán là $O(k^d)$, tức là tương đương với một đa thức của k.

Các thuật toán với thời gian $O(n^{\alpha})$, $\alpha>0$ được gọi là thuật toán với độ phức tạp mũ hoặc thời gian mũ.

Chú ý rằng nếu một thuật toán nào đó có độ phức tạp $O(g)$ thì cũng có thể nói nó có độ phức tạp $O(h)$ với mọi hàm $h>g$. Tuy nhiên ta luôn luôn cố gắng tìm ước lượng tốt nhất có thể để tránh hiểu sai về độ phức tạp thực sự của thuật toán.

Cũng có những thuật toán có độ phức tạp trung gian giữa đa thức và mũ. Ta thường gọi đó là thuật toán dưới mũ. Chẳng hạn thuật toán nhanh nhất được biết hiện nay để phân tích một số nguyên n ra thừa số là thuật toán có độ phức tạp:

$$\exp\left(\sqrt{\log n\,\log\log n}\right)$$

Khi giải một bài toán không những ta chỉ cố gắng tìm ra một thuật toán nào đó, mà còn muốn tìm ra thuật toán "tốt nhất". Đánh giá độ phức tạp là một trong những cách để phân tích, so sánh và tìm ra thuật toán tối ưu. Tuy nhiên độ phức tạp không phải là tiêu chuẩn duy nhất để đánh giá thuật toán. Có những thuật toán về lý thuyết thì có độ phức tạp cao hơn một thuật toán khác, nhưng khi sử dụng lại có kết quả (gần đúng) nhanh hơn nhiều. Điều này còn tùy thuộc những bài toán cụ thể, những mục tiêu cụ thể và cả kinh nghiệm của người sử dụng.

Chúng ta cần lưu ý thêm một số điểm sau đây. Mặc dù định nghĩa thuật toán mà chúng ta đưa ra chưa phải là chặt chẽ, nó vẫn quá "cứng nhắc" trong những ứng dụng thực tế. Bởi vậy chúng ta còn cần đến các thuật toán "xác suất", tức là các thuật toán phụ thuộc vào một hay nhiều tham số ngẫu nhiên. Những "thuật toán" này về nguyên tắc không được gọi là thuật toán vì chúng có thể với xác suất rất bé, không bao giờ kết thúc. Tuy nhiên thực nghiệm chỉ ra rằng, các thuật toán xác suất thường hữu hiệu hơn các thuật toán không xác suất. Thậm chí trong rất nhiều trường hợp, chỉ có các thuật toán như thế là sử dụng được.

Khi làm việc với các thuật toán xác suất, ta thường hay phải sử dụng các số "ngẫu nhiên". Khái niệm chọn số ngẫu nhiên cũng cần được chính xác hóa. Thường thì người ta sử dụng một "máy" sản xuất số giả ngẫu nhiên nào đó. Tuy nhiên ở đây khi nói đến việc chọn số ngẫu nhiên ta hiểu đó là được thực hiện trên máy.

Cần chú ý ngay rằng, đối với các thuật toán xác suất, không thể nói đến thời gian tuyệt đối, mà chỉ có thể nói đến thời gian hy vọng (expected).

Để hình dung được phần nào "độ phức tạp" của các thuật toán khi làm việc với những số lớn, ta xem Bảng 1.1 dưới đây cho khoảng thời gian cần thiết để phân tích một số nguyên n ra thừa số nguyên tố bằng thuật toán nhanh nhất được biết hiện nay.

Bảng 1.1. Độ phức tạp để phân tích số nguyên ra thừa số nguyên tố

Số chữ số thập phân	**Số phép tính bit**	**Thời gian**
50	$1,4.10^{10}$	3,9 giờ
75	9.10^{12}	104 ngày
100	$2,3.10^{15}$	74 năm
200	$1,2.10^{23}$	$3,8.10^{9}$ năm
300	$1,5.10^{29}$	$4,9.10^{15}$ năm
500	$1,3.10^{39}$	$4,2.10^{25}$ năm

Từ Bảng 1.1 trên, ta thấy rằng ngay với một thuật toán dưới mũ, thời gian làm việc với các số nguyên lớn là quá lâu. Vì thế nói chung người ta luôn cố gắng tìm những thuật toán đa thức.

1.4. LÝ THUYẾT THÔNG TIN TRONG CÁC HỆ MẬT

Năm 1949, Claude Shannon đã công bố một bài báo có nhan đề "Lý thuyết thông tin trong các hệ mật" trên tạp chí "The Bell System Technical Journal". Bài báo đã có ảnh hưởng lớn đến việc nghiên cứu khoa học mật mã. Trong chương này ta sẽ thảo luận một vài ý tưởng trong lý thuyết của Shannon.

1.4.1. Độ mật hoàn thiện.

Có hai quan điểm cơ bản về độ an toàn của một hệ mật.

1.4.1.1. Độ an toàn tính toán

Độ đo này liên quan đến những nỗ lực tính toán cần thiết để phá một hệ mật. Một hệ mật là an toàn về mặt tính toán nếu một thuật toán tốt nhất để phá nó cần ít nhất N phép toán, N là số rất lớn nào đó. Vấn đề là ở chỗ, không có một hệ mật thực tế đã biết nào có thể được chứng tỏ là an toàn theo định nghĩa này. Trên thực tế, người ta gọi một hệ mật là "an toàn về mặt tính toán" nếu có một phương pháp tốt nhất phá hệ này nhưng yêu cầu thời gian lớn đến mức không chấp nhận được. (Điều này tất nhiên là rất khác với việc chứng minh về độ an toàn).

Một quan điểm chứng minh về độ an toàn tính toán là quy độ an toàn của một hệ mật về một bài toán đã được nghiên cứu kỹ và bài toán này được coi là khó. Ví dụ, ta có thể chứng minh một khẳng định có dạng. "Một hệ mật đã cho là an toàn nếu không thể phân tích ra thừa số một số nguyên n cho trước". Các hệ mật loại này đôi khi gọi là "An toàn chứng minh được". Tuy nhiên cần phải hiểu rằng, quan điểm này chỉ cung cấp một chứng minh về độ an toàn có liên quan để một bài toán khác chứ không phải là một chứng minh hoàn chỉnh về độ an toàn. (Tình hình này cũng tương tự như việc chứng minh một bài toán là NP đầy đủ: Có thể chứng tỏ bài toán đã cho chí ít cũng khó như một bài toán NP đầy đủ khác, song không phải là một chứng minh hoàn chỉnh về độ khó tính toán của bài toán).

1.4.1.2. Độ an toàn không điều kiện

Độ đo này liên quan đến độ an toàn của các hệ mật khi không có một hạn chế nào được đặt ra về khối lượng tính toán mà Oscar được phép thực hiện. Một hệ mật được gọi là an toàn không điều kiện nếu nó không thể bị phá thậm chí với khả năng tính toán không hạn chế.

Khi thảo luận về độ an toàn của một hệ mật, ta cũng phải chỉ ra kiểu tấn công đang được xem xét. Trong chương ta thấy rằng, không một hệ mật nào trong các hệ mã dịch vòng, mã thay thế và mã Vigenère được coi là an toàn về mặt tính toán với phương pháp tấn công chỉ với bản mã (Với khối lượng bản mã thích hợp).

Điều mà ta sẽ làm trong phần này là để phát triển lý thuyết về các hệ mật có độ an toàn không điều kiện với phương pháp tấn công chỉ với bản mã. Có thể thấy rằng, cả ba hệ mật nêu trên đều là các hệ mật an toàn vô điều kiện chỉ khi mỗi phần tử của bản rõ được mã hoá bằng một khoá cho trước.

Rõ ràng là độ an toàn không điều kiện của một hệ mật không thể được nghiên cứu theo quan điểm độ phức tạp tính toán vì thời gian tính toán cho phép không hạn chế. Ở đây lý

thuyết xác suất là nền tảng thích hợp để nghiên cứu về độ an toàn không điều kiện. Tuy nhiên ta chỉ cần một số kiến thức sơ đẳng trong xác suất; các định nghĩa chính sẽ được nêu dưới đây.

Định nghĩa 1.4:

Giả sử X và Y là các biến ngẫu nhiên. Kí hiệu xác suất để X nhận giá trị x là p(x) và để Y nhận giá trị y là $p(y)$. *Xác suất đồng thời* $p(x,y)$ *là xác suất để X nhận giá trị x và Y nhận giá trị y. Xác suất có điều kiện* $p(x|y)$ *là xác suất để X nhận giá trị x với điều kiện Y nhận giá trị y. Các biến ngẫu nhiên X và Y được gọi là độc lập nếu* $p(x,y) = p(x)p(y)$ *với mọi giá trị có thể x của X và y của Y.*

Quan hệ giữa xác suất đồng thời và xác suất có điều kiện được biểu thị theo công thức:

$$p(x,y) = p(x|y)\,p(y) \tag{1.1}$$

Đổi chỗ x và y ta có:

$$p(x,y) = p(y|x)\,p(x) \tag{1.2}$$

Từ hai biểu thức trên ta có thể rút ra kết quả sau:(được gọi là định lý Bayes)

Định lý 1.1: (Định lý Bayes)

Nếu $p(y) > 0$ thì:

$$p(x|y) = \frac{p(x)p(y|x)}{p(y)} \tag{1.3}$$

Hệ quả 1.1.

x và y là các biến độc lập khi và chỉ khi: $p(x|y) = p(x)$, $\forall x,y$.

Trong phần này ta giả sử rằng, một khoá cụ thể chỉ dùng cho một bản mã. Giả sử có một phân bố xác suất trên không gian bản rõ $\mathcal{P}$. Kí hiệu xác suất tiên nghiệm để bản rõ xuất hiện là $p_{\mathcal{P}}(x)$. Cũng giả sử rằng, khóa *K* được chọn (bởi Alice và Bob) theo một phân bố xác suất xác định nào đó. (Thông thường khoá được chọn ngẫu nhiên, bởi vậy tất cả các khoá sẽ đồng khả năng, tuy nhiên đây không phải là điều bắt buộc). Kí hiệu xác suất để khóa *K* được chọn là $p_{\mathcal{K}}(K)$. Cần nhớ rằng khóa được chọn trước khi Alice biết bản rõ. Bởi vậy có thể giả định rằng khoá *K* và bản rõ x là các sự kiện độc lập.

Hai phân bố xác suất trên $\mathcal{P}$ và $\mathcal{K}$ sẽ tạo ra một phân bố xác suất trên $\mathcal{C}$. Thật vậy, có thể dễ dàng tính được xác suất $p_{\mathcal{C}}(y)$ với y là bản mã được gửi đi. Với một khoá $K \in \mathcal{K}$, ta xác định:

$$C(K) = \{e_K(x) : x \in \mathcal{P}\}$$

Ở đây $C(K)$ biểu thị tập các bản mã có thể nếu *K* là khóa. Khi đó với mỗi $y \in C$, ta có:

$$p_C(y) = \sum_{\{K : y \in C(K)\}} p_{\mathcal{K}}(K) p_{\mathcal{P}}(d_K(y)) \tag{1.4}$$

Nhận thấy rằng, với bất kì $y \in C$ và $x \in \mathcal{P}$, có thể tính được xác suất có điều kiện $p_C(y|x)$. (Tức là xác suất để y là bản mã với điều kiện bản rõ là x):

$$p_C(y|x) = \sum_{\{K : x = d_K(y)\}} p_{\mathcal{K}}(K) \tag{1.5}$$

Bây giờ ta có thể tính được xác suất có điều kiện $p_{\mathcal{P}}(x|y)$ (tức xác suất để x là bản rõ với điều kiện y là bản mã) bằng cách dùng định lý Bayes. Ta thu được công thức sau:

$$p_{\mathcal{P}}(y|x) = \frac{p_{\mathcal{P}}(x) = \sum_{\{K : x = d_K(y)\}} p_{\mathcal{K}}(K)}{\sum_{\{K : y \in c(K)\}} p_{\mathcal{K}}(K) p_{\mathcal{P}}(d_K(y))} \tag{1.6}$$

Các phép tính này có thể thực hiện được nếu biết được các phân bố xác suất.

Sau đây sẽ trình bày một ví dụ đơn giản để minh hoạ việc tính toán các phân bố xác suất này.

Ví dụ 1.2.

Giả sử $\mathcal{P} = \{a, b\}$ với $p_{\mathcal{P}}(a) = 1/4$, $p_{\mathcal{P}}(b) = 3/4$.

Cho $K = \{K_1, K_2, K_3\}$ với $p_{\mathcal{K}}(K_1) = 1/2$, $p_{\mathcal{K}}(K_2) = p_{\mathcal{K}}(K_3) = 1/4$.

Giả sử $C = \{1, 2, 3, 4\}$ và các hàm mã được xác định là $e_{K_1}(a) = 1, e_{K_1}(b) = 2$, $e_{K_2}(a) = 2, e_{K_2}(b) = 3, e_{K_3}(a) = 3, e_{K_3}(b) = 4$. Hệ mật này được biểu thị bằng ma trận mã hoá sau:

	a	b
K_1	1	2
K_2	2	3
K_3	2	4

Tính phân bố xác suất p_C ta có:

$$p_C(1) = 1/8$$
$$p_C(2) = 3/8 + 1/16 = 7/16$$
$$p_C(3) = 3/16 + 1/16 = 1/4$$
$$p_C(4) = 3/16$$

Bây giờ ta đã có thể các phân bố xác suất có điều kiện trên bản rõ với điều kiện đã biết bản mã. Ta có:

$$p_{\mathcal{P}}(a|1)=1 \quad p_{\mathcal{P}}(b|1)=0 \quad p_{\mathcal{P}}(a|2)=1/7 \quad p_{\mathcal{P}}(b|2)=6/7$$
$$p_{\mathcal{P}}(a|3)=1/4 \quad p_{\mathcal{P}}(b|3)=3/4 \quad p_{\mathcal{P}}(a|4)=0 \quad p_{\mathcal{P}}(b|4)=1$$

Bây giờ ta đã có đủ điều kiện để xác định khái niệm về độ mật hoàn thiện. Một cách không hình thức, độ mật hoàn thiện có nghĩa là Oscar với bản mã trong tay không thể thu được thông tin gì về bản rõ. Ý tưởng này sẽ được làm chính xác bằng cách phát biểu nó theo các thuật ngữ của các phân bố xác suất định nghĩa ở trên như sau:

Định nghĩa 1.5:

Một hệ mật có độ mật hoàn thiện nếu $p_{\mathcal{P}}(x|y)=p_{\mathcal{P}}(x)$ *với mọi* $x\in\mathcal{P}, y\in C$. *Tức xác suất hậu nghiệm để bản rõ là* x *với điều kiện đã thu được bản mã* y *là đồng nhất với xác suất tiên nghiệm để bản rõ là* x.

Trong ví dụ trên chỉ có bản mã 3 mới thoả mãn tính chất độ mật hoàn thiện, các bản mã khác không có tính chất này.

Sau đây sẽ chứng tỏ rằng, mã dịch vòng (MDV - xem chương 2) có độ mật hoàn thiện. Về mặt trực giác, điều này dường như quá hiển nhiên. Với mã dịch vòng, nếu đã biết một phần tử bất kỳ của bản mã $y\in Z_{26}$, thì một phần tử bất kỳ của bản rõ $x\in Z_{26}$ cũng có thể là bản mã đã giải của y tuỳ thuộc vào giá trị của khoá. Định lý sau cho một khẳng định hình thức hoá và được chứng minh theo các phân bố xác suất.

Định lý 1.2:

Giả sử 26 khoá trong MDV có xác suất như nhau và bằng1/26. Khi đó MDV sẽ có độ mật hoàn thiện với mọi phân bố xác suất của bản rõ.

Chứng minh: Ta có $\mathcal{P}=C=\mathcal{K}=Z_{26}$ và với $0\le K\le 25$, quy tắc mã hoá e_K là $e_K(x)=x+K \bmod 26$ ($x\in Z_{26}$). Trước tiên tính phân bố p_C. Giả sử $y\in Z_{26}$, khi đó:

$$\begin{aligned} p_C(y) &= \sum_{K\in Z_{26}} p_{\mathcal{K}}(K)\,p_{\mathcal{P}}(d_K(y)) \\ &= \sum_{K\in Z_{26}} 1/26\,p_{\mathcal{P}}(y-K) \\ &= 1/26\sum_{K\in Z_{26}} p_{\mathcal{P}}(y-K) \end{aligned} \qquad (1.7)$$

Xét thấy với y cố định, các giá trị $y-K \bmod 26$ sẽ tạo thành một hoán vị của Z_{26} và $p_{\mathcal{P}}$ là một phân bố xác suất. Bởi vậy ta có:

$$\sum_{K\in Z_{26}} p_{\mathcal{P}}(y-K)=\sum_{K\in Z_{26}} p_{\mathcal{P}}(y)=1$$

Do đó: $p_C(y)=1/26$ với bất kỳ $y\in Z_{26}$.

Tiếp theo ta có:

$$p_C(y|x) = p_K(y - x \bmod 26) = 1/26$$

Với mọi x, y vì với mỗi cặp x, y khóa duy nhất K (khoá đảm bảo $e_K(x) = y$) là khoá $K = y - x \bmod 26$. Bây giờ sử dụng định lý Bayes, ta có thể dễ dàng tính:

$$\begin{aligned} p_C(x|y) &= \frac{p_{\mathcal{P}}(x)p_C(y|x)}{p_C(y)} \\ &= \frac{p_{\mathcal{P}}(x).(1/26)}{(1/26)} \\ &= p_{\mathcal{P}}(x) \end{aligned}$$

Bởi vậy, MDV có độ mật hoàn thiện.

Như vậy, mã dịch vòng là hệ mật không phá được miễn là chỉ dùng một khoá ngẫu nhiên đồng xác suất để mã hoá mỗi ký tự của bản rõ.

Sau đây sẽ nghiên cứu độ mật hoàn thiện trong trường hợp chung. Trước tiên thấy rằng, (sử dụng định lý Bayes) điều kiện để $p_{\mathcal{P}}(x|y) = p_{\mathcal{P}}(x)$ với mọi $x \in \mathcal{P}, y \in \mathcal{P}$ là tương đương với $p_C(y|x) = p_C(y)$ với mọi $x \in \mathcal{P}, y \in \mathcal{P}$.

Giả sử rằng $p_C(y) > 0$ với mọi $y \in C\,(p_C(y) = 0)$ thì bản mã sẽ không được dùng và có thể loại khỏi C). Cố định một giá trị nào đó $x \in \mathcal{P}$. Với mỗi $y \in C$ ta có $p_C(y|x) = p_C(y) > 0$. Bởi vậy, với mỗi $y \in C$ phải có ít nhất một khoá K và một x sao cho $e_K(x) = y$. Điều này dẫn đến $|\mathcal{K}| \geq |C|$. Trong một hệ mật bất kỳ ta phải có $|C| \geq |\mathcal{P}|$ vì mỗi quy tắc mã hoá là một đơn ánh. Trong trường hợp giới hạn, $|\mathcal{K}| = |C| = |\mathcal{P}|$, ta có định lý sau (Theo Shannon).

Định lý 1.3:

Giả sử $(\mathcal{P}, C, \mathcal{K}, \mathcal{E}, \mathcal{D})$ là một hệ mật , trong đó $|\mathcal{K}| = |C| = |\mathcal{P}|$. Khi đó, hệ mật có độ mật hoàn thiện khi và chỉ khi khoá K được dùng với xác suất như nhau bằng $1/|\mathcal{K}|$, và với mỗi $x \in \mathcal{P}$, mỗi $y \in C$ có một khoá duy nhất K sao cho $e_K(x) = y$.

Chứng minh

Giả sử hệ mật đã cho có độ mật hoàn thiện. Như đã thấy ở trên, với mỗi $x \in \mathcal{P}$ và $y \in C$, phải có ít nhất một khoá K sao cho $e_K(x) = y$. Bởi vậy ta có bất đẳng thức:

$$|C| = |\{e_K(x) : K \in \mathcal{K}\}| = |\mathcal{K}|$$

Tuy nhiên, ta giả sử rằng $|C| = |\mathcal{K}|$, bởi vậy ta phải có:

$$|\{e_K(x) : K \in C\}| = |\mathcal{K}|$$

Tức là ở đây không tồn tại hai khoá K_1 và K_2 khác nhau để $e_{K_1}(x) = e_{K_2}(x) = y$. Như vậy ta đã chứng tỏ được rằng, với bất kỳ $x \in \mathcal{P}$ và $y \in C$ có đúng một khoá K để $e_K(x) = y$.

Ký hiệu $n = |\mathcal{K}|$. Giả sử $\mathcal{P} = \{x_i : 1 \le i \le n\}$ và cố định một giá trị $y \in C$ Ta có thể ký hiệu các khoá $K_1, K_2, ..., K_n$ sao cho $e_{K_i}(x_i) = y_i$, $1 \le i \le n$. Sử dụng định lý Bayes ta có:

$$p_{\mathcal{P}}(x_i|y) = \frac{p_C(y|x_i)p_{\mathcal{P}}(x_i)}{p_C(y)}$$
$$= \frac{p_{\mathcal{K}}(K_i).(p_{\mathcal{P}}(x_i))}{p_C(y)}$$

Xét điều kiện độ mật hoàn thiện $p_{\mathcal{P}}(x_i|y) = p_{\mathcal{P}}(x_i)$. Điều kiện này kéo theo $p_{\mathcal{K}}(K_i) = p_C(y)$ với $1 \le i \le n$. Tức là khoá được dùng với xác suất như nhau (chính bằng $p_C(y)$). Tuy nhiên vì số khoá là $n = |\mathcal{K}|$ nên ta có $p_{\mathcal{K}}(K) = 1/|\mathcal{K}|$ với mỗi $K \in \mathcal{K}$.

Ngược lại, giả sử hai điều giả định đều thoả mãn. Khi đó dễ dàng thấy được hệ mật có độ mật hoàn thiện với mọi phân bố xác suất bất kỳ của bản rõ (tương tự như chứng minh định lý 2.3). Các chi tiết dành cho bạn đọc xem xét.

Mật mã khoá sử dụng một lần của Vernam (One-Time-Pad: OTP) là một ví dụ quen thuộc về hệ mật có độ mật hoàn thiện. Gillbert Vernam lần đầu tiên mô tả hệ mật này vào năm 1917. Hệ OTP dùng để mã và giải mã tự động các bản tin điện báo. Điều thú vị là trong nhiều năm OTP được coi là một hệ mật không thể bị phá nhưng không thể chứng minh cho tới khi Shannon xây dựng được khái niệm về độ mật hoàn thiện hơn 30 năm sau đó.

Mô tả về hệ mật dùng một lần nêu trên hình 1.2.

Giả sử $n \ge 1$ là số nguyên và $\mathcal{P} = C = \mathcal{K} = (Z_2)^n$. Với $\mathcal{K} \in (Z_2)^n$, ta xác định $e_K(x)$ là tổng vector theo modulo 2 của K và x (hay tương đương với phép hoặc loại trừ của hai dãy bit tương ứng). Như vậy, nếu $x = (x_1, ..., x_n)$ và $K = (K_1, ..., K_n)$ thì:

$$e_K(x) = (x_1 + K_1, ..., x_n + K_n) \bmod 2$$

Phép mã hoá là đồng nhất với phép giải mã. Nếu $y = (y_1, ..., y_n)$ thì:

$$d_K(x) = (y_1 + K_1, ..., y_n + K_n) \bmod 2$$

Hình 1.2. Hệ mật sử dụng khoá một lần (OTP)

Sử dụng định lý 2.4, dễ dàng thấy rằng OTP có độ mật hoàn thiện. Hệ thống này rất hấp dẫn do dễ thực hiện mã và giải mã.

Vernam đã đăng ký phát minh của mình với hy vọng rằng nó sẽ có ứng dụng thương mại rộng rãi. Đáng tiếc là có những nhược điểm quan trọng đối với các hệ mật an toàn không điều kiện, chẳng hạn như OTP. Điều kiện $|\mathcal{K}| \ge |\mathcal{P}|$ có nghĩa là lượng khóa (cần được thông

báo một cách bí mật) cũng lớn như bản rõ. Ví dụ , trong trường hợp hệ OTP, ta cần n bit khoá để mã hoá n bit của bản rõ. Vấn đề này sẽ không quan trọng nếu có thể dùng cùng một khoá để mã hoá các bản tin khác nhau; tuy nhiên, độ an toàn của các hệ mật an toàn không điều kiện lại phụ thuộc vào một thực tế là mỗi khoá chỉ được dùng cho một lần mã. Ví dụ OTP không thể đứng vững trước tấn công chỉ với bản rõ đã biết vì ta có thể tính được K bằng phép hoặc loại trừ xâu bit bất kỳ x và $e_K(x)$. Bởi vậy, cần phải tạo một khóa mới và thông báo nó trên một kênh bảo mật đối với mỗi bản tin trước khi gửi đi. Điều này tạo ra khó khăn cho vấn đề quản lý khoá và gây hạn chế cho việc sử dụng rộng rãi OTP. Tuy nhiên OTP vẫn được áp dụng trong lĩnh vực quân sự và ngoại giao, ở những lĩnh vực này độ an toàn không điều kiện có tầm quan trọng rất lớn.

Lịch sử phát triển của mật mã học là quá trình cố gắng tạo các hệ mật có thể dùng một khoá để tạo một xâu bản mã tương đối dài (tức có thể dùng một khoá để mã nhiều bản tin) nhưng chí ít vẫn còn giữ được độ an toàn tính toán. Chuẩn mã dữ liệu (DES) là một hệ mật thuộc loại này.

1.4.2. Entropy

Trong phần trước ta đã thảo luận về khái niệm độ mật hoàn thiện và đặt mối quan tâm vào một trường hợp đặc biệt, khi một khoá chỉ được dùng cho một lần mã. Bây giờ ta sẽ xét điều sẽ xảy ra khi có nhiều bản rõ được mã bằng cùng một khoá và bằng cách nào mà thám mã có thể thực hiện có kết quả phép tấn công chỉ với bản mã trong thời gian đủ lớn.

Công cụ cơ bản trong nghiên cứu bài toán này là khái niệm Entropy. Đây là khái niệm trong lý thuyết thông tin do Shannon đưa ra vào năm 1948. Có thể coi Entropy là đại lượng đo thông tin hay còn gọi là độ bất định. Nó được tính như một hàm của phân bố xác suất.

Giả sử ta có một biến ngẫu nhiên X nhận các giá trị trên một tập hữu hạn theo một phân bố xác suất $p(X)$. Thông tin thu nhận được bởi một sự kiện xảy ra tuân theo một phân bố $p(X)$ là gì? Tương tự, nếu sự kiện còn chưa xảy ra thì cái gì là độ bất định và kết quả bằng bao nhiêu? Đại lượng này được gọi là Entropy của X và được kí hiệu là $H(X)$.

Các ý tưởng này có vẻ như khá trừu tượng, bởi vậy ta sẽ xét một ví dụ cụ thể hơn. Giả sử biến ngẫu nhiên X biểu thị phép tung đồng xu. Phân bố xác suất là: p(mặt xấp) = p(mặt ngửa) = 1/2. Có thể nói rằng, thông tin (hay Entropy) của phép tung đồng xu là một bit vì ta có thể mã hoá mặt xấp bằng 1 và mặt ngửa bằng 0. Tương tự Entropy của n phép tung đồng tiền có thể mã hoá bằng một xâu bit có độ dài n .

Xét một ví dụ phức tạp hơn một chút. Giả sử ta có một biến ngẫu nhiên X có 3 giá trị có thể là x_1, x_2, x_3 với các xác suất tương ứng bằng 1/2, 1/4, 1/4. Cách mã hiệu quả nhất của 3 biến cố này là mã hoá x_1 là 0, mã của x_2 là 10 và mã của x_3 là 11. Khi đó số bit trung bình trong phép mã hoá này là:

$$1/2 \times 1 + 1/4 \times 2 + 1/4 \times 2 = 3/2.$$

Các ví dụ trên cho thấy rằng, một biến cố xảy ra với xác suất 2^{-n} có thể mã hoá được bằng một xâu bit có độ dài n . Tổng quát hơn, có thể coi rằng, một biến cố xảy ra với xác suất

p có thể mã hoá bằng một xâu bit có độ dài xấp xỉ $-\log_2 p$. Nếu cho trước phân bố xác suất tuỳ ý $p_1, p_2, ..., p_n$ của biến ngẫu nhiên X, khi đó độ đo thông tin là trọng số trung bình của các lượng $-\log_2 p_i$. Điều này dẫn tới định nghĩa hình thức hoá sau.

Định nghĩa 1.6:

Giả sử X là một biến ngẫu nhiên lấy các giá trị trên một tập hữu hạn theo phân bố xác suất p(X). Khi đó entropy của phân bố xác suất này được định nghĩa là lượng:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 P_i \tag{1.8}$$

Nếu các giá trị có thể của X là x_i, $1 \le i \le n$ thì ta có:

$$H(X) = -\sum_{i=1}^{n} p(X = x_i) \log_2 P(X = x_i) \tag{1.9}$$

Nhận xét:

Nhận thấy rằng $\log_2 p_i$ không xác định nếu $p_i = 0$. Bởi vậy đôi khi entropy được định nghĩa là tổng tương ứng trên tất cả các xác suất khác 0. Vì $\lim_{x \to 0} x \log_2 x = 0$ nên trên thực tế cũng không có trở ngại gì nếu cho $p_i = 0$ với giá trị i nào đó. Tuy nhiên ta sẽ tuân theo giả định là khi tính entropy của một phân bố xác suất p_i, tổng trên sẽ được lấy trên các chỉ số i sao cho $p_i \neq 0$. Ta cũng thấy rằng việc chọn cơ số của logarit là tuỳ ý; cơ số này không nhất thiết phải là 2. Một cơ số khác sẽ chỉ làm thay đổi giá trị của entropy đi một hằng số.

Chú ý rằng, nếu $p_i = 1/n$ với $1 \le i \le n$ thì $H(X) = \log_2 n$. Cũng dễ dàng thấy rằng $H(X) \ge 0$ và $H(X) = 0$ khi và chỉ khi $p_i = 1$ với một giá trị i nào đó và $p_j = 0$ với mọi $j \neq i$.

Xét entropy của các thành phần khác nhau của một hệ mật. Ta có thể coi khoá là một biến ngẫu nhiên *K* nhận các giá trị tuân theo phân bố xác suất p_K và bởi vậy có thể tính được $H(K)$. Tương tự ta có thể tính các entropy $H(P)$ và $H(C)$ theo các phân bố xác suất tương ứng của bản mã và bản rõ.

Ví dụ 1.2: (tiếp)

Ta có:

$$\begin{aligned} H(P) &= -1/4 \log_2 1/4 - 3/4 \log_2 3/4 \\ &= -1/4(-2) - -3/4(\log_2 3 - 2) \\ &= 2 - 3/4 \log_2 3 \\ &\approx 0,81 \end{aligned}$$

bằng các tính toán tương tự, ta có $H(K) = 1,5$ và $H(C) \approx 1,85$.

1.4.2.1. Các tính chất của Entropy

Trong phần này sẽ chứng minh một số kết quả quan trọng liên quan đến Entropy. Trước tiên ta sẽ phát biểu bất đẳng thức Jensen. Đây là một kết quả cơ bản và rất hữu ích. Bất đẳng thức Jensen có liên quan đến hàm lồi có định nghĩa như sau.

Định nghĩa 1.7:

Một hàm có giá trị thực f là lồi trên khoảng I nếu:

$$f\left(\frac{x+y}{2}\right) \geq \frac{f(x)+f(y)}{2} \tag{1.10}$$

với mọi $x, y \in I$. f là hàm lồi thực sự trên khoảng I nếu:

$$f\left(\frac{x+y}{2}\right) > \frac{f(x)+f(y)}{2} \tag{1.11}$$

với mọi $x, y \in I$, $x \neq y$.

Sau đây ta sẽ phát biểu mà không chứng minh bất đẳng thức Jensen.

Định lý 1.4: (Bất đẳng thức Jensen).

Giả sử h là một hàm lồi thực sự và liên tục trên khoảng l,

$$\sum_{i=1}^{n} a_i = 1$$

và $a_i > 0; 1 \leq i \leq n$ Khi đó:

$$\sum_{i=1}^{n} a_i f(x_i) \leq f\left(\sum_{i=1}^{n} a_i x_i\right) \tag{1.12}$$

trong đó $x_i \in I; 1 \leq i \leq n$. Ngoài ra dấu "=" chỉ xảy ra khi và chỉ khi $x_1 = x_2 = ... = x_n$

Bây giờ ta sẽ đưa ra một số kết quả về Entropy. Trong định lý sau sẽ sử dụng khẳng định: hàm $\log_2 x$ là một hàm lồi thực sự trong khoảng (0, ∞) (Điều này dễ dàng thấy được từ những tính toán sơ cấp vì đạo hàm cấp 2 của hàm logarith là âm trên khoảng (0, ∞)).

Định lý 1.5:

Giả sử X là biến ngẫu nhiên có phân bố xác suất $p_1, p_2, ..., p_n$ trong đó $p_i > 0;\ 1 \leq i \leq n$. Khi đó $H(X) < \log_2 n$. Dấu "=" xảy ra khi và chỉ khi $p_i = 1/n$, $1 \leq i \leq n$

Chứng minh:

Áp dụng bất đẳng thức Jensen, ta có:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i = \sum_{i=1}^{n} p_i \log_2 (1/p_i)$$
$$\leq \log_2 \sum_{i=1}^{n} (p_i \times 1/p_i)$$
$$= \log_2 n$$

Ngoài ra, dấu "=" chỉ xảy ra khi và chỉ khi $p_i = 1/n$, $1 \leq i \leq n$.

Định lý 1.6:

$$H(X,Y) \leq H(X) + H(Y) \tag{1.13}$$

Đẳng thức (dấu "=") xảy ra khi và chỉ khi X và Y là các biến cố độc lập

Chứng minh.

Giả sử X nhận các giá trị $x_i, 1 \leq i \leq m$; Y nhận các giá trị $y_j, 1 \leq j \leq n$. Kí hiệu: $p_i = p(X = x_i), 1 \leq i \leq m$ và $q_i = p(Y = y_j), 1 \leq j \leq n$. Kí hiệu $r_{ij} = p(X = x_i, Y = y_j)$, $1 \leq i \leq m$, $1 \leq j \leq n$. (Đây là phân bố xác suất hợp).

Nhận thấy rằng

$$p_i = \sum_{j=1}^{n} r_{ij} \; ; \; (1 \leq i \leq m)$$

và

$$q_j = \sum_{i=1}^{m} r_{ij} \; ; \; (1 \leq j \leq n)$$

Ta có

$$H(X) + H(Y) = -(\sum_{i=1}^{m} p_i \log_2 p_i + \sum_{j=1}^{n} q_j \log_2 q_j)$$
$$= -(\sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} \log_2 p_i + \sum_{j=1}^{n} \sum_{i=1}^{m} r_{ij} \log_2 q_j)$$
$$= -\sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} \log_2 p_i q_j$$

Mặt khác:

$$H(X,Y) = -\sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} \log_2 r_{ij}$$

Kết hợp lại ta thu được kết quả sau:

$$H(X,Y)-H(X)-H(Y)=\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2(1/r_{ij})+\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2 p_i q_j$$
$$\le \log_2\sum_{i=1}^{m}\sum_{j=1}^{n}p_i q_j$$
$$=\log_2 1$$
$$=0$$

(Ở đây đã áp dụng bất đẳng thức Jensen khi biết rằng các r_{ij} tạo nên một phân bố xác suất).

Khi đẳng thức xảy ra, có thể thấy rằng phải có một hằng số c sao cho $p_{ij}/r_{ij}=c$ với mọi i,j . Sử dụng đẳng thức sau:

$$=\sum_{i=1}^{m}\sum_{j=1}^{n}r_{ij}\log_2(p_i q_j/r_{ij})$$
$$\sum_{j=1}^{n}\sum_{i=1}^{m}r_{ij}=\sum_{j=1}^{n}\sum_{i=1}^{m}p_i q_j=1$$

Điều này dẫn đến $c=1$. Bởi vậy đẳng thức (dấu "=") sẽ xảy ra khi và chỉ khi $r_{ij}=p_j q_j$, nghĩa là:

$$p\left(X=x_j,Y=y_j\right)=p\left(X=x_j\right)p\left(Y=y_j\right)$$

với $1\le i\le m, 1\le j\le n$. Điều này có nghĩa là *X* và *Y* độc lập.

Tiếp theo ta sẽ đưa ra khái niệm Entropy có điều kiện

Định nghĩa 1.8:

Giả sử X và Y là hai biến ngẫu nhiên. Khi đó với giá trị xác định bất kỳ y của Y, ta có một phân bố xác suất có điều kiện $p(X|y)$. *Rõ ràng là:*

$$H(X|y)=-\sum_{x}p(x|y)\log_2 p(x|y) \tag{1.14}$$

Ta định nghĩa Entropy có điều kiện H(X|Y) là trung bình có trọng số (ứng với các xác suất $p(y)$*) của Entropy H(X|y) trên mọi giá trị có thể y. H(X|y) được tính bằng:*

$$H(X|Y)=-\sum_{y}\sum_{x}p(y)p(x|y)\log_2 p(x|y) \tag{1.15}$$

Entropy có điều kiện đo lượng thông tin trung bình về X do Y mang lại.

Sau đây là hai kết quả trực tiếp (Bạn đọc có thể tự chứng minh)

Định lý 1.7:

$$H\left(X,Y\right)=H\left(Y\right)+H\left(X|Y\right) \tag{1.16}$$

Hệ quả 1.2.

$$H(X|Y) \le H(X)$$

Dấu bằng chỉ xảy ra khi và chỉ khi X và Y độc lập.

1.4.2.2. Các khoá giả và khoảng duy nhất

Trong phần này chúng ta sẽ áp dụng các kết quả về Entropy ở trên cho các hệ mật. Trước tiên sẽ chỉ ra một quan hệ cơ bản giữa các Entropy của các thành phần trong hệ mật. Entropy có điều kiện $H(K|C)$ được gọi là độ bất định về khoá. Nó cho ta biết về lượng thông tin về khoá thu được từ bản mã.

Định lý 1.8:

Giả sử $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ *là một hệ mật. Khi đó:*

$$H(K|C) = H(K) + H(P) - H(C) \tag{1.17}$$

Chứng minh:

Trước tiên ta thấy rằng $H(K,P,C) = H(C|K,P) + H(K,P)$. Do $y = e_K(x)$ nên khoá và bản rõ sẽ xác định bản mã duy nhất. Điều này có nghĩa là $H(C|K,P) = 0$. Bởi vậy $H(K,P,C) = H(K,P)$. Nhưng K và P độc lập nên $H(K,P) = H(K) + H(P)$. Vì thế:

$$H(K,P,C) + H(K,P) = H(K) + H(P)$$

Tương tự vì khoá và bản mã xác định duy nhất bản rõ (tức $x = d_K(y)$) nên ta có $H(K,P,C) = 0$ và bởi vậy $H(K,P,C) = H(K,P)$. Bây giờ ta sẽ tính như sau:

$$\begin{aligned} H(K|C) &= H(K,C) - H(C) \\ &= H(K,P,C) - H(C) \\ &= H(K) + H(P) - H(C) \end{aligned}$$

Đây là nội dung của định lý.

Ta sẽ quay lại ví dụ 2.1 để minh hoạ kết quả này.

Ví dụ 1.1 (tiếp)

Ta đã tính được $H(P) \approx 0,81$, $H(K) = 1,5$ và $H(C) \approx 1,85$. Theo định lý 1.8 ta có $H(K|,C) \approx 1,5 + 0,81 - 0,85 \approx 0,46$. Có thể kiểm tra lại kết quả này bằng cách áp dụng định nghĩa về Entropy có điều kiện như sau. Trước tiên cần phải tính các xác suất xuất $p(K_j|j)$, $1 \le i \le 3$; $1 \le j \le 4$. Để thực hiện điều này có thể áp dụng định lý Bayes và nhận được kết quả như sau:

$$p(K_1|1)=1 \qquad p(K_2|1)=0 \qquad p(K_3|1)=0$$

$$p(K_1|2)=6/7 \qquad p(K_2|2)=6/7 \qquad p(K_3|2)=0$$

$$p(K_1|3)=0 \qquad p(K_2|3)=3/4 \qquad p(K_3|3)=1/4$$

$$p(K_1|4)=0 \qquad p(K_2|4)=0 \qquad p(K_3|4)=1$$

Bây giờ ta tính:

$$H(K|C) = 1/8\times 0+7/16\times 0,59+1/4\times 0,81+3/16\times 0 = 0,46$$

Giá trị này bằng giá trị được tính theo định lý 2.10.

Giả sử $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ là hệ mật đang được sử dụng. Một xâu của bản rõ $x_1,x_2,..,x_n$ sẽ được mã hoá bằng một khoá để tạo ra bản mã $y_1,y_2,...,y_n$. Nhớ lại rằng, mục đích cơ bản của thám mã là phải xác định được khoá. Ta xem xét các phương pháp tấn công chỉ với bản mã và coi Oscar có khả năng tính toán vô hạn. Ta cũng giả sử Oscar biết bản rõ là một văn bản theo ngôn ngữ tự nhiên (chẳng hạn văn bản tiếng Anh). Nói chung Oscar có khả năng rút ra một số khoá nhất định (các khoá có thể hay các khoá chấp nhận được) nhưng trong đó chỉ có một khoá đúng, các khoá có thể còn lại (các khoá không đúng) được gọi là các khoá giả.

Ví dụ, giả sử Oscar thu được một xâu bản mã WNAJW mã bằng phương pháp mã dịch vòng. Dễ dàng thấy rằng, chỉ có hai xâu bản rõ có ý nghĩa là *river* và *arena* tương ứng với các khoá F(= 5) và W(= 22). Trong hai khoá này chỉ có một khoá đúng, khoá còn lại là khoá giả. (Trên thực tế, việc tìm một bản mã của MDV có độ dài 5 và 2 bản giải mã có nghĩa không phải quá khó khăn, bạn đọc có thể tìm ra nhiều ví dụ khác). Mục đích của ta là phải tìm ra giới hạn cho số trung bình các khoá giả. Trước tiên, phải xác định giá trị này theo Entropy (cho một kí tự) của một ngôn ngữ tự nhiên L (kí hiệu là H_L). H_L là lượng thông tin trung bình trên một kí tự trong một xâu có nghĩa của bản rõ. (Chú ý rằng, một xâu ngẫu nhiên các kí tự của bảng chữ cái sẽ có Entropy trên một kí tự bằng $\log_2 26 \approx 4,76$). Ta có thể lấy H(P) là xấp xỉ bậc nhất cho H_L. Trong trường hợp L là Anh ngữ, ta tính được $H(P) \approx 4,19$.

Dĩ nhiên các kí tự liên tiếp trong một ngôn ngữ không độc lập với nhau và sự tương quan giữa các kí tự liên tiếp sẽ làm giảm Entropy. Ví dụ, trong Anh ngữ, chữ Q luôn kéo theo sau là chữ U. Để làm xấp xỉ bậc hai, tính Entropy của phân bố xác suất của tất cả các bộ đôi rồi chia cho 2. Một cách tổng quát, ta định nghĩa P^n là biến ngẫu nhiên có phân bố xác suất của tất cả các bộ n của bản rõ. Ta sẽ sử dụng tất cả các định nghĩa sau:

Định nghĩa 1.9:

Giả sử L là một ngôn ngữ tự nhiên. Entropy của L được xác định là lượng sau:

$$H_L = \lim_{n\to\infty} \frac{H(P^n)}{n} \tag{1.18}$$

Độ dư của L là: $$R_L = 1-\left(H_L/\log_2|\mathcal{P}|\right) \tag{1.19}$$

Nhận xét: H_L đo Entropy trên mỗi kí tự của ngôn ngữ L. Một ngôn ngữ ngẫu nhiên sẽ có Entropy là $\log_2 |\mathcal{P}|$. Bởi vậy đại lượng R_L đo phần "kí tự vượt trội" là phần dư.

Trong trường hợp Anh ngữ, dựa trên bảng chứa một số lớn các bộ đôi và các tần số, ta có thể tính được $H(P^2)$. Ước lượng theo cách này, ta tính được $H(P^2) \approx 3,90$. Cứ tiếp tục như vậy bằng cách lập bảng các bộ ba v.v... ta thu được ước lượng cho H_L. Trên thực tế, bằng nhiều thực nghiệm khác nhau, ta có thể đi tới kết quả sau $1,0 \le H_L \le 1,5$. Tức là lượng thông tin trung bình trong tiếng Anh vào khoảng 1 bit tới 1,5 bit trên mỗi kí tự.

Giả sử lấy 1,25 là giá trị ước lượng của giá trị của H_L. Khi đó độ dư vào khoảng 0,75. Tức là tiếng Anh có độ dư vào khoảng 75%! (Điều này không có nghĩa loại bỏ tuỳ ý 3 trên 4 kí tự của một văn bản tiếng Anh mà vẫn có khả năng đọc được nó. Nó chỉ có nghĩa là tìm được một phép mã Huffman cho các bộ n với n đủ lớn, phép mã này sẽ nén văn bản tiếng Anh xuống còn 1/4 độ dài của bản gốc).

Với các phân bố xác suất $\mathcal{C}^n$ đã cho trên $\mathcal{K}$ và $\mathcal{P}^n$. Có thể xác định phân bố xác suất trên là tập các bộ n của bản mã. (Ta đã làm điều này trong trường hợp $n = 1$). Ta đã xác định P^n là biến ngẫu nhiên biểu diễn bộ n của bản rõ. Tương tự C^n là biến ngẫu nhiên biểu thị bộ n của bản mã.

Với $y \in C^n$, định nghĩa: $K(y) = \left\{K \in \mathcal{K}; \exists x \in \mathcal{P}^n, p_{\mathcal{P}^n(x)} > 0, e_K(x) = y\right\}$ nghĩa là $K(y)$ là tập các khoá K sao cho y là bản mã của một xâu bản rõ độ dài n có nghĩa, tức là tập các khoá "có thể" với y là bản mã đã cho. Nếu y là dãy quan sát được của bản mã thì số khoá giả sẽ là $|K(y)0| - 1$ vì chỉ có một khoá là khoá đúng trong số các khoá có thể. Số trung bình các khoá giả (trên tất cả các xâu bản mã có thể độ dài n) được kí hiệu là $\overline{s_n}$ và nó được tính như sau:

$$\begin{aligned}\overline{s_n} &= \sum_{y \in C^n} p(y)\left(|K(y)| - 1\right) \\ &= \sum_{y \in C^n} p(y)|K(y)| - \sum_{y \in C^n} p(y) \\ &= \sum_{y \in C^n} p(y)|K(y)| - 1\end{aligned}$$

Từ định lý 1.8 ta có:

$$H\left(K|C^n\right) = H(K) + H\left(P^n\right) - H\left(C^n\right)$$

Có thể dùng ước lượng sau:

$$H\left(P^n\right) \approx nH_L = n(1 - R_L)\log_2 |\mathcal{P}|$$

với điều kiện n đủ lớn. Hiển nhiên là:

$$H\left(C^n\right) \le n \log_2 |C|$$

Khi đó nếu $|\mathcal{P}| = |C|$ thì:

$$H(K|C^n) \geq H(K) - nR_L \log_2 |\mathcal{P}| \tag{1.20}$$

Tiếp theo xét quan hệ của lượng $H(K|C^n)$ với số khoá giả s_n. Ta có:

$$\begin{aligned} H(K|C^n) &= \sum_{y \in C^n} p(y)(|K|y) \\ &\leq \sum_{y \in C^n} p(y) \log_2 |K(y)| \\ &\leq \sum_{y \in C^n} p(y) |K(y)| \\ &= \log_2 (\overline{s_n} + 1) \end{aligned}$$

Ở đây ta áp dụng bất đẳng thức Jensen (định lý 1.5) với $f(x) = \log_2(x)$. Bởi vậy ta có bất đẳng thức sau:

$$H(K|C^n) \leq \log_2(\overline{s_n} + 1) \tag{1.21}$$

Kết hợp hai bất đẳng thức (1.20) và (1.21), ta có:

$$\log_2(\overline{s_n} + 1) \geq H(K) - nR_L \log_2 |P|$$

Trong trường hợp các khoá được chọn đồng xác suất (Khi đó $H(K)$ có giá trị lớn nhất) ta có kết quả sau.

Định lý 1.9:

Giả sử $(\mathcal{P}, C, K, \mathcal{E}, \mathcal{D})$ là một hệ mật trong đó $|C| = |\mathcal{P}|$ và các khoá được chọn đồng xác suất. Giả sử R_L là độ dư của ngôn ngữ gốc. Khi đó với một xâu bản mã độ dài n cho trước (n là số đủ lớn), số trung bình các khoá giả s_n thoả mãn bất đẳng thức như sau:

$$\overline{s_n} \geq \{|K| / (|P| nR_L)\} - 1$$

Lượng $|K| / (|P| nR_L) - 1$ tiến tới 0 theo hàm mũ khi n tăng. Ước lượng này có thể không chính xác với các giá trị n nhỏ. Đó là do $H(P^n)/n$ không phải là một ước lượng tốt cho H_L nếu n nhỏ.

Ta đưa ra đây một khái niệm nữa

Định nghĩa 1.10:

Khoảng duy nhất của một hệ mật được định nghĩa là giá trị của n mà ứng với giá trị này, số khoá giả trung bình bằng 0 (kí hiệu giá trị này là n_0). Điều đó có nghĩa là n_0 là độ dài trung bình cần thiết của bản mã để thám mã có thể tính toán khoá một cách duy nhất với thời gian đủ lớn.

Nếu đặt $s_n = 0$ trong định lý 1.11 và giải theo n ta sẽ nhận được ước lượng cho khoảng duy nhất:

$$n_0 \approx \log_2 |\mathcal{K}| / R_L \log_2 |\mathcal{P}|$$

Ví dụ với mã thay thế, ta có $|\mathcal{P}| = 26$ và $|\mathcal{K}| = 26!$. Nếu lấy $R_L = 0,75$ thì ta nhận được ước lượng cho khoảng duy nhất bằng:

$$n_0 \approx 88,4/(0,75 \times 4,7) \approx 25$$

Điều đó có nghĩa là thông thường nếu mã thám có được xâu bản mã với độ dài tối thiểu là 25, anh ta có thể nhận được bản giải mã duy nhất.

BÀI TẬP CHƯƠNG 1.

Bài 1.1: Cho n là một số nguyên dương. Một hình vuông lớn latin cấp $n(L)$ là một bảng $n \times n$ các số nguyên $1,...,n$ sao cho mỗi một số trong n số nguyên này chỉ xuất hiện đúng một lần ở hàng và mỗi cột của *L*. Ví dụ hình vuông Latin cấp 3 có dạng:

1	2	3
3	1	2
2	3	1

Với một hình vuông Latin *L* bất kỳ cấp n , ta có thể xác định một hệ mã tương ứng. Giả sử $\mathcal{P} = C = \mathcal{K} = \{1,...,n\}$. Với $1 \le i \le n$, quy tắc mã hóa e_1 được xác định là $e_1(j) = L(i,j)$ (Do đó mỗi hàng của L sẽ cho một quy tắc mã hóa).

Chứng minh rằng hệ mật hình vuông Latin này có độ mật hoàn thiện.

Bài 1.2: Hãy chứng tỏ rằng mã Affine có độ mật hoàn thiện

Bài 1.3: Giả sử một hệ mật đạt được độ hoàn thiện với phân bố xác suất p_0 nào đó của bản rõ. Hãy chứng tỏ rằng độ mật hoàn thiện vẫn còn giữ được đối với một phân bố xác suất bất kỳ của bản rõ.

Bài 1.4: Hãy chứng tỏ rằng nếu một hệ mật có độ hoàn thiện và $|\mathcal{K}| = |C| = |\mathcal{P}|$ thì mọi bản mã là đồng xác suất.

Bài 1.5: Hãy chứng tỏ rằng $H(X,Y) = H(Y) + H(X|Y)$. Sau đó hãy chứng minh bổ đề là $H(X|Y) \le H(X)$, đẳng thức chỉ xảy ra khi và chỉ khi *X* và *Y* độc lập.

Bài 1.5: Chứng minh rằng một hệ mật có độ mật hoàn thiện khi và chỉ khi $H(P|C) = H(P)$.

Bài 1.6: Chứng minh rằng trong một hệ mật $H\left(K\left|C\right.\right) \geq H\left(PC\right)$ (về mặt trực giác kết quả này nói rằng với bản mã cho trước độ bất định của thám mã về khóa ít nhất cũng lớn bằng độ bất định khi thám mã rõ).

Bài 1.7: Xét một hệ mật trong đó $\mathcal{P} = \{a,b,c\}$, $\kappa = \{K_1, K_2, K_3\}$ và $C = \{1,2,3,4\}$. Giả sử ma trận mã hóa như sau:

	a	b	c
K_1	1	2	3
K_2	2	3	4
K_3	3	4	1

Giả sử các khóa được chọn đồng xác suất và phân bố xác suất của bản rõ là $p_{\mathcal{P}}(a) = 1/2$, $p_{\mathcal{P}}(b) = 1/3$, $p_{\mathcal{P}}(c) = 1/6$. Hãy tính $H(P), H(C), H(K)$, $H\left(K\left|C\right.\right)$ và $H\left(P\left|C\right.\right)$.

LỜI NÓI ĐẦU

Trong sự phát triển của xã hội loài người, kể từ khi có sự trao đổi thông tin, an toàn thông tin trở thành một nhu cầu gắn liền với nó như hình với bóng. Từ thủa sơ khai, an toàn thông tin được hiểu đơn giản là giữ được bí mật và điều này được xem như một nghệ thuật chứ chưa phải là một ngành khoa học. Với sự phát triển của khoa học kỹ thuật và công nghệ, cùng với các nhu cầu đặc biệt có liên quan tới an toàn thông tin, ngày nay các kỹ thuật chính trong an toàn thông tin bao gồm:

- Kỹ thuật mật mã (Cryptography)
- Kỹ thuật ngụy trang (mã ẩn) (Steganography)
- Kỹ thuật tạo bóng mờ, thủy vân số (Watermarking)

Kỹ thuật mật mã nhằm đảm bảo ba dịch vụ an toàn cơ bản:

- Bí mật (Confidential)
- Xác thực (Authentication)
- Đảm bảo tính toàn vẹn (Intergrity)

Có thể thấy rằng mật mã học là một lĩnh vực khoa học rộng lớn có liên quan rất nhiều ngành toán học như: Đại số tuyến tính, Lý thuyết thông tin, Lý thuyết độ phức tạp tính toán...

Bởi vậy việc trình bày đầy đủ mọi khía cạnh của mật mã học trong khuôn khổ một giáo trình là một điều khó có thể làm được. Chính vì lý do đó, trong giáo trình này chúng tôi chỉ dừng ở mức mô tả ngắn gọn các thuật toán mật mã chủ yếu. Các thuật toán này hoặc đang được sử dụng trong các chương trình ứng dụng hiện nay hoặc không còn được dùng nữa, nhưng vẫn được xem như là một ví dụ hay, cho ta hình dung rõ hơn bức tranh tổng thể về sự phát triển của mật mã học cả trên phương diện lý thuyết và ứng dụng. Còn một nội dung rất lý thú chưa được nêu trong giáo trình này là vấn đề thám mã. Bạn đọc quan tâm có thể tham khảo thêm trong các tài liệu [1], [2], [3].

Nội dung giáo trình bao gồm sáu chương:

Chương I - Nhập môn mật mã học: Trình bày những khái niệm và sơ lược về mật mã học, độ phức tạp tính toán, và cơ sở lý thuyết thông tin trong các hệ mật.

Chương II - Mật mã khóa bí mật: Trình bày các phương pháp xử lý thông tin của các hệ mật khóa bí mật (hệ mật khóa đối xứng) bao gồm các thuật toán hoán vị, thay thế và chuẩn mã dữ liệu của Mỹ (DES) và AES.

Chương III - Mật mã khóa công khai: Trình bày một số bài toán một chiều và các thuật toán mật mã khóa công khai (hay mật mã khoá bất đối xứng) liên quan bao

gồm: hệ mật RSA, Merkle-Hellman, Rabin, McEliece, hệ mật trên đường cong elliptic...

Chương IV - Hàm băm, xác thực và chữ ký số: Khái quát về hàm băm, một số vấn đề về xác thực, đảm bảo tính toàn vẹn và chữ ký số.

Chương V - Các thủ tục và các chú ý trong thực tế khi sử dụng mã hóa

Chương VI: Các chuẩn và áp dụng

Sau mỗi chương đều có các câu hỏi và bài tập nhằm giúp cho bạn đọc nắm vững hơn các vấn đề đã được trình bày

Phần phụ lục cung cấp chương trình nguồn của DES.

Do thời gian và trình độ còn hạn chế, việc lựa chọn và trình bày các thuật toán này không thể tránh khỏi những khiếm khuyết nhất định. Rất mong bạn đọc đóng góp ý kiến về mặt cấu trúc, các nội dung được trình bày và các sai sót cụ thể.

Các đóng góp ý kiến xin gửi về

KHOA KỸ THUẬT ĐIỆN TỬ 1 - HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KM 10. ĐƯỜNG NGUYỄN TRÃI - HÀ ĐÔNG

Email: KhoaDT1@hn.vnn.vn

Hoặc: nguyenbinh1999@yahoo.com

Xin chân thành cảm ơn!

NGƯỜI BIÊN SOẠN

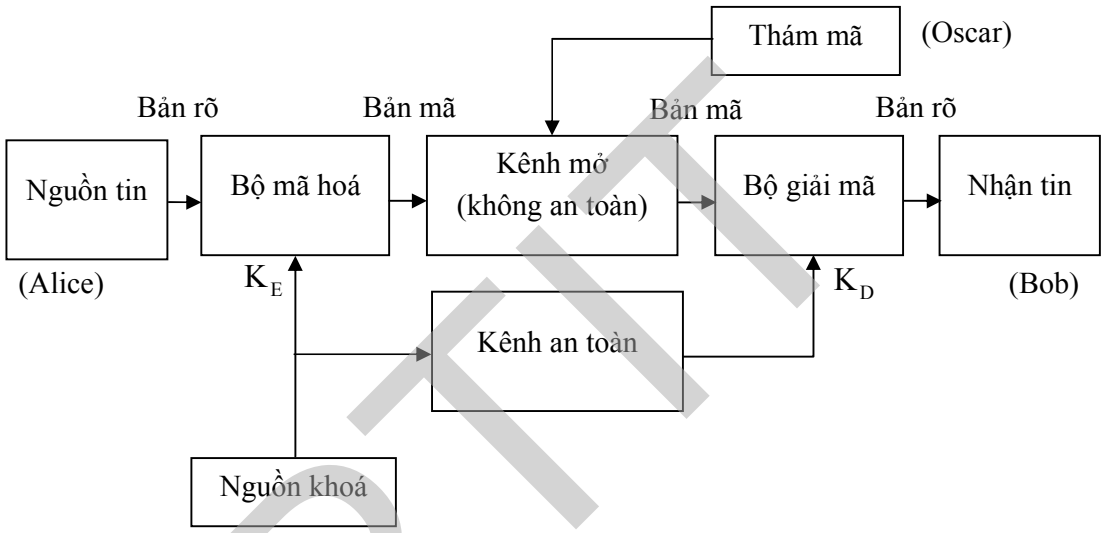
CHƯƠNG 2. MẬT MÃ KHÓA BÍ MẬT

Có ba phương pháp chính trong mật mã cổ điển (còn gọi là mật mã khoá riêng, mật mã khoá bí mật hay mật mã khóa đối xứng):

- Hoán vị
- Thay thế
- Xử lý bit (chủ yếu nằm trong các ngôn ngữ lập trình)

Ngoài ra còn có phương pháp hỗn hợp thực hiện kết hợp các phương pháp trên mà điển hình là chuẩn mã dữ liệu (DES – Data Encryption Standard) của Mỹ.

2.1. SƠ ĐỒ KHỐI MỘT HỆ TRUYỀN TIN MẬT



Hình 2.1. Sơ đồ hệ mật khóa bí mật

Định nghĩa 2.1:

Một hệ mật là một bộ 5 $(\mathcal{P},\mathcal{C},\mathcal{K},\mathcal{E},\mathcal{D})$ thoả mãn các điều kiện sau:

a) $\mathcal{P}$ là một tập hữu hạn các bản rõ có thể

b) $\mathcal{C}$ là một tập hữu hạn các bản mã có thể

c) $\mathcal{K}$ là một tập hữu hạn các khoá có thể (không gian khoá)

d) Đối với mỗi $k \in \mathcal{K}$ có một quy tắc mã $e_k \in \mathcal{E}$

$$e_k : \mathcal{P} \to \mathcal{C} \tag{2.1}$$

và một quy tắc giải mã tương ứng $d_k \in \mathcal{D}$

$$d_k : \mathcal{C} \to \mathcal{P} \tag{2.2}$$

sao cho: $d_k(e_k(x)) = x$ với $\forall x \in \mathcal{P}$.

2.2. MẬT MÃ THAY THẾ

2.2.1. Mật mã dịch vòng (MDV)

Giả sử $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbf{Z}_{26}$ với $0 \le k \le 25$, ta định nghĩa:

$$e_k(x) = x + k \bmod 26$$
$$d_k(y) = y - k \bmod 26 \qquad (2.3)$$
$$(x, y \in \mathbf{Z}_{26})$$

Ta sử dụng MDV (với modulo 26) để mã hoá một văn bản tiếng Anh thông thường bằng cách thiết lập sự tương ứng giữa các ký tự và các thặng dư theo mod 26 như sau:

Ký tự	A	B	C	D	E	F	G	H	I	J	K	L	M
Mã tương ứng	0	1	2	3	4	5	6	7	8	9	10	11	12
Ký tự	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Mã tương ứng	13	14	15	16	17	18	19	20	21	22	23	24	25

Ví dụ 2.1:

Giả sử khoá cho MDV là $k = 5$ và bản rõ là: **meet me at sunset**.

Trước tiên, ta biến đổi bản rõ thành chữ in hoa và biến đổi thành dãy các số nguyên theo bảng trên (không biến đổi dấu cách (space) giữa 2 từ):

12.4.4.19.12.4.0.19.18.20.13.18.4.19

Sau đó ta cộng 5 vào mỗi giá trị ở trên và rút gọn tổng theo mod 26, ta được dãy số sau:

17.9.9.24.17.9.5.24.23.25.18.23.9.24

Cuối cùng, ta lại biến đổi dãy số nguyên trên thành các ký tự tương ứng, ta có bản mã sau:

RJJY RJ FY XZSXJY

Để giải mã cho bản mã này, trước tiên ta biến bản mã thành dãy số nguyên rồi trừ mỗi giá trị cho 5 (rút gọn theo modulo 26), và cuối cùng là lại biến đổi lại dãy số nhận được này thành các ký tự.

Nhận xét:

Khi $k = 3$, hệ mật này thường được gọi là mã Caesar đã từng được Hoàng đế Caesar sử dụng.

MDV (theo mod 26) là không an toàn vì nó có thể bị thám theo phương pháp tìm khoá vét cạn (thám mã có thể dễ dàng thử mọi khoá d_k có thể cho tới khi tìm được bản rõ có nghĩa). Trung bình có thể tìm được bản rõ đúng sau khi thử khoảng $(26/2) = 13$ quy tắc giải mã.

Từ ví dụ trên ta thấy rằng, điều kiện cần để một hệ mật an toàn là phép tìm khoá vét cạn phải không thể thực hiện được. Tuy nhiên, một không gian khoá lớn vẫn chưa đủ để đảm bảm độ mật.

2.2.2. Mã thay thế (MTT)

Cho $\mathcal{P} = C = \mathbf{Z}_{26}$. $\mathcal{K}$ chứa mọi hoán vị có thể có của 26 ký tự từ 0 đến 25. Với mỗi phép hoán vị $\pi \in \mathcal{K}$, ta định nghĩa:

$$e_\pi(x) = \pi(x)$$

và

$$d_\pi(y) = \pi^{-1}(y)$$

trong đó π^{-1} là hoán vị ngược của π

Sau đây là một ví dụ về phép hoán vị ngẫu nhiên π tạo nên một hàm mã hoá (tương tự như trên, các ký tự của bản rõ được viết bằng chữ thường, còn các ký tự của bản mã được viết bằng chữ in hoa).

Ký tự bản rõ	a	b	c	d	e	f	g	h	i	j	k	l	m
Ký tự bản mã	X	N	Y	A	H	P	O	G	Z	Q	W	B	T
Ký tự bản rõ	n	o	p	q	r	s	t	u	v	w	x	y	z
Ký tự bản mã	S	F	L	R	C	V	M	U	E	K	J	D	I

Như vậy, $e_\pi(a) = X, e_\pi(b) = N, \ldots$

Hàm giải mã là phép hoán vị ngược. Điều này được thực hiện bằng cách viết hàng thứ hai lên trước rồi sắp xếp theo thứ tự chữ cái. Ta có:

Ký tự bản mã	A	B	C	D	E	F	G	H	I	J	K	L	M
Ký tự bản rõ	d	l	r	y	v	o	h	e	z	x	w	p	t
Ký tự bản mã	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Ký tự bản rõ	b	g	f	j	q	n	m	u	s	k	a	c	i

Ví dụ 2.2:

Với phép thay thế trên, từ bản rõ: **meet me at sunset**

ta thu được bản rõ sau: **THHM TH XM VUSHM**

Sử dụng phép hoán vị ngược, ta dễ dàng tìm lại được bản rõ ban đầu.

Mỗi khoá của mã thay thế là một phép hoán vị của 26 ký tự. Số các hoán vị này là $26! > 4.10^{26}$. Đây là một số rất lớn nên khó có thể tìm được khoá bằng phép tìm khoá vét cạn. Tuy nhiên, bằng phương pháp thống kê, ta có thể dễ dàng thám được các bản mã loại này.

2.2.3. Mật mã Vigenère

Trong hai hệ MDV và MTT ở trên, một khi khoá đã được chọn thì mỗi ký tự sẽ được ánh xạ vào một ký tự duy nhất. Vì vậy, các hệ trên còn được gọi là các hệ thay thế đơn biểu. Sau đây ta sẽ trình bày một hệ thay thế đa biểu được gọi là hệ mật Vigenere.

Sử dụng phép tương ứng $A \leftrightarrow 0, B \leftrightarrow 1, ..., Z \leftrightarrow 25$ mô tả ở trên, ta có thể gắn cho mỗi khoá k một chuỗi ký tự có độ dài m, được gọi là từ khoá. Mật mã Vigenère sẽ mã hoá đồng thời m ký tự: mỗi phần tử của bản rõ tương đương với m ký tự.

Ví dụ 2.3:

Giả sử $m = 6$ và từ khoá là CIPHER. Từ khoá này tương ứng với dãy số $k = (2,8,15,7,4,17)$. Giả sử bản rõ là:

meet me at sunset

Ta sẽ biến đổi các phần tử của bản rõ thành các thặng dư theo mod 26, viết chúng thành các nhóm 6 rồi cộng với từ khoá theo modulo 26 như sau:

12	4	4	19	12	4	0	19	18	20	13	18	4	19	Bản rõ
2	8	15	7	4	17	2	8	15	7	4	17	2	8	Khoá
14	12	19	0	16	21	2	1	7	1	17	9	6	1	Bản mã

Như vậy, dãy ký tự tương ứng với xâu bản mã sẽ là:

OMTA QV CB HBRJGB

Ta có thể mô tả mật mã Vigenère như sau:

Cho m là một số nguyên dương cố định nào đó.

Ta định nghĩa $P = C = K = (\mathbf{Z}_{26})^n$

Với khoá $k = (k_1, k_2, \ldots, k_m)$, ta xác định:

$e_k(x_1, x_2, \ldots, x_m) = (x_1 + k_1, x_2 + k_2, \ldots, x_m + k_m)$

và $d_k(y_1, y_2, \ldots, y_m) = (y_1 - k_1, y_2 - k_2, \ldots, y_m - k_m)$

trong đó tất cả các phép toán được thực hiện trong $\mathbf{Z}_{26}$.

Chú ý: Để giải mã, ta có thể dùng cùng từ khoá nhưng thay cho cộng, ta trừ nó theo modulo 26.

Ta thấy rằng, số các từ khoá có thể với độ dài m trong mật mã Vigenere là 26^m. Bởi vậy, thậm chí với m khá nhỏ, phương pháp tìm kiếm vét cạn cũng yêu cầu thời gian khá lớn. Ví dụ, với $m = 6$ thì không gian khoá cũng có kích thước lớn hơn 3.10^8 khoá.

2.2.3.1. Mã Affine

MDV là một trường hợp đặc biệt của MTT chỉ gồm 26 trong số 26! các hoán vị có thể của 26 phần tử. Một trường hợp đặc biệt khác của MTT là mã Affine được mô tả dưới đây. Trong mã Affine, ta giới hạn chỉ xét các hàm mã có dạng:

$$e(x) = ax + b \bmod 26;\ a, b \in \mathbf{Z}_{26}$$

Các hàm này được gọi là các hàm Affine (chú ý rằng khi $a = 1$, ta có MDV).

Để việc giải mã có thể thực hiện được, yêu cầu cần thiết là hàm Affine phải là đơn ánh. Nói cách khác, với bất kỳ $y \in \mathbf{Z}_{26}$, ta muốn có đồng nhất thức sau:

$$ax + b \equiv y \bmod 26$$

phải có nghiệm x duy nhất. Đồng dư thức này tương đương với:

$$ax = y - b \pmod{26}$$

Vì y thay đổi trên $\mathbf{Z}_{26}$ nên $y - b$ cũng thay đổi trên $\mathbf{Z}_{26}$. Bởi vậy, ta chỉ cần nghiên cứu phương trình đồng dư:

$$ax \equiv y \bmod 26$$

Ta biết rằng, phương trình này có một nghiệm duy nhất đối với mỗi y khi và chỉ khi $\text{ƯCLN}(a, 26) = 1$ (ở đây hàm ƯCLN là ước chung lớn nhất của các biến của nó).

Trước tiên ta giả sử rằng, $\text{ƯCLN}(a, 26) = d > 1$. Khi đó, đồng dư thức $ax \equiv 0 \bmod 26$ sẽ có ít nhất hai nghiệm phân biệt trong $\mathbf{Z}_{26}$ là $x = 0$ và $x = 26/d$. Trong trường hợp này, $e(x) = ax + b \bmod 26$ không phải là một hàm đơn ánh và bởi vậy nó không thể là hàm mã hoá hợp lệ.

Ví dụ 2.4:

Do $\text{ƯCLN}(4, 26) = 2$ nên $4x + 7$ không là hàm mã hoá hợp lệ: x và $x + 13$ sẽ mã hoá thành cùng một giá trị đối với bất kỳ $x \in \mathbf{Z}_{26}$.

Ta giả thiết $\text{ƯCLN}(a, 26) = 1$. Giả sử với x_1 và x_2 nào đó thoả mãn:

$$ax_1 \equiv ax_2 \bmod 26$$

Khi đó:

$$a(x_1 - x_2) \equiv 0 \bmod 26$$

bởi vậy

$$26 \mid a(x_1 - x_2)$$

Bây giờ ta sẽ sử dụng một tính chất của phép chia sau: Nếu $\text{ƯCLN}(a, b) = 1$ và $a \mid bc$ thì $a \mid c$. Vì $26 \mid a(x_1 - x_2)$ và $\text{ƯCLN}(a, 26) = 1$ nên ta có:

$$26 \mid (x_1 - x_2)$$

tức là

$$x_1 \equiv x_2 \bmod 26$$

Tới đây ta đã chứng tỏ rằng, nếu $\text{ƯCLN}(a, 26) = 1$ thì một đồng dư thức dạng $ax \equiv y \bmod 26$ chỉ có (nhiều nhất) một nghiệm trong $\mathbf{Z}_{26}$. Do đó, nếu ta cho x thay đổi trên

$\mathbf{Z}_{26}$ thì $ax \bmod 26$ sẽ nhận được 26 giá trị khác nhau theo modulo 26 và đồng dư thức $ax \equiv y \bmod 26$ chỉ có một nghiệm y duy nhất.

Không có gì đặc biệt đối với số 26 trong khẳng định này. Bởi vậy, bằng cách tương tự, ta có thể chứng minh được kết quả sau:

Định lý 2.1:

Đồng dư thức $ax \equiv b \bmod m$ *chỉ có một nghiệm duy nhất* $x \in \mathbf{Z}_m$ *với mọi* $b \in \mathbf{Z}_m$ *khi và chỉ khi* ƯCLN$(a,m) = 1$.

Vì $26 = 2 \times 13$ nên các giá trị $a \in \mathbf{Z}_{26}$ thoả mãn ƯCLN$(a,26) = 1$ là a = 1 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23 và 25. Tham số b có thể là một phần tử bất kỳ trong $\mathbf{Z}_{26}$. Như vậy , mã Affine có $12 \times 26 = 312$ khoá có thể (dĩ nhiên, con số này là quá nhỏ để bảo đảm an toàn).

Bây giờ, ta sẽ xét bài toán chung với modulo m . Ta cần một định nghĩa khác trong lý thuyết số.

Định nghĩa 2.2:

Giả sử $a \geq 1$ *và* $m \geq 2$ *là các số nguyên.* ƯCLN$(a,m) = 1$ *thì ta nói rằng* a *và* m *là nguyên tố cùng nhau. Số các số nguyên trong* $\mathbf{Z}_m$ *nguyên tố cùng nhau với* m *thường được ký hiệu là* $\varphi(m)$ *(hàm này được gọi là hàm phi-Euler).*

Một kết quả quan trọng trong lý thuyết số cho ta giá trị của $\varphi(m)$ theo các thừa số trong phép phân tích theo luỹ thừa các số nguyên tố của m . (Một số nguyên $p > 1$ là số nguyên tố nếu nó không có ước dương nào khác ngoài 1 và p). Mọi số nguyên $m > 1$ có thể phân tích được thành tích của các luỹ thừa các số nguyên tố theo cách duy nhất.

Ví dụ $60 = 2^3 \times 3 \times 5$ và $98 = 2 \times 7^2$).

Ta sẽ ghi lại công thức cho $\Phi(m)$ trong định lí sau:

Định lý 2.2:

Giả sử $m = \prod_{i=1}^{n} p_i^{e_i}$;

Trong đó các số nguyên tố p_i khác nhau và $e_i > 0; \quad 1 \leq i \leq n$. Khi đó :

$$\varphi(m) = \prod_i \left(p_i^{e_i} - p_i^{e_i - 1} \right) \tag{2.4}$$

Định lý này cho thấy rằng, số khoá trong mã Affine trên $\mathbf{Z}_{26}$ bằng $m.\varphi(m)$, trong đó $\varphi(m)$ được cho theo công thức trên. (Số các phép chọn của b là m và số các phép chọn của a là $\phi(m)$ với hàm mã hoá là $e(x) = ax + b$).

Ví dụ, khi $m = 60$, $\varphi(60) = 2 \times 2 \times 4 = 16$ và số các khoá trong mã Affine là 960.

Bây giờ, ta sẽ xét xem các phép toán giải mã trong mật mã Affine với modulo $m = 26$. Giả sử ƯCLN$(a,m) = 1$. Để giải mã cần giải phương trình đồng dư $y = ax + b \bmod 26$ theo x. Từ thảo luận trên thấy rằng, phương trình này có một nghiệm duy nhất trong $\mathbf{Z}_{26}$. Tuy nhiên, ta vẫn chưa biết một phương pháp hữu hiệu để tìm nghiệm. Điều cần thiết ở đây là có một thuật toán hữu hiệu để làm việc đó. Rất may là một số kết quả tiếp sau về số học modulo sẽ cung cấp một thuật toán giải mã hữu hiệu cần tìm.

Định nghĩa 2.3:

Giả sử $a \in \mathbf{Z}_m$. Phần tử nghịch đảo (theo phép nhân) của a là phần tử $a^{-1} \in \mathbf{Z}_m$ sao cho $aa^{-1} = a^{-1}a = 1 \bmod m$.

Bằng các lý luận tương tự như trên, có thể chứng tỏ rằng a có nghịch đảo theo modulo m khi và chỉ khi ƯCLN$(a,m) = 1$, và nếu nghịch đảo này tồn tại thì nó phải là duy nhất. Ta cũng thấy rằng, nếu $b = a^{-1}$ thì $a = b^{-1}$. Nếu p là số nguyên tố thì mọi phần tử khác không của $\mathbf{Z}_p$ đều có nghịch đảo. Một vành trong đó mọi phần tử khác 0 đều có nghịch đảo được gọi là một trường.

Trong [3] có một thuật toán hữu hiệu để tính các nghịch đảo của $\mathbf{Z}_m$ với m tuỳ ý. Tuy nhiên, trong $\mathbf{Z}_{26}$, chỉ bằng phương pháp thử và sai cũng có thể tìm được các nghịch đảo của các phần tử nguyên tố cùng nhau với 26:

$1^{-1} = 1,\ 3^{-1} = 9,\ 5^{-1} = 21,\ 7^{-1} = 15,\ 11^{-1} = 19,\ 17^{-1} = 23,\ 25^{-1} = 25$. (Có thể dễ dàng kiểm chứng lại điều này, ví dụ: $7 \times 15 = 105 \equiv 1 \bmod 26$, bởi vậy $7^{-1} = 15$).

Xét phương trình đồng dư $y \equiv ax + b \bmod 26$. Phương trình này tương đương với

$$ax \equiv y - b (\bmod 26)$$

Vì ƯCLN$(a,26) = 1$ nên a có nghịch đảo theo modulo 26. Nhân cả hai vế của đồng dư thức với a^{-1}, ta có:

$$a^{-1}(ax) \equiv a^{-1}(y - b) \bmod 26$$

Áp dụng tính kết hợp của phép nhân modulo:

$$a^{-1}(ax) \equiv (a^{-1}a)x = 1x = x$$

Kết quả là $x \equiv a^{-1}(y - b) \bmod 26$. Đây là một công thức tường minh cho x. Như vậy hàm giải mã là:

$$d(y) = a^{-1}(y - b) \bmod 26$$

Hình 2.2 cho mô tả đầy đủ về mã Affine. Sau đây là một ví dụ nhỏ.

Cho $P = C = \mathbf{Z}_{26}$ và giả sử:

$$K = \{ (a,b) \in \mathbf{Z}_{26} \times \mathbf{Z}_{26} : UCLN(a, 26) = 1 \}$$

Với $k = (a,b) \in K$, ta định nghĩa:

$$e_k(x) = ax + b \bmod 26$$

Và $$d_k(y) = a^{-1}(y - b) \bmod 26$$

$$x, y \in Z_{26}$$

Hình 2.2. Mã Affine

Ví dụ 2.5:

Giả sử $k = (7,3)$. Như đã nêu ở trên, $7^{-1} = 15$. Hàm mã hoá là:

$$e_k(x) = 7x + 3$$

Và hàm giải mã tương ứng là: $d_k(x) = 15(y - 3) = 15y - 19$

Ở đây, tất cả các phép toán đều thực hiện trên $\mathbf{Z}_{26}$. Ta sẽ kiểm tra liệu $d_k(e_k(x)) = x$ với mọi $x \in \mathbf{Z}_{26}$ không. Dùng các tính toán trên $\mathbf{Z}_{26}$ ta có:

$$d_k(e_k(x)) = d_k(7x + 3) = 15(7x + 3) - 19 = x + 45 - 19 = x$$

Để minh hoạ, ta hãy mã hoá bản rõ **"hot"**. Trước tiên, biến đổi các chữ **h, o, t** thành các thặng dư theo modulo 26. Ta được các số tương ứng là 7, 14 và 19. Bây giờ sẽ mã hoá:

$$7 \times 7 + 3 \bmod 26 = 52 \bmod 26 = 0$$
$$7 \times 14 + 3 \bmod 26 = 101 \bmod 26 = 23$$
$$7 \times 19 + 3 \bmod 26 = 136 \bmod 26 = 6$$

Bởi vậy, ba ký hiệu của bản mã là 0, 23 và 6, tương ứng với xâu ký tự **AXG**. Việc giải mã sẽ do bạn đọc thực hiện như một bài tập.

2.3. MẬT MÃ HOÁN VỊ (MHV)

Khác với MTT, ý tưởng của MHV là giữ các ký tự của bản rõ không thay đổi nhưng sẽ thay đổi vị trí của chúng bằng cách sắp xếp lại các ký tự này. Ở đây không có một phép toán đại số nào cần thực hiện khi mã hoá và giải mã.

Ví dụ 2.6:

Giả sử $m = 6$ và khoá là phép hoán vị sau:

1	2	3	4	5	6
3	5	1	6	4	2

Khi đó, phép hoán vị ngược sẽ là:

1	2	3	4	5	6
3	6	1	5	2	4

Giả sử ta có bản rõ: **a second class carriage on the train**

Trước tiên, ta nhóm bản rõ thành các nhóm 6 ký tự:

asecon | dclass | carria | geonth | etrain

Sau đó, mỗi nhóm 6 chữ cái lại được sắp xếp lại theo phép hoán vị π, ta có:

EOANCS | LSDSAC | RICARA | OTGHNE | RIENAT

Cuối cùng, ta có bản mã sau:

EOANCSLSDSACRICARAOTGHNERIENAT

Khi sử dụng phép hoán vị ngược π^{-1} trên dãy bản mã (sau khi đã nhóm lại theo các nhóm 6 ký tự), ta sẽ nhận lại được bản rõ ban đầu.

Từ ví dụ trên, ta có thể định nghĩa MHV như sau:

Cho m là một số nguyên dương xác định nào đó.

Cho $\mathcal{P} = C = (\mathbf{Z}_{26})^m$ và cho $\mathcal{K}$ là tất cả các hoán vị có thể có của:

$$\{ 1, 2, \ldots, m \}.$$

Đối với một khoá π (tức là một phép hoán vị nào đó), ta xác định:

$$e_\pi = (x_1, \ldots, x_m) = (x_{\pi(1)}, \ldots, x_{\pi(m)})$$

và $$d_\pi = (x_1, \ldots, x_m) = (y_{\pi^{-1}(1)}, \ldots, y_{\pi^{-1}(m)})$$

trong đó π^{-1} là phép hoán vị ngược của π

2.4. MẬT MÃ HILL

Trong phần này sẽ mô tả một hệ mật thay thế đa biểu khác được gọi là mật mã Hill. Mật mã này do Lester S.Hill đưa ra năm 1929. Giả sử m là một số nguyên dương, đặt $\mathcal{P} = C = (\mathbf{Z}_{26})^m$. Ý tưởng ở đây là lấy m tổ hợp tuyến tính của m ký tự trong một phần tử của bản rõ để tạo ra m ký tự ở một phần tử của bản mã.

Ví dụ nếu $m = 2$ ta có thể viết một phần tử của bản rõ là $x = (x_1, x_2)$ và một phần tử của bản mã là $y = (y_1, y_2)$. Ở đây, y_1 cũng như y_2 đều là một tổ hợp tuyến tính của x_1 và x_2. Chẳng hạn, có thể lấy:

$$\begin{aligned} y_1 &= 11x_1 + 3x_2 \\ y_2 &= 8x_1 + 7x_2 \end{aligned} \qquad (2.5)$$

Tất nhiên có thể viết gọn hơn theo ký hiệu ma trận như sau:

$$(y_1 \quad y_2) = (x_1 \quad x_2)\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} \tag{2.6}$$

Nói chung, có thể lấy một ma trận k kích thước $m \times m$ làm khoá. Nếu một phần tử ở hàng i và cột j của k là $k_{i,j}$ thì có thể viết $k = \left(k_{i,j}\right)$, với $x = (x_1, x_2, ..., x_m) \in \mathcal{P}$ và $k \in \mathcal{K}$, ta tính $y = e_k(x) = (y_1, y_2, ..., y_m)$ như sau :

$$(y_1, y_2, ..., y_m) = (x_1, x_2, ..., x_m)\begin{pmatrix} k_{1,1} & k_{1,2} & ... & k_{1,m} \\ k_{2,1} & k_{2,2} & ... & k_{2,m} \\ ... & ... & ... & ... \\ k_{m,1} & k_{m,2} & ... & k_{m,m} \end{pmatrix} \tag{2.7}$$

Nói cách khác $y = xk$.

Chúng ta nói rằng bản mã nhận được từ bản rõ nhờ phép biến đổi tuyến tính. Ta sẽ xét xem phải thực hiện giải mã như thế nào, tức là làm thế nào để tính x từ y. Bạn đọc đã làm quen với đại số tuyến tính sẽ thấy rằng phải dùng ma trận nghịch đảo k^{-1} để giải mã. Bản mã được giải mã bằng công thức $x = yk^{-1}$.

Sau đây là một số định nghĩa về những khái niệm cần thiết lấy từ đại số tuyến tính. Nếu $\mathrm{A} = \left(x_{i,j}\right)$ là một ma trận cấp $1 \times m$ và $\mathrm{B} = \left(b_{l,k}\right)$ là một ma trận cấp $m \times n$ thì tích ma trận $\mathrm{AB} = \left(c_{l,k}\right)$ được định nghĩa theo công thức :

$$c_{l,k} = \sum_{j=1}^{m} a_{i,j} b_{j,k} \tag{2.8}$$

với $1 \le i \le l$ và $1 \le k \le l$. Tức là các phần tử ở hàng i và cột thứ k của AB được tạo ra bằng cách lấy hàng thứ i của A và cột thứ k của B, sau đó nhân tương ứng các phần tử với nhau và cộng lại. Cần để ý rằng AB là một ma trận cấp $1 \times n$.

Theo định nghĩa này, phép nhân ma trận là kết hợp (tức $(\mathrm{AB})\mathrm{C} = \mathrm{A}(\mathrm{BC})$ nhưng nói chung là không giao hoán (không phải lúc nào $\mathrm{AB} = \mathrm{BA}$, thậm chí đối với ma trận vuông A và B).

Ma trận đơn vị $m \times m$ (ký hiệu là I_m) là ma trận cấp $m \times m$ có các số 1 nằm ở đường chéo chính, và các số 0 ở vị trí còn lại. Như vậy, ma trận đơn vị 2×2 là:

$$\mathrm{I}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.9}$$

I_m được gọi là ma trận đơn vị vì $\mathrm{AI}_m = \mathrm{A}$ với mọi ma trận cấp $1 \times m$ và $\mathrm{I}_m\mathrm{B} = \mathrm{B}$ với mọi ma trận cấp $m \times n$. Ma trận nghịch đảo của ma trận A cấp $m \times m$ (nếu tồn tại) là ma trận A^{-1} sao cho $\mathrm{AA}^{-1} = \mathrm{I}_m$. Không phải mọi ma trận đều có nghịch đảo, nhưng nếu tồn tại thì nó duy nhất.

Với các định nghĩa trên, có thể dễ dàng xây dựng công thức giải mã đã nêu: Vì $y = xk$, ta có thể nhân cả hai vế của đẳng thức với k^{-1} và nhận được:

$$yk^{-1} = (xk)k^{-1} = x\mathrm{I}_m = x \tag{2.10}$$

(Chú ý: sử dụng tính chất kết hợp)

Có thể thấy rằng, ma trận mã hoá ở trên có nghịch đảo trong $\mathbf{Z}_{26}$:

$$\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$$

vì

$$\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}\begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix} = \begin{pmatrix} 11\times 7 + 8\times 23 & 11\times 18 + 8\times 11 \\ 3\times 7 + 7\times 23 & 3\times 18 + 7\times 11 \end{pmatrix}$$
$$= \begin{pmatrix} 261 & 286 \\ 182 & 131 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(Hãy nhớ rằng mọi phép toán số học đều được thực hiện theo modulo 26).

Sau đây là một ví dụ minh hoạ cho việc mã hoá và giải mã trong hệ mật mã Hill.

Ví dụ 2.7:

Giả sử khoá $k = \begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}$

Từ các tính toán trên, ta có:

$$k^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$$

Giả sử cần mã hoá bản rõ "July". Ta có hai phần tử của bản rõ để mã hoá: $(9,20)$ (ứng với Ju) và $(11,24)$ (ứng với ly). Ta tính như sau:

$$(9 \quad 20)\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = \begin{pmatrix} 99+60 & 72+140 \end{pmatrix} = \begin{pmatrix} 3 & 4 \end{pmatrix}$$

$$(11 \quad 24)\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = \begin{pmatrix} 121+72 & 88+168 \end{pmatrix} = \begin{pmatrix} 11 & 22 \end{pmatrix}$$

Bởi vậy, bản mã của **July** là **DELW**. Để giải mã, Bob sẽ tính

$$\begin{pmatrix} 3 & 4 \end{pmatrix}k^{-1} = \begin{pmatrix} 9 & 20 \end{pmatrix} \text{ và } \begin{pmatrix} 11 & 22 \end{pmatrix}k^{-1} = \begin{pmatrix} 11 & 24 \end{pmatrix}$$

Như vậy, Bob đã nhận được bản đúng.

Cho tới lúc này, ta đã chỉ ra rằng có thể thực hiện phép giải mã nếu k có một nghịch đảo. Trên thực tế, để phép giải mã là có thể thực hiện được, điều kiện cần là k phải có nghịch

đảo. (Điều này dễ dàng rút ra từ đại số tuyến tính sơ cấp, tuy nhiên sẽ không chứng minh ở đây). Bởi vậy, ta chỉ quan tâm tới các ma trận k khả nghịch.

Tính khả nghịch của một ma trận vuông phụ thuộc vào giá trị định thức của nó. Để tránh sự tổng quát hoá không cần thiết, ta chỉ giới hạn trong trường hợp 2×2.

Định nghĩa 2.4:

Định thức của ma trận $A=\left(a_{i,j}\right)$ *cấp* 2×2 *là giá trị*

$$\det A = a_{1,1}a_{2,2} - a_{1,2}a_{2,1}$$

Nhận xét: Định thức của một ma trận vuông cấp $m\times m$ có thể được tính theo các phép toán hàng sơ cấp (hãy xem một giáo trình bất kỳ về đại số tuyến tính).

Hai tính chất quan trọng của định thức là:

$$\det \mathrm{I}_m = 1$$

và quy tắc nhân: $\det(\mathrm{AB}) = \det\mathrm{A}\times\det\mathrm{B}$.

Một ma trận thực k là có nghịch đảo khi và chỉ khi định thức của nó khác 0. Tuy nhiên, điều quan trọng cần nhớ là ta đang làm việc trên $\mathbf{Z}_{26}$. *Kết quả tương ứng là ma trận* k *có nghịch đảo theo modulo 26 khi và chỉ khi ƯCLN* $(\det k, 26) = 1$.

Sau đây sẽ chứng minh ngắn gọn kết quả này.

Trước tiên, giả sử rằng $\text{ƯCLN}(\det k, 26) = 1$. Khi đó $\det k$ có nghịch đảo trong $\mathbf{Z}_{26}$. Với $1\le i\le m$, $1\le j\le m$, định nghĩa k_{ij} là ma trận thu được từ k bằng cách loại bỏ hàng thứ i và cột thứ j. Và định nghĩa ma trận k^* có phần tử (i,j) của nó nhận giá trị $(-1)^{i+j}\det k_{ij}$ (k^* được gọi là ma trận bù đại số của k). Khi đó, có thể chứng tỏ rằng:

$$k^{-1} = (\det k)^{-1}k^*$$

Bởi vậy k là khả nghịch.

Ngược lại, k có nghịch đảo k^{-1}. Theo quy tắc nhân của định thức:

$$1 = \det I = \det(kk^{-1}) = \det k \det k^{-1}$$

Bởi vậy $\det k$ có nghịch đảo trong $\mathbf{Z}_{26}$.

Nhận xét: Công thức đối với k^{-1} ở trên không phải là một công thức tính toán có hiệu quả trừ các trường hợp m nhỏ (chẳng hạn m = 2, 3). Với m lớn, phương pháp thích hợp để tính các ma trận nghịch đảo phải dựa vào các phép toán hàng sơ cấp.

Trong trường hợp 2×2, ta có công thức sau:

Định lý 2.3:

Giả sử $\mathrm{A}=(a_{ij})$ là một ma trận cấp 2×2 trên $\mathbf{Z}_{26}$ sao cho:

$\det A = (a_{1,1}a_{2,2} - a_{1,2}a_{2,1})$ có nghịch đảo. Khi đó:

$$A^{-1} = (\det A)^{-1}\begin{pmatrix} a_{2,2} & -a_{1,2} \\ -a_{2,1} & a_{1,1} \end{pmatrix}$$

Trở lại ví dụ đã xét ở trên. Trước hết ta có:

$$\det\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix} = 11\times 7 - 8\times 3 \bmod 2 = 77 - 24 \bmod 2 = 53 \bmod 2 = 1$$

Vì $1^{-1} \bmod 26 = 1$ nên ma trận nghịch đảo là:

$$\begin{pmatrix} 11 & 8 \\ 3 & 7 \end{pmatrix}^{-1} = \begin{pmatrix} 7 & 18 \\ 23 & 11 \end{pmatrix}$$

Đây chính là ma trận đã có ở trên. (Chú ý: $-8 = 18; -3 = 23$)

Bây giờ ta sẽ mô tả chính xác mật mã Hill trên $\mathbf{Z}_{26}$ (Hình 2.3).

Cho m là một số nguyên dương cố định. Cho $\mathcal{P} = C = (\mathbf{Z}_{26})^m$ và cho $\mathcal{K}$ = { các ma trận khả nghịch cấp $m \times m$ trên $\mathbf{Z}_{26}$ }

Với một khoá $k \in \mathcal{K}$ ta xác định:

$$e_k(x) = xk$$

và $$d_k(y) = yk^{-1}$$

Tất cả các phép toán được thực hiện trong $\mathbf{Z}_{26}$

Hình 2.3. Mật mã Hill

2.5. HỆ MẬT XÂY DỰNG TRÊN CÁC CẤP SỐ NHÂN XYCLIC CỦA VÀNH ĐA THỨC

Trong phần này ta xét một ứng dụng của nhóm nhân xyclic trên vành đa thức $\mathbf{Z}_x[x]/x^n + 1$ với $n = 2^k$. Đây là một trường hợp đặc biệt không được xem xét tới khi xây dựng các mã khống chế sai.Tuy nhiên,trường hợp này lại có những ứng dụng khá lý thú trong mật mã [4].

2.5.1. Nhóm nhân của vành

Bổ đề 2.1:

Trong vành $\mathbf{Z}_x[x]/x^n + 1$ với $n = 2^k$, tập các đa thức có trọng số lẻ sẽ tạo nên một nhóm nhân các đa thức theo modulo $x^n + 1$.

Chứng minh: Vì $n = 2^k$ nên: $(x^n + 1) = (1 + x)^n$. Do đó, mọi đa thức $a(x)$ có trọng số lẻ đều thoả mãn điều kiện:

$$\left(a(x),(1+x)^n\right)=1 \tag{2.11}$$

Các đa thức này sẽ tạo nên một nhóm nhân G có luỹ đẳng $e(x)=1$ và có cấp bằng: $|G|=2^{n-1}$.

Bổ đề 2.2:

Mọi phần tử trong nhóm nhân G có cấp là 2^k hoặc có cấp là ước của 2^k.

Chứng minh: Đây là một trường hợp riêng của định lý ở phần 2.4.10. Ta có thể chứng minh bằng qui nạp:

$k=1$: vành này chứa nhóm nhân cấp 2 là nhóm nhân xyclic đơn vị I.

$k=i$: Giả sử $A=\{a(x),a^2(x),a^3(x),...,a^n(x)\}$ là một nhóm nhân xyclic cấp n trong vành ($n=2^i$).

$k=i+1$: Bình phương các phần tử của A ta có nhóm nhân xyclic sau:

$$A^2=\{a^2(x),a^4(x),a^6(x),...,a^{2n}(x)\}$$

Nhóm nhân xyclic này hiển nhiên là nhóm con của nhóm nhân xyclic cấp $2.2^i=2^{i+1}$ có phần tử sinh là một trong các căn bậc hai của $a(x)$.

Gọi Q là tập các thặng dư bậc hai trong G. Ta có bổ đề sau:

Bổ đề 2.3:

Số các thặng dư bậc hai trong nhóm nhân G của vành được xác định theo biểu thức sau :

$$|Q|=2^{2^{k-1}-1} \tag{2.12}$$

Chứng minh: Xét $f(x)\in Q$. Giả sử căn bậc hai của $f(x)$ là $g(x)$, tức là:

$$g^2(x)=f(x)\bmod x^n+1$$

Nếu $g(x)=\sum g_i x^i$ thì $f(x)=\sum g_i x^{2i}$.

Tức là $f(x)$ (có trọng số lẻ) chỉ gồm một số lẻ các đơn thức có mũ chẵn.

Số lượng các đa thức này bằng:

$$|Q|=C_{n/2}^1+C_{n/2}^3+...+C_{n/2}^{(n/2)-1}$$

2.5.2. Các phần tử cấp n và các nhóm nhân xyclic cấp n

Xét $a(x)\in G$. $a(x)=\sum a_i x^i$. Ta có bổ đề sau:

Bổ đề 2.4:

Đa thức $a(x)$ là phần tử cấp n khi nó có chứa một số lẻ các đơn thức có mũ lẻ có cấp n và một số chẵn các đơn thức có mũ chẵn có cấp là ước của n. Số các đa thức cấp n bằng 2^{n-2}.

Chứng minh: Vì $a(x) \in G$ nên nó có trọng số lẻ. Số lượng các đơn thức có cấp n là $(n/2)$ và số lượng các đơn thức còn lại là $(n/2)$. Như vậy, số các đa thức $a(x)$ có cấp n bằng:

$$\left(\sum_i C_{n/2}^{2i-1}\right)\left(\sum_j C_{n/2}^{2j}\right) = 2^{(n/2)-1}2^{(n/2)-1} = 2^{n-2}$$

Ví dụ 2.8:

Với trường hợp $n = 8$, có tất cả $2^6 = 64$ các phần tử cấp n.

Ta có thể sử dụng các phần tử này để xây dựng các nhóm nhân xyclic cấp n.

$$A_i = \{a_i(x), a_i^2(x), a_i^3(x), ..., a_i^n(x) = a_i^0(x) = 1\}$$

Có tất cả 2^{n-2} các nhóm nhân xyclic cấp n và nhóm nhân I cũng thuộc vào lớp các nhóm nhân này. Ta gọi nó là nhóm nhân xyclic đơn vị.

2.5.3. Hệ mật xây dựng trên các cấp số nhân xyclic

2.5.3.1. Các cấp số nhân xyclic cấp n

Nếu ta nhân các phần tử của một nhóm nhân xyclic cấp n với một phần tử bất kỳ trong nhóm nhóm nhân G của vành đa thức ta sẽ thu được một cấp số nhân xyclic có công bội là phần tử sinh của nhóm nhân và có số hạng ban đầu là đa thức đem nhân.

Bổ đề 2.5:

Số các cấp số nhân xyclic cấp n xây dựng được trong G xác định theo biểu thức sau:

$$N = 2^{2^k-1}.2^{2^k-2} \tag{2.13}$$

Ví dụ 2.9:

$n = 8$ $\quad N = 2^{8-1}.2^{8-2} = 2^{13} = 8.192$

$n = 16$ $\quad N = 2^{16-1}.2^{16-2} = 2^{29} = 536.870.912$

$n = 32$ $\quad N = 2^{32-1}.2^{32-2} = 2^{61}$

$n = 64$ $\quad N = 2^{64-1}.2^{64-2} = 2^{125}$

$n = 128$ $\quad N = 2^{128-1}.2^{128-2} = 2^{253}$

2.5.3.2. Hệ mật xây dựng trên các cấp số nhân xyclic

Mỗi cấp số nhân xyclic cấp n có thể coi là một phép biến đổi tuyến tính của vector mã ban đầu (được coi là nhóm nhân xyclic đơn vị I).

Gọi α là phần tử sinh của một nhóm nhân xyclic cấp n. Ta có bổ đề sau:

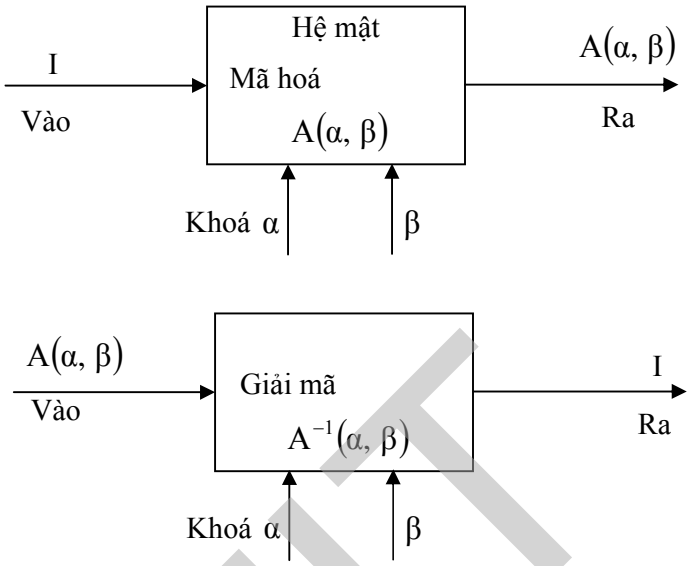
Bổ đề 2.6:

Tổng các số hạng của một cấp số nhân xyclic cấp n có công bội α và số hạng đầu β được xác định theo biểu thức sau:

$$S_n = \beta\left[\prod_{i=0}^{k-1}\left(1+\alpha^{2^i}\right)\right] \tag{2.14}$$

Hiển nhiên là $S_n \neq 0$.

Hệ mật xây dựng trên các cấp số nhân này có thể được mô tả theo sơ đồ khối sau:



Hình 2.4.

Mỗi phép biến đổi (mã hoá) A có thể được đặc trưng bởi một ma trận vuông cấp n có dạng sau:

$$A = \begin{pmatrix} \beta.\alpha \\ \beta.\alpha^2 \\ \vdots \\ \beta.\alpha^0 \end{pmatrix}$$

A là một ma trận không suy biến và bởi vậy, luôn tồn tại A^{-1} thoả mãn:

$$A.A^{-1} = I$$

Tập các phép biến đổi này là một tập kín đối với phép tính (nhân ma trận) và tạo nên một nhóm nhân có phần tử đơn vị là phép biến đổi đồng nhất (ma trận đơn vị I).

Nhóm nhân trong vành các ma trận vuông này là nhóm tuyến tính đầy đủ và được ký hiệu là $GL\left(n, GF(2)\right)$.

Thuật toán mã hoá khá đơn giản, chỉ dựa trên phép toán nhân và bình phương một đa thức $a(x) \in G$ theo modulo $(x^n + 1)$ ($a(x)$ có cấp n) với một đa thức $b(x)$ bất kỳ $\in G$.

2.5.3.3. Vấn đề giải mã

Để giải mã ta phải tìm phép biến đổi ngược A^{-1} là ma trận nghịch đảo của ma trận A. Tuy nhiên ta có thể dễ dàng thực hiện giải mã dựa trên bổ đề sau:

Bổ đề 2.7:

Ma trận A có cấp (order) hoặc là n , hoặc là ước của n . Tức là ta luôn có:

$$A^n = I$$

Hay $$\underbrace{\left(\left((A^2)^2\right)^2 ...\right)^2}_{k \text{ lần}}$$

Ở đây, A được xem là phần tử sinh của một nhóm nhân xyclic có cấp bằng n hoặc bằng ước của n .

Ví dụ 2.10: $n = 8$

$$A = \{(012),(024),(01356),(4),(456),(046),(12457),(0)\}$$

Ma trận tương ứng:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A^2 = \{(014),(2),(236),(4),(045),(6),(267),(0)\}$$

$$A^3 = \{(124),(024),(01235),(4),(056),(046),(14567),(0)\}$$

$$A^4 = I = \{(1),(2),(3),(4),(5),(6),(7),(0)\}$$

Chú ý: Ở đây ta biểu diễn các đa thức qua các số mũ của các thành phần khác không. Ví dụ: $(01235) = 1 + x + x^2 + x^3 + x^5$.

Vào	Mã hoá	Ra	Vào	Giải mã	Ra
I $\rightarrow$	A $\rightarrow$	A	A $\rightarrow$	$(A^2)^2$ =	I

Ví dụ 2.11:

Xét cấp số nhân có công bội (023) với số hạng đầu (023) (012) = (015).

$$B = \{(015),(12457),(03467),(456),(145),(01356),(02347),(012)\}$$

$$B^2 = \{(124),(136),(346),(035),(056),(257),(027),(147)\}$$

$$B^3 = \{(02567),(047),(167),(23567),(12346),(034),(235),(12367)\}$$

$$B^4 = \{(02456),(13567),(02467),(01357),(01246),(12357),(02346),(13457)\}$$

$$B^5 = \{(347),(12345),(01245),(146),(037),(01567),(01456),(025)\}$$

$$B^6 = \{(245),(123),(467),(345),(016),(567),(023),(017)\}$$

$$B^7 = \{(24567),(236),(127),(01347),(01236),(267),(356),(03457)\} = B^{-1}$$

$$B^8 = \{(1),(2),(3),(4),(5),(6),(7),(0)\}$$

$$I = \left(\left(B^2\right)^2\right)^2$$

Thuật toán giải mã chỉ là một thuật toán lặp của thuật toán mã hoá. Số lần lặp tối đa là k .

2.5.3.4. Các ma trận luân hoàn

Khi sử dụng cấp số nhân có công bội x và có số hạng đầu là một đa thức $a(x) \in G$ ta sẽ có một lớp các biến đổi đặc biệt, được đặc trưng bởi một loại ma trận đặc biệt, được gọi là ma trận luân hoàn.

Định nghĩa 2.5:

Ma trận vuông A $n \times n$ trên trường F được gọi là ma trận luân hoàn nếu nó có dạng sau:

$$A = \begin{pmatrix} a(x) \\ xa(x) \\ \vdots \\ x^{n-1}a(x) \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & \dots & a_{n-1} \\ a_{n-1} & a_0 & \dots & a_{n-2} \\ \vdots & \vdots & & \vdots \\ a_1 & a_2 & \dots & a_0 \end{pmatrix}; \qquad a \in F$$

Bổ đề 2.8:

Đại số các ma trận luân hoàn cấp n trên trường F đẳng cấu với đại số $F[x]/(x^n - 1)$ đối với phép ánh xạ các ma trận luân hoàn thành các đa thức dạng:

$$a(x) = \sum_{i=0}^{n-1} a_i x^i$$

Bổ đề 2.9:

Tổng và tích của hai ma trận luân hoàn là một ma trận luân hoàn.

Ta có: A.B = C

Trong đó: $c(x) = a(x)b(x) \bmod (x^n - 1)$

Bổ đề 2.10:

Ma trận luân hoàn A là khả nghịch khi và chỉ khi đa thức $a(x)$ là nguyên tố cùng nhau với $(x^n - 1)$. Ma trận nghịch đảo B nếu tồn tại sẽ tương ứng với $b(x)$ thoả mãn điều kiện:

$$a(x)b(x) \equiv 1 \bmod (x^{2^k} - 1)$$

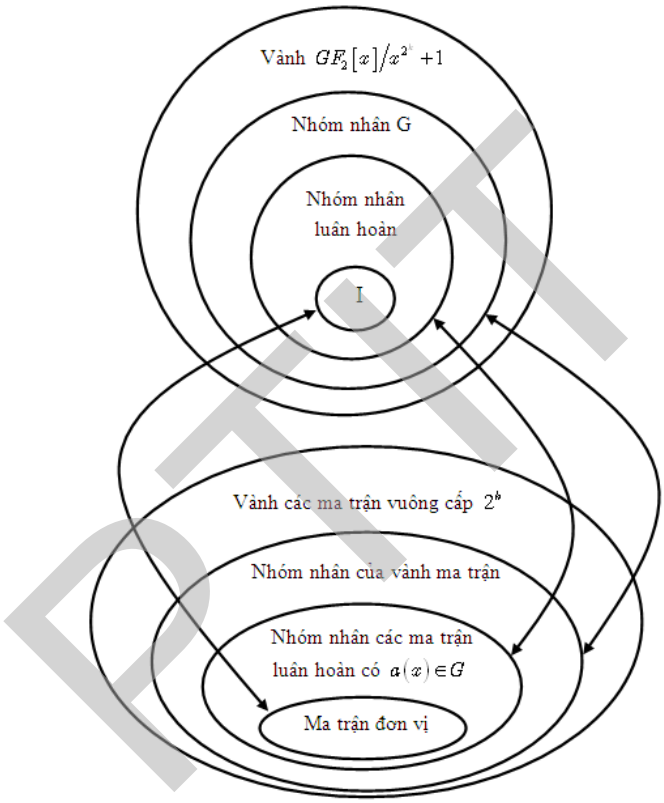
Trong trường hợp vành $GF_2[x]/(x^n+1)$ và $a(x) \in G$, ta luôn có:

$$\left(a(x),(x^{2^k}+1)\right)=\left(a(x),(x+1)^{2^k}\right)=1$$

Bổ đề 2.11:

Tập các ma trận luân hoàn A ứng với $a(x) \in G$ sẽ tạo nên một nhóm con nhân Abel trong nhóm nhân của vành các ma trận vuông. Trong nhóm này tồn tại các nhóm con là các nhóm nhân xyclic có cấp bằng n hoặc ước của n.

Mối quan hệ giữa nhóm nhân của vành đa thức và nhóm nhân của vành các ma trận vuông được mô tả trên hình sau (Hình 2.5).



Hình 2.5. Quan hệ giữa vành đa thức và vành ma trận

Bổ đề 2.12:

Cấp của ma trận luân hoàn A bằng cấp của đa thức $a(x)$ tương ứng của nó.

Khi $ord(a(x))=2$ thì ma trận luân hoàn A tương ứng là một ma trận tự nghịch đảo.

Bổ đề 2.13:

Số các ma trận luân hoàn dùng để lập mã bằng số các phần tử của nhóm nhân trong vành đa thức.

Trong trường hợp ma trận luân hoàn, thuật toán mã hoá chỉ là một phép cộng với n bước dịch vòng.

Thuật toán giải mã bao gồm một phép tính nghịch đảo của một đa thức theo modulo $(x^n + 1)$ và n bước dịch vòng tương ứng của phần tử nghịch đảo này.

Ví dụ 2.12: $a(x) = 1 + x + x^2$

$$A = \{(012),(123),(234),(345),(456),(567),(067),(017)\}$$

$$A^2 = \{(124),(135),(246),(357),(046),(157),(026),(137)\}$$

$$A^3 = \{(01356),(12467),(02357),(01346),(12457),(02356),(13467),(02457)\}$$

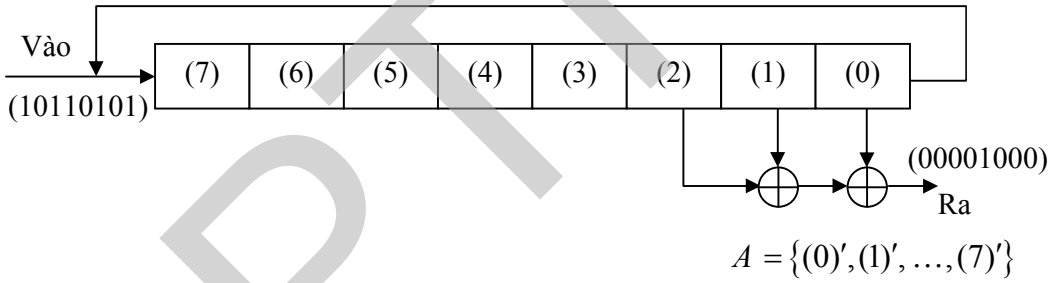
$$A^4 = \{(4),(5),(6),(7),(0),(1),(2),(3)\}$$

$$A^5 = \{(456),(567),(067),(017),(012),(123),(234),(345)\}$$

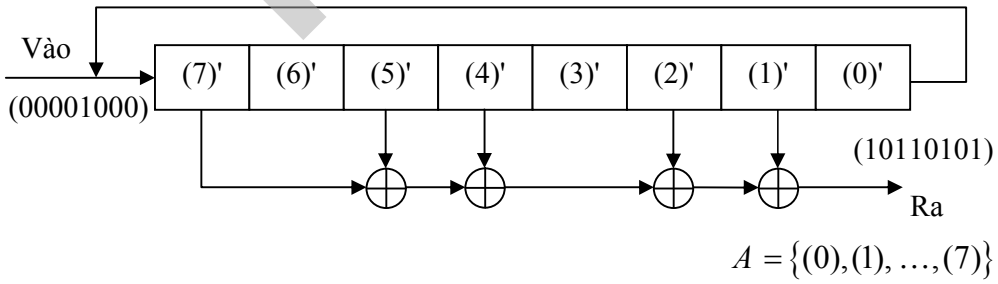
$$A^6 = \{(046),(157),(026),(137),(024),(135),(246),(357)\}$$

$$A^7 = \{(12457),(02356),(13467),(02457),(01356)$$
$$(12467),(02357),(01346)\} = A^{-1}$$

$$A^8 = \{(1),(2),(3),(4),(5),(6),(7),(0)\} = I$$



Hình 2.6. Sơ đồ thiết bị mã hoá



Hình 2.7. Sơ đồ thiết bị giải mã

$$a^{-1}(x) = x + x^2 + x^4 + x^5 + x^7$$

Ta có:

$$AA^{-1} = \begin{pmatrix} 1&1&1&0&0&0&0&0\\ 0&1&1&1&0&0&0&0\\ 0&0&1&1&1&0&0&0\\ 0&0&0&1&1&1&0&0\\ 0&0&0&0&1&1&1&0\\ 0&0&0&0&0&1&1&1\\ 1&0&0&0&0&0&1&1\\ 1&1&0&0&0&0&0&1 \end{pmatrix} \times \begin{pmatrix} 0&1&1&0&1&1&0&1\\ 1&0&1&1&0&1&1&0\\ 0&1&0&1&1&0&1&1\\ 1&0&1&0&1&1&0&1\\ 1&1&0&1&0&1&1&0\\ 0&1&1&0&1&0&1&1\\ 1&0&1&1&0&1&0&1\\ 1&1&0&1&1&0&1&0 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} = I$$

2.6. CÁC HỆ MẬT MÃ TÍCH

Một phát minh khác do Shannon đưa ra trong bài báo của mình năm 1949 là ý tưởng kết hợp các hệ mật bằng cách tạo tích của chúng. Ý tưởng này có tầm quan trọng to lớn trong việc thiết kế các hệ mật hiện nay (chẳng hạn, chuẩn mã dữ liệu - DES).

Để đơn giản, trong phần này chỉ hạn chế xét các hệ mật trong đó $C = \mathcal{P}$: các hệ mật loại này được gọi là tự đồng cấu. Giả sử $S_1 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1, \mathcal{E}_1, \mathcal{D}_1)$ và $S_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_2, \mathcal{E}_2, \mathcal{D}_2)$ là hai hệ mật tự đồng cấu có cùng các không gian bản mã và rõ. Khi đó, tích của S_1 và S_2 (kí hiệu là $S_1 \times S_2$) được xác định là hệ mật sau:

$$(\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2, \mathcal{E}, \mathcal{D})$$

Khoá của hệ mật tích có dạng $k = (k_1, k_2)$ trong đó $k_1 \in \mathcal{K}_1$ và $k_2 \in \mathcal{K}_2$. Các quy tắc mã và giải mã của hệ mật tích được xác định như sau: Với mỗi $k = (k_1, k_2)$, ta có một quy tắc mã e_k xác định theo công thức:

$$e_{(k_1,k_2)}(x) = e_{k_2}\left(e_{k_1}(x)\right)$$

và quy tắc giải mã:

$$d_{(k_1,k_2)}(y) = d_{k1}\left(d_{k_2}(y)\right)$$

Nghĩa là, trước tiên ta mã hoá x bằng e_{k_1} rồi mã lại mã hóa bản kết quả bằng e_{k_2}. Quá trình giải mã tương tự nhưng thực hiện theo thứ tự ngược lại:

$$d_{(k_1,k_2)}\left(e_{(k_1,k_2)}(x)\right) = d_{(k_1,k_2)}\left(e_{k_2}\left(e_{k_1}(x)\right)\right) = d_{k_1}\left(d_{k_2}\left(e_{k_2}\left(e_{k_1}(x)\right)\right)\right) = d_{k_1}\left(e_{k_1}(x)\right) = x$$

Ta biết rằng, các hệ mật đều có các phân bố xác suất ứng với các không gian khoá của chúng. Bởi vậy, cần phải xác định phân bố xác suất cho không gian khoá K của hệ mật tích. Hiển nhiên ta có thể viết:

$$p_K(k_1, k_2) = p_{K_1}(k_1) \times p_{K_2}(k_2)$$

Nói một cách khác, ta chọn k_1 có phân bố $p_{K_1}(k_1)$ rồi chọn một cách độc lập k_2 có phân bố $p_{K_2}(k_2)$.

Sau đây là một ví dụ đơn giản để minh hoạ khái niệm hệ mật tích. Giả sử định nghĩa hệ mật mã nhân như trong Hình 2.8 sau.

Giả sử $P = C = \mathbf{Z}_{26}$ và giả sử:

$$k = \{\, a, \mathbf{Z}_{26} : UCLN(a, 26) = 1 \,\}$$

Với $a \in K$, ta xác định: $e_a(x) = ax \mod 26$

và $d_a(y) = a^{-1}y \mod 26$

$(x, y) \in \mathbf{Z}_{26}$

Hình 2.8. Mật mã tích

Cho M là một hệ mã nhân (với các khoá được chọn đồng xác suất) và S là MDV (với các khoá chọn đồng xác suất). Khi đó dễ dàng thấy rằng $M \times S$ chính là hệ mã Affine (cùng với các khoá được chọn đồng xác suất). Tuy nhiên, việc chứng tỏ $S \times M$ cũng là hệ mã Affine khó hơn một chút (cũng với các khóa đồng xác suất).

Ta sẽ chứng minh các khẳng định này. Một khoá dịch vòng là phần tử $k \in \mathbf{Z}_{26}$ và quy tắc mã hóa tương ứng là $e_k(x) = x + k \mod 26$. Còn khoá trong hệ mã nhân là phần từ $a \in \mathbf{Z}_{26}$ sao cho ƯCLN$(a, 26) = 1$. Quy tắc mã tương ứng là $e_a(x) = a \mod 26$. Bởi vậy, một khoá trong mã tích $M \times S$ có dạng (a, k), trong đó

$$e_{(a,k)} = ax + k \mod 26$$

Đây chính là định nghĩa về khoá trong hệ mã Affine. Hơn nữa, xác suất của một khoá trong hệ mã Affine là: $1/312 = (1/12)(1/26)$. Đó là tích của xác suất tương ứng của các khoá a và k. Bởi vậy $M \times S$ là hệ mã Affine.

Bây giờ ta sẽ xét $S \times M$. Một khoá này trong hệ mã này có dạng (k, a), trong đó:

$$e_{(k,a)}(x) = a(x + k) = ax + ak \mod 26$$

Như vậy, khoá (k, a) của mã tích $S \times M$ đồng nhất với khoá (a, ak) của hệ mã Affine. Vấn đề còn lại là phải chứng tỏ rằng mỗi khoá của mã Affine xuất hiện với cùng xác suất 1/312 như trong mã tích $S \times M$. Nhận thấy rằng $ak = k_1$ khi và chỉ khi $k = a^{-1}k_1$, (hãy nhớ lại rằng ƯCLN$(a, 26) = 1$, bởi vậy a có phần tử nghịch đảo). Nói cách khác, khoá (a, k_1) của hệ mã Affine tương đương với khoá $(a^{-1}k_1, a)$ của mã tích $S \times M$. Bởi vậy, ta có một song ánh giữa hai không gian khoá. Vì mỗi khoá là đồng xác suất nên có thể thấy rằng $S \times M$ thực sự là mã Affine.

Ta chứng minh rằng $M \times S = S \times M$. Bởi vậy, hai hệ mật là giao hoán. Tuy nhiên, không phải mọi cặp hệ mật đều giao hoán; có thể tìm ta được các cặp phản ví dụ. Mặt khác ta thấy rằng phép tích luôn kết hợp:

$$(S_1 \times S_2) \times S_3 = S_1 \times (S_2 \times S_3)$$

Nếu lấy tích của một hệ mật tự đồng cấu với chính nó thì ta thu được hệ mật $S \times S$ (kí hiệu là S^2). Nếu lấy tích n lần thì hệ mật kết quả là S^n. Ta gọi S^n là hệ mật lặp.

Một hệ mật S được gọi là luỹ đẳng nếu $S^2 = S$. Có nhiều hệ mật đã nghiên cứu trong chương 1 là hê mật luỹ đẳng. Chẳng hạn các hệ MDV, MTT, Affine, Hill, Vigenère và hoán vị đều là luỹ đẳng. Hiển nhiên là nếu hệ mật S là luỹ đẳng thì không nên sử dụng hệ mật tích S^2 vì nó yêu cầu lượng khoá cực lớn mà không có độ bảo mật cao hơn.

Nếu một hệ mật không phải là luỹ đẳng thì có thể làm tăng độ mật bằng cách lặp nhiều lần. Ý tưởng này đã được dùng trong chuẩn mã dữ liệu (DES). Trong DES dùng 16 phép lặp, tất nhiên hệ mật ban đầu phải là hệ mật không luỹ đẳng. Một phương pháp có thể xây dựng các hệ mật không luỹ đẳng đơn giản là lấy tích của hai hệ mật đơn giản khác nhau.

Nhận xét:

Có thể dễ dàng chứng tỏ rằng, nếu cả hai hệ mật S_1 và S_2 là luỹ đẳng và giao hoán thì S_1 và S_2 cũng là luỹ đẳng. Điều này rút ra từ các phép toán đại số sau:

$$\begin{aligned}(S_1 \times S_2) \times (S_1 \times S_2) &= S_1 \times (S_2 \times S_1) \times S_2 \\ &= S_1 (S_1 \times S_2) \times S_2 \\ &= (S_1 \times S_1) \times (S_2 \times S_2) \\ &= S_1 \times S_2\end{aligned}$$

(Chú ý: Dùng tính chất kết hợp trong chứng minh trên).

Bởi vậy, nếu cả S_1 và S_2 đều là luỹ đẳng và ta muốn $S_1 \times S_2$ là không luỹ đẳng thì điều kiện cần là S_1 và S_2 không giao hoán.

Rất may mắn là nhiều hệ mật đơn giản thoả mãn điều kiện trên. Kỹ thuật thường được sử dụng trong thực tế là lấy tích các hệ mã kiểu thay thế và các hệ mã kiểu hoán vị.

2.7. CÁC HỆ MÃ DÒNG

2.7.1. Sơ đồ chức năng của hệ mật mã dòng

Trong các hệ mật nghiên cứu ở trên, các phần tử liên tiếp của bản rõ đều được mã hoá bằng cùng một khoá k. Tức xâu bản mã y nhận được có dạng:

$$y = y_1 y_2 \ldots = e_k(x_1) e_k(x_2) \ldots$$

Các hệ mật thuộc dạng này thường được gọi là các mã khối. Một quan điểm sử dụng khác là mật mã dòng. Ý tưởng cơ bản ở đây là tạo ra một dòng khoá $z = z_1 z_2 \ldots$ và dùng nó để mã hoá một xâu bản rõ $x = x_1 x_2 \ldots$ theo quy tắc:

$$y = y_1 y_2 \ldots = e_{z_1}(x_1) e_{z_2}(x_2) \ldots$$

Sơ đồ mã hóa và giải mã của hệ mật mã dòng mô tả trong Hình 2.9

a) Mã hóa *b) Giải mã*

Hình 2.9. Sơ đồ chức năng hệ mật mã dòng

Mã dòng hoạt động như sau: Giả sử $k \in \mathcal{K}$ là khoá, và $x = x_1 x_2 ...$ là xâu bản rõ. Hàm f_i được dùng để tạo z_i (z_i là phần tử thứ i của dòng khoá), trong đó f_i là một hàm của khoá k và $i-1$ là ký tự đầu tiên của bản rõ:

$$z_i = f_i(k, x_1, ..., x_{i-1})$$

Phần tử z_i của dòng khoá được dùng để mã x_i tạo ra $y_i = e_{z_i}(x_i)$. Bởi vậy, để mã hoá xâu bản rõ $x = x_1 x_2 ...$ ta phải tính liên tiếp $z_1, y_1, z_2, y_2, ...$

Việc giải mã xâu bản mã $y_1 y_2 ...$ có thể được thực hiện bằng cách tính liên tiếp $z_1, x_1, z_2, x_2, ...$

Sau đây là định nghĩa dưới dạng toán học:

Định nghĩa 3.6:

Mật mã dòng là một bộ $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{F}, \mathcal{E}, \mathcal{D})$ *thoả mãn các điều kiện sau:*

(1) $\mathcal{P}$ *là một tập hữu hạn các bản rõ có thể.*

(2) $\mathcal{C}$ *là tập hữu hạn các bản mã có thể.*

(3) $\mathcal{K}$ là *tập hữu hạn các khoá có thể (không gian khoá)*

(4) $\mathcal{L}$ *là tập hữu hạn các bộ chữ của dòng khoá.*

(5) $\mathcal{F} = (f_1 f_2 ...)$ *là bộ tạo dòng khoá. Với* $i \geq 1$

$$f_i : \mathcal{K} \times \mathcal{P}^{i-1} \to \mathcal{L}$$

(6) Với mỗi $z \in \mathcal{L}$ *có một quy tắc mã* $e_z \in \mathcal{E}$ *và một quy tắc giải mã tương ứng* $d_z \in \mathcal{D}$. $e_z : \mathcal{P} \to \mathcal{C}$ *và* $d_z : \mathcal{C} \to \mathcal{P}$ *là các hàm thoả mãn* $d_z(e_z(x)) = x$ *với mọi bản rõ* $x \in \mathcal{P}$.

Ta có thể coi mã khối là một trường hợp đặc biệt của mã dòng, trong đó dùng khoá không đổi: $z_i = k$ với mọi $i \geq 1$.

Nhận xét:

- Để hệ thống an toàn, dãy bit khóa ngẫu nhiên và $|k| \geq |m|$. (Dãy ngẫu nhiên được lấy là kết quả của quá trình tung đồng xu: $p(0) = p(1) = 0,5$).

- Việc tạo dãy ngẫu nhiên rất tốn kém và việc lưu trữ không hiệu quả, do vậy ta phải sử dụng dãy giả ngẫu nhiên, các dãy này có tính tiền định và được xây dựng từ các bit mầm.

2.7.2. Tạo dãy giả ngẫu nhiên (M-dãy)

2.7.2.1. Tạo dãy giả ngẫu nhiên theo đa thức nguyên thủy

Định nghĩa 2.6: Đa thức nguyên thủy

Đa thức bất khả quy bậc m được gọi là đa thức nguyên thủy nếu nó là ước của x^n+1 với $n=2^m-1$ nhưng không là ước của $x^\ell+1$ với $\ell<n$.

(chú ý: đa thức bất khả quy là đa thức chia hết cho 1 và chính nó, tương đương số nguyên tố)

Ví dụ 2.13:

+ $m=3\rightarrow n=7$, ta có: $x^7+1=(1+x)(1+x+x^3)(1+x^2+x^3)$.

Trên vành đa thức x^7+1 có hai đa thức: $f_1(x)=(1+x+x^3)$ và $f_2(x)=(1+x^2+x^3)$ là hai đa thức nguyên thủy vì nó không là ước của $x^\ell+1$ với $\ell\le 6$.

+ $m=4\rightarrow n=15$, ta có

$$x^{15}+1=(1+x)(1+x+x^2)(1+x+x^4)(1+x^3+x^4)(1+x+x^2+x^3+x^4)$$

Với trường hợp này chỉ có hai đa thức $1+x+x^4$ và $1+x^3+x^4$ là nguyên thủy, còn đa thức $1+x+x^2+x^3+x^4$ không phải nguyên thủy vì nó là ước của x^5+1.

$$x^5+1=(1+x)(1+x+x^2+x^3+x^4)$$

Bổ đề 2.14:

Dãy giả ngẫu nhiên (M-dãy) được tạo từ phương trình đồng dư sau đây:

$$\boxed{a(x)\equiv b(x)x^i \bmod g(x)};\ i=1,2,...,2^n-1 \qquad (2.15)$$

Với: $g(x)$ là đa thức nguyên thủy bậc m

$b(x)$ là đa thức mầm ứng với m bit, được chọn ngẫu nhiên thỏa mãn $\deg b(x)\le m-1$

Ví dụ 2.14:

Với $m=4; g(x)=1+x+x^4$, M-dãy được tạo như sau:

$$a(x)\equiv b(x)x^i \bmod 1+x+x^4$$

Coi $1+x+x^4=0\rightarrow x^4=1+x$. Giả sử $b(x)=1+x\leftrightarrow 1100$

Trạng thái của M-dãy này được mô tả trong Bảng 2.1

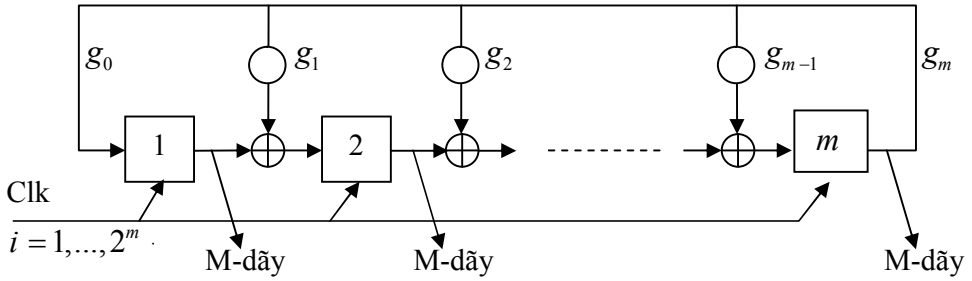
Bảng 2.1. Trạng thái của M-dãy

TT	$a(x)$	$\vec{a}$
0	$1+x$	1 1 0 0
1	$x+x^2$	0 1 1 0
2	x^2+x^3	0 0 1 1
3	$1+x+x^3$	1 1 0 1
4	$1+x^2$	1 0 1 0
5	$x+x^3$	0 1 0 1
6	$1+x+x^2$	1 1 1 0
7	$x+x^2+x^3$	0 1 1 1
8	$1+x+x^2+x^3$	1 1 1 1
9	$1+x^2+x^3$	1 0 1 1
10	$1+x^3$	1 0 0 1
11	1	1 0 0 0
12	x	0 1 0 0
13	x^2	0 0 1 0
14	x^3	0 0 0 1
15	$1+x$	1 1 0 0

Nhận xét:

- Khi lấy bất kỳ một cột nào trong 4 cột của $\vec{a}$ ta sẽ được một M-dãy.
- Chu kỳ của dãy: $t = 2^m - 1$ với trường hợp này $m = 4 \rightarrow t = 15$.
- Số con "1" trong dãy: $N_1 = 2^m$, với $m = 4 \rightarrow N_1 = 8$.
- Số con "0" trong dãy: $N_0 = 2^m - 1$, với $m = 4 \rightarrow N_0 = 7$.
- Khi $m \rightarrow \infty$ ta có: $\lim_{m\to\infty} p(0) = \lim_{m\to\infty} p(1) = 1/2$

Cấu trúc tổng quát mạch điện phần cứng M-dãy được thực hiện bằng các thanh ghi dịch hồi tiếp tuyến tính (LFSR) và có dạng như Hình 2.10.

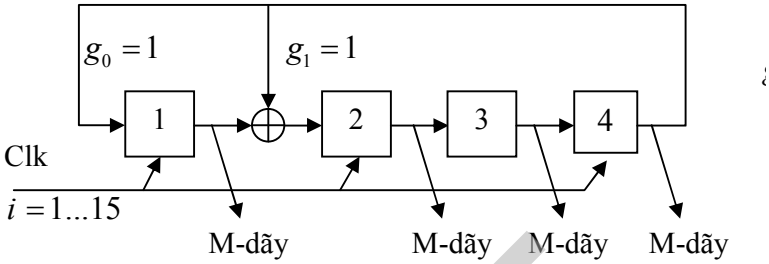


Hình 2.10. Mạch điện thực hiện M-dãy

Trong sơ đồ Hình 2.10 các g_i nhận giá trị "0" hoặc "1" là các hệ số của đa thức nguyên thủy $g(x) = g_0 + g_1x + g_2x^2 + ... + g_mx^m$. Riêng $g_0 = g_m = 1$ (luôn bằng "1"). Nếu $g_i = 1$ thì mạch điện sẽ nối tắt, còn $g_i = 0$ thì sẽ hở mạch.

Ví dụ 2.15:

Giả sử $g(x) = 1 + x + x^4, (m = 4)$, ta thấy $g_0 = g_1 = g_4 = 1; g_2 = g_3 = 0$, mạch điện tạo M-dãy như sau:



Hình 2.11. Mạch điện tạo M-dãy với $g(x) = 1 + x + x^4$

Bảng 2.2. Trạng thái hoạt động của các ô nhớ

Nhịp i	**Trạng thái các ô nhớ**			
	1	**2**	**3**	**4**
0	1	1	0	0
1	0	1	1	1
2	0	0	1	1
3	1	1	0	1
4	1	0	1	0
5	0	1	0	1
6	1	1	1	0
7	0	1	1	1
8	1	1	1	1
9	1	0	1	1
10	1	0	0	1
11	1	0	0	0
12	0	1	0	0
13	0	0	1	0
14	0	0	0	1
15	1	1	0	0

Trong Bảng 2.2 thì trạng thái các ô nhớ tại dòng đầu tiên tương ứng với các bit mầm (đa thức $b(x) = 1 + x$).

2.7.2.2. Tạo dãy giả ngẫu nhiên trên vành đa thức có hai lớp kề xyclic

Định nghĩa 2.7: Vành đa thức có hai lớp kề xyclic là vành có phân tích như sau:

$$x^n + 1 = (1 + x)\sum_{i=0}^{n-1} x^i$$

Trong đó: $(1+x)$ và $\sum_{i=0}^{n-1} x^i$ là các đa thức bất khả quy.

Ví dụ: $n = 5,11,19,...$

$$x^5 + 1 = (1+x)(1+x+x^2+x^3+x^4)$$

Bổ đề 2.15:

M-dãy trên vành đa thức có hai lớp kề xyclic $x^n + 1$ được tạo từ phương trình đồng dư sau:

$$\boxed{a(x) \equiv b(x)c^i(x) \bmod \sum_{i=0}^{n-1} x^i} \quad (2.16)$$

Trong đó: $\sum_{i=0}^{n-1} x^i$ là đa thức bất khả quy

$b(x)$ là đa thức bất kỳ, thỏa mãn $\deg b(x) \le n-2$

$c(x)$ là đa thức thỏa mãn $ord\, c(x) = \max = 2^{n-1} - 1$

Định nghĩa 2.8:

Cấp của đa thức $c(x)$ là cấp của nhóm nhân xyclic sinh bởi $c(x)$

$$C = \{c^i(x), i = 1,2,...\} \Rightarrow ord\, c(x) = |C|$$

Ví dụ 2.16:

Xét trường hợp $n = 5$ và $\sum_{i=0}^{4} x^i$ là đa thức bất khả quy, khi đó M-dãy được xây dựng theo công thức (2.16) có dạng như sau:

$$a(x) \equiv b(x)c^i(x) \bmod \sum_{i=0}^{4} x^i$$

Chú ý: để lấy modulo theo $\sum_{i=0}^{4} x^i = 1+x+x^2+x^3+x^4$ ta coi $1+x+x^2+x^3+x^4 = 0$ tức là coi $x^4 = 1+x+x^2+x^3$.

Chọn $b(x) = 1 \leftrightarrow (0)$ và $c(x) = 1+x^2+x^4 \leftrightarrow (024)$, chú ý (024) là dạng biểu diễn số mũ của $c(x)$. Nhóm nhân xây dựng từ đa thức sinh $c(x)$ như sau:

$$C = \{c^i(x), i = 1,2,...\} = \{(024),(034),(1),(013),(014),(2),(124)$$
$$(012),(3),(023),(123),(4),(134),(234),(0)\}$$

Trên cơ sở nhóm nhân C ta tính được M-dãy như sau:

$$A=\left\{c^i(x) \bmod \sum_{i=0}^{4} x^i\right\}=\{(13),(12),(1),(013),(23),(2),(03),(012)$$
$$(3),(023),(123),(0123),(02),(01),(0)\}$$

Bảng 2.3. M-dãy trên vành x^5+1

TT	Đa thức dạng số mũ	Đa thức	Dạng vectơ
1	(13)	$x+x^3$	0 1 0 1 ← M -dãy
2	(12)	$x+x^2$	0 1 1 0
3	(1)	x	0 1 0 0
4	(013)	$1+x+x^3$	1 1 0 1
5	(23)	x^2+x^3	0 0 1 1
6	(2)	x^2	0 0 1 0
7	(03)	$1+x^3$	1 0 0 1
8	(012)	$1+x+x^2$	1 1 1 0
9	(3)	x^3	0 0 0 1
10	(023)	$1+x^2+x^3$	1 0 1 1
11	(123)	$x+x^2+x^3$	0 1 1 1
12	(0123)	$1+x+x^2+x^3$	1 1 1 1
13	(02)	$1+x^2$	1 0 1 0
14	(01)	$1+x$	1 1 0 0
15	(0)	1	1 0 0 0

M-dãy trong Bảng 2.3 thực chất là một hoán vị của M-dãy trong Bảng 2.1.

Số các đa thức nguyên thủy (các phần tử sinh) tính theo công thức sau:

$$N=\varphi(|C|)$$

Trong đó φ là hàm Phi-Euler, giá trị của $\varphi(|C|)$ cho biết số các số nguyên tố cùng nhau với $|C|$. Cách tính φ đã được trình bày ở mục 1.1.12.

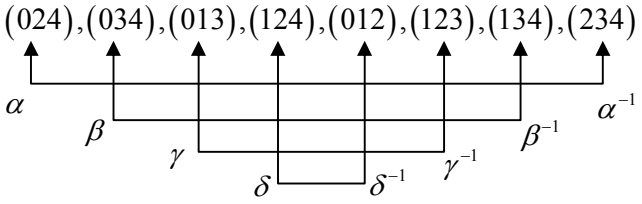
Ví dụ 2.17:

Xét $n=15=3.5$, khi đó:

$$\Phi(15)=15\left(1-\frac{1}{3}\right)\left(1-\frac{1}{5}\right)=8$$

Nếu α là một phần tử nguyên thủy thì α^i cũng là phần tử nguyên thủy với $(i,n)=1$.

Với trường hợp $n = 15$ ta có: $i = \{1,2,4,7,8,11,13,14\}$ tức là có 8 phần tử bằng với $\varphi(15)$. Và 8 phần tử nguyên thủy này sẽ tạo thành một nhóm nhân, các phần tử đều có nghịch đảo như mô tả như hình 2.12 sau đây:



Hình 2.12.

$$(024) = (234)^{-1} \; hay \; 1 + x^2 + x^4 = \frac{1}{x^2 + x^3 + x^4}$$

2.8. CHUẨN MÃ DỮ LIỆU

2.8.1. Mở đầu

Ngày 15/5/1973. Uỷ ban tiêu chuẩn quốc gia Mỹ đã công bố một khuyến nghị cho các hệ mật trong Hồ sơ quản lý liên bang. Điều này cuối cùng đã dẫn đến sự phát triển của Chuẩn mã dữ liệu (DES) và nó đã trở thành một hệ mật được sử dụng rộng rãi nhất trên thế giới. DES được IBM phát triển và được xem như một cải biên của hệ mật LUCIPHER. DES được công bố lần đầu tiên trong Hồ sơ Liên bang vào ngày 17/3/1975. Sau nhiều cuộc tranh luận công khai, DES đã được chấp nhận chọn làm chuẩn cho các ứng dụng không được coi là mật vào 5/1/1977. Kể từ đó cứ 5 năm một lần, DES lại được Uỷ ban Tiêu chuẩn Quốc gia xem xét lại. Lần đổi mới gần đây nhất của DES là vào tháng 1/1994 và tiếp tới sẽ là 1998. Người ta đoán rằng DES sẽ không còn là chuẩn sau 1998.

2.8.2. Mô tả DES

Mô tả đầy đủ của DES được nêu trong Công bố số 46 về các chuẩn xử lý thông tin Liên bang (Mỹ) vào 15/1/1977. DES mã hoá một xâu bit x của bản rõ độ dài 64 bằng một khoá 54 bit. Bản mã nhận được cũng là một xâu bit có độ dài 64. Trước hết ta mô tả ở mức cao về hệ thống.

Thuật toán tiến hành theo 3 giai đoạn:

1. Với bản rõ cho trước x, một xâu bit x_0 sẽ được xây dựng bằng cách hoán vị các bit của x theo phép hoán vị cố định ban đầu IP. Ta viết:

   $x_0 = \text{IP}(x) = L_0 R_0$

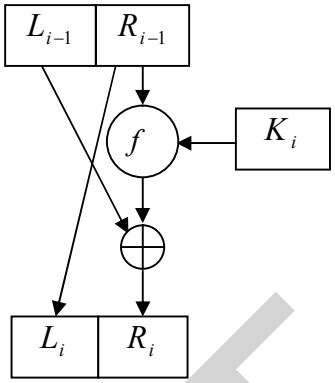
   trong đó L_0 gồm 32 bit đầu và R_0 là 32 bit cuối.

2. Sau đó tính toán 16 lần lặp theo một hàm xác định. Ta sẽ tính $L_i, R_i, 1 \le i \le 16$ theo quy tắc sau:

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, k_i) \end{cases}$$

trong đó $\oplus$ kí hiệu phép hoặc loại trừ của hai xâu bit (cộng theo modulo 2). f là một hàm mà ta sẽ mô tả ở sau, còn $k_1,k_2,...,k_{16}$ là các xâu bit độ dài 48 được tính như hàm của khoá k. (trên thực tế mỗi k_i là một phép chọn hoán vị bit trong k).

$k_1,k_2,...,k_{16}$ sẽ tạo thành bảng khoá. Một vòng của phép mã hoá được mô tả trên Hình 2.13.

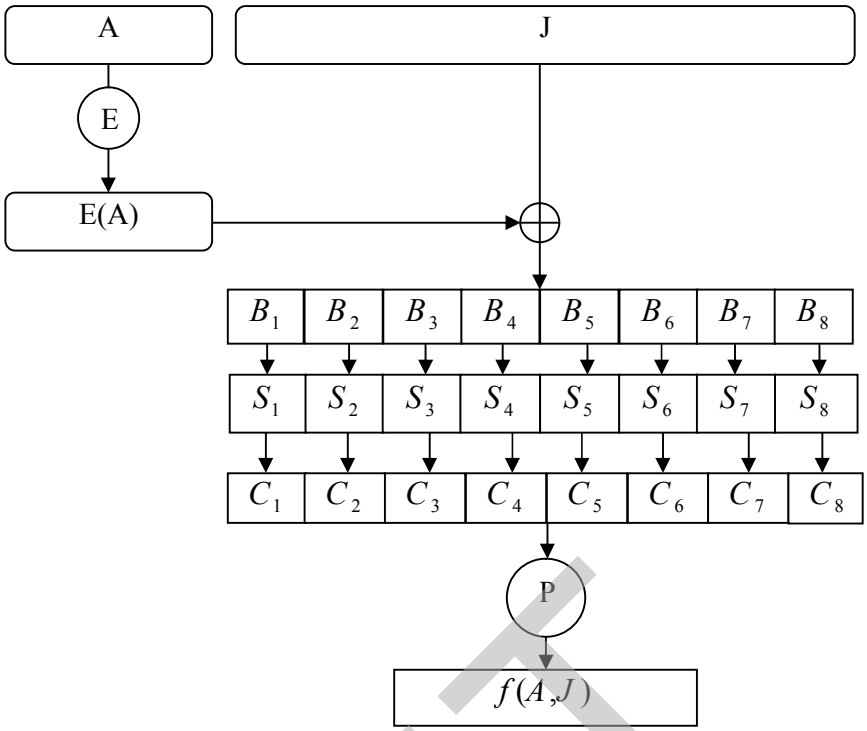


Hình 2.13. Một vòng của DES

3. Áp dụng phép hoán vị ngược IP^{-1} cho xâu bit $R_{16}L_{16}$, ta thu được bản mã y. Tức là $y = \text{IP}^{-1}(R_{16}L_{16})$. Hãy chú ý thứ tự đã đảo của R_{16} và L_{16}.

Hàm f có hai biến vào: biến thứ nhất *A* là xâu bit độ dài 32, biến thứ hai *J* là một xâu bit độ dài 48. Đầu ra của *f* là một xâu bit độ dài 32. Các bước sau được thực hiện:

1. Biến thứ nhất *A* được mở rộng thành một xâu bit độ dài 48 theo một hàm mở rộng cố định *E*(*EA*) gồm 32 bit của *A* (được hoán vị theo cách cố định) với 16 bit xuất hiện hai lần.
2. Tính $E(A) \oplus J$ và viết kết quả thành một chuỗi 8 xâu 6 bit = $B_1B_2B_3B_4B_5B_6$.
3. Bước tiếp theo dùng 8 bảng $S_1,S_2,...,S_8$ (được gọi là các hộp S). Với mỗi S_i là một bảng 4×16 cố định có các hàng là các số nguyên từ 0 đến 15. Với xâu bit có độ dài 6 (kí hiệu $B_i = b_1b_2b_3b_4b_5b_6$), ta tính $S_j(B_j)$ như sau: hai bit b_1b_6 xác định biểu diễn nhị phân của hàng r của $S_i(0 \le r \le 3)$ và bốn bit $b_2b_3b_4b_5$ xác định biểu diễn nhị phân của cột c của $S_i(0 \le c \le 15)$. Khi đó, $S_j(B_j)$ sẽ xác định phần tử $S_j(r,c)$; phần tử này viết dưới dạng nhị phân là một xâu bit có độ dài 4. (Bởi vậy, mỗi S_j có thể được coi là một hàm mã mà đầu vào là một xâu bit có độ dài 2 và một xâu bit có độ dài 4, còn đầu ra là một xâu bit có độ dài 4). Bằng cách tương tự tính các $C_j = S_j(B_j), 1 \le j \le 8$.
4. Xâu bit $C = C_1C_2..C_8$ có độ dài 32 được hoán vị theo phép hoán vị cố định P. Xâu kết quả là $P(C)$ được xác định là $f(A,J)$.



Hình 2.14. Hàm *f* của DES

Hàm *f* được mô tả trong Hình 2.14. Chủ yếu nó gồm một phép thế (sử dụng hộp *S*), tiếp sau đó là phép hoán vị *P*. 16 phép lặp của *f* sẽ tạo nên một hệ mật tích nêu như ở mục 2.6.

Trong phần còn lại của mục này, ta sẽ mô tả hàm cụ thể được dùng trong DES. Phép hoán vị ban đầu IP như sau:

Bảng 2.4. Bảng IP của DES

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Bảng này có nghĩa là bit thứ 58 của x là bit đầu tiên của $\mathrm{IP}(x)$; bit thứ 50 của x là bit thứ hai của $\mathrm{IP}(x)$.v.v...

Phép hoán vi ngược IP^{-1} là:

Bảng 2.5. Bảng IP^{-1} của DES

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Hàm mở rộng E được xác đinh theo bảng sau:

Bảng 2.6. Hàm mở rộng E của DES

Bảng chọn E bit					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Tám hộp S như sau:

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	15	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Và phép hoán vị P có dạng:

Bảng 2.7. Phép hoàn vị P trong hàm *f* của DES

P			
16	7	20	21
29	12	28	17
1	15	23	26
2	8	24	14
5	18	31	10
32	27	3	9
19	13	30	6
22	11	4	25

Cuối cùng, ta cần mô tả việc tính toán bảng khoá từ khoá k. Trên thực tế, k là một xâu bit độ dài 64, trong đó 56 bit là khoá và 8 bit để kiểm tra tính chẵn lẻ nhằm phát hiện sai. Các bit ở các vị trí 8,16,..., 64 được xác định sao cho mỗi byte chứa một số lẻ các số "1". Bởi vậy, một sai sót đơn lẻ có thể phát hiện được trong mỗi nhóm 8 bit. Các bit kiểm tra bị bỏ qua trong quá trình tính bảng khoá.

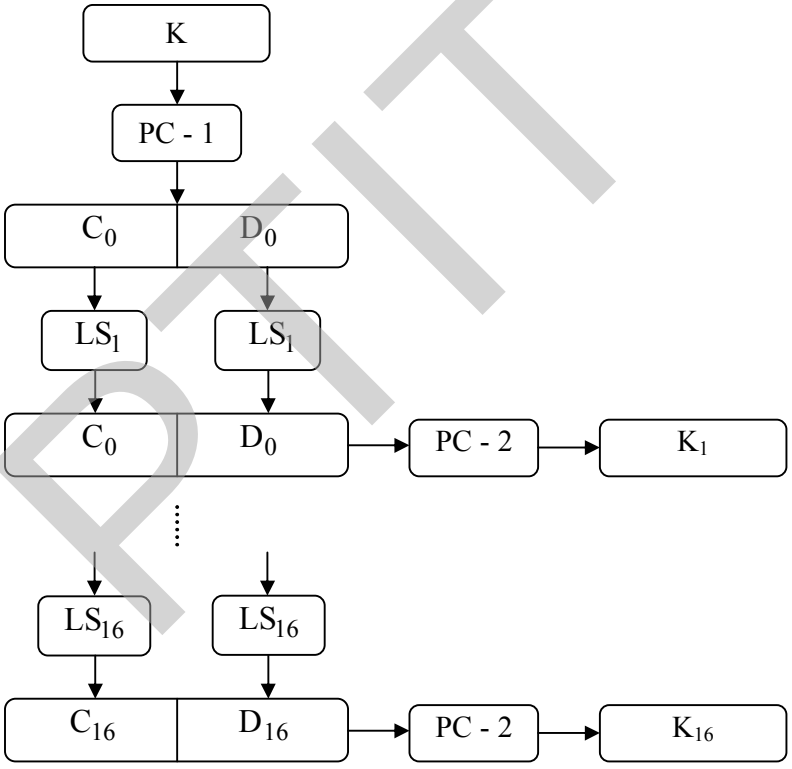
Với một khoá k 64 bit cho trước, ta loại bỏ các bit kiểm tra tính chẵn lẻ và hoán vị các bit còn lại của k theo phép hoán vị cố định PC-1. Ta viết:

$$PC-1(k)=C_0D_0$$

Với i thay đổi từ 1 đến 16:

$$C_i = LS_i(C_{i-1})$$
$$D_i = LS_i(D_{i-1})$$

Việc tính bảng khoá được mô tả trên Hình 2.15.



Hình 2.15. Tính bảng khoá DES

Các hoán vị PC-1 và PC-2 được dùng trong bảng khoá.

Bảng 2.8. Hoán vị PC-1

PC-1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Bảng 2.9. Hoán vị PC-2

PC-2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Bây giờ ta sẽ đưa ra bảng khoá kết quả. Như đã nói ở trên, mỗi vòng sử dụng một khoá 48 bit gồm 48 bit nằm trong K. Các phần tử trong các bảng dưới đây biểu thị các bit trong K trong các vòng khoá khác nhau.

Vòng 1												
10	51	34	60	49	17	33	57	2	9	19	42	
3	35	26	25	44	58	59	1	36	27	18	41	
22	28	39	54	37	4	47	30	5	53	23	29	
61	21	38	63	15	20	45	14	13	62	55	31	

Vòng 2												
2	43	26	52	41	9	25	49	59	1	11	34	
60	27	18	17	36	50	51	58	57	19	10	33	
14	20	31	46	29	63	39	22	28	45	15	21	
53	13	30	55	7	12	37	6	5	54	47	23	

Vòng 3												
51	27	10	36	25	58	9	33	43	50	60	18	
44	11	2	1	49	34	35	42	41	3	59	17	
61	4	15	30	13	47	23	6	12	29	62	5	
37	28	14	39	54	63	21	53	20	38	31	7	

Vòng 4
35 11 59 49 9 42 58 17 27 34 44 2
57 60 51 50 33 18 19 26 25 52 43 1
45 55 62 14 28 31 7 53 63 13 46 20
21 12 61 23 38 47 5 37 4 22 15 54

Vòng 5
19 60 43 33 58 26 42 1 11 18 57 51
41 44 35 34 17 2 3 10 9 36 27 50
29 39 46 61 12 15 54 37 47 28 30 4
5 63 45 7 22 31 20 21 55 6 62 38
Vòng 6
3 44 27 17 42 10 26 50 60 2 41 35
25 57 19 18 1 51 52 59 58 49 11 34
13 23 30 45 63 62 38 21 31 12 14 55
20 47 29 54 6 15 4 5 39 53 46 22

Vòng 7
52 57 11 1 26 59 10 34 44 51 25 19
9 41 3 2 50 35 36 43 42 33 60 18
28 7 14 29 47 46 22 5 15 63 61 39
4 31 13 38 53 62 55 20 23 37 30 6

Vòng 8
36 41 60 50 10 43 59 18 57 35 9 3
58 25 52 51 34 19 49 27 26 17 44 2
12 54 61 13 31 30 6 20 62 47 45 23
55 15 28 22 37 46 39 4 7 21 14 53

Vòng 9
57 33 52 42 2 35 51 10 49 27 1 60
50 17 44 43 26 11 41 19 18 9 36 59
4 46 53 5 23 22 61 12 54 39 37 15
47 7 20 14 29 38 31 63 62 13 6 45

Vòng 10
41 17 36 26 51 19 35 59 33 11 50 44
34 1 57 27 10 60 25 3 2 58 49 43
55 30 37 20 7 6 45 63 38 23 21 62
31 54 4 61 13 22 15 47 46 28 53 29

Vòng 11
25 1 49 10 35 3 19 43 17 60 34 57
18 50 41 11 59 44 9 52 51 42 33 27
39 14 21 4 54 53 29 47 22 7 5 46
15 38 55 45 28 6 62 31 30 12 37 13

Vòng 12
9 50 33 59 19 52 3 27 1 44 18 41
2 34 25 60 43 57 58 36 35 26 17 11
23 61 5 55 38 37 13 31 6 54 20 30
62 22 39 29 12 53 46 15 14 63 21 28
Vòng 13
58 34 17 43 3 36 52 11 50 57 2 25
51 18 9 44 27 41 42 49 19 10 1 60
7 45 20 39 22 21 28 15 53 38 4 14
46 6 23 13 63 37 30 62 61 47 5 12

Vòng 14
42 18 1 27 52 49 36 60 34 41 51 9
35 2 58 57 11 25 26 33 3 59 50 44
54 29 4 23 6 5 12 62 37 22 55 61
30 53 7 28 47 21 14 46 45 31 20 63

Vòng 15
26 2 50 11 36 33 49 44 18 25 35 58
19 51 42 41 60 9 10 17 52 43 34 57
38 13 55 7 53 20 63 46 21 6 39 45
14 37 54 12 31 5 61 30 29 15 4 47

Vòng 16
18 59 42 3 57 25 41 36 10 17 27 50
11 43 34 33 52 1 2 9 44 35 26 49
30 5 47 62 45 12 55 38 13 61 31 37
6 29 46 4 23 28 53 22 21 7 63 39

Phép giải mã được thực hiện nhờ dùng cùng thuật toán như phép mã nếu đầu vào là y nhưng dùng bảng khoá theo thứ tự ngược lại $K_{16}...K_1$. Đầu ra của thuật toán sẽ là bản rõ x.

Một ví dụ về DES

Sau đây là một ví dụ về phép mã DES. Giả sử ta mã bản rõ (ở dạng mã hexa):

0 1 2 3 4 5 6 7 8 9 A B C D E F

Bằng cách dùng khoá

1 2 3 4 5 7 7 9 9 B B C D F F 1

Khoá ở dạng nhị phân (không chứa các bit kiểm tra) là:

0001001001101001010110111100100110110111101101111111000

Sử dụng IP, ta thu được L_0 và R_0 (ở dạng nhị phân) như sau:

L_0 = 11001100000000000110011001111111
$L_1 = R_0$ = 11110000101010101111000010101010

Sau đó thực hiện 16 vòng của phép mã như sau:

$E(R_0)$ = 011110100001010101010101011110100001010101010101 K_1 = 000110110000001011101111111111000111000001110010 $E(R_0) \oplus K_1$ = 011000010001011110111010100001100110010100100111 S-box outputs 01011100100000101011010110010111 $f(R_0,K_1)$ = 00100011010010101010100110111011 $L_2 = R_1$ = 11101111010010100110010101000100
$E(R_1)$ = 011101011110101001010100001100001010101000001001 K_2 = 011110011010111011011001110110111100100111100101 $E(R_1) \oplus K_2$ = 000011000100010010001101111010110110001111101100 S-box outputs 11111000110100000011101010101110 $f(R_1,K_2)$ = 00111100101010111000011110100011 $L_3 = R_2$ = 11001100000000001011011100001001
$E(R_2)$ = 11100101100000000000001010111010111010000101010011 K_3 = 01010101111111001000101001000010110011111100111001 $E(R_2) \oplus K_3$ = 10110000011111001000100011111000001001111100101O S-box outputs 00100111000100001110000101101111 $f(R_2,K_3)$ = 01001101000101100110111010110000 $L_4 = R_3$ = 10100010010111000000101111110100
$E(R_3)$ =01010000010000101111100000000101011111111010100 K_4 = 01110010101011011101011011011011001101010001101 $E(R_3) \oplus K_4$ = 00100010111011110010111011011110010010101011010O S-box outputs 00100001111011011001111100111010 $f(R_3,K_4)$ = 10111011001000110111011101001100 $L_5 = R_4$ = 01110111001000100000000001000101
$E(R_4)$ = 10111010111010010000010000000000000000001000001010 K_5 = 01111100111011000000011111101011010100111010100O $E(R_4) \oplus K_5$ = 11000110000001010000001111101011010100011010001O S-box outputs 01010000110010000011000111101011 $f(R_4,K_5)$ = 00101000000100111010110111000011 $L_6 = R_5$ = 10001010010011111010011000110111

$E(R_5)$ = 11000101010000100101111111010000110000011010111
K_6 = 01100011101001010011111001010000011110110010111
$E(R_5) \oplus K_6$ =10100110111001110110000110000000101110101000000
S-box outputs 01000001111100110100110000111101
$f(R_5,K_6)$ = 10011110010001011100110100101100
$L_7 = R_6$ = 11101001011001111100110101101001
$E(R_6)$ = 11110101001010110000111111100101101010110101010011
K_7 = 11101100100001001011011111111011000011000101111100
$E(R_6) \oplus K_7$ = 00011001101011111011100000010011101100111111101111
S- box outputs 00010000011101010100000010101101
$f(R_6,K_7)$ = 10001100000001010001110000100111
$L_8 = R_7$ = 00000110010010101011101000010000
$E(R_7)$ = 00000000110000100101010101011111010000001010000
K_8 = 11110111100010100011101011000001001110111111111011
$E(R_7) \oplus K_8$ = 11110111010010000110111110011110011110110101011011
S-box outputs 01101100000110000111110010101110
$f(R_7,K_8)$ = 00111100000011101000011011111001
$L_9 = R_8$ = 11010101011010010100101110010000
$E(R_8)$ = 01101010101010110101001010100101011111100101000001
K_9 = 11100000110110111110101111101101111001111000000001
$E(R_8) \oplus K_9$ = 10001010011100001011100101001000100110110010000
S-box outputs 00010001000011000101011101110111
$f(R_8,K_9)$ = 00100010001101100111110001101010
$L_{10} = R_9$ = 00100100011111001100011001111010
$E(R_9)$ = 00010000100000111111100101100000110000111111010
K_{10} = 10110001111100110100011110111010010000110010011111
$E(R_9) \oplus K_{10}$ = 10100001011100001011111011011010100001011011101
S-box outputs 11011010000001000101001001110101
$f(R_9,K_{10})$ = 01100010101111001001110000100010
$L_{11} = R_{10}$ = 10110111110101011101011110110010
$E(R_{10})$ = 01011010111111101010101111110101011111101101001
K_{11} = 00100001010111111101001111011110110100111000011
$E(R_{10}) \oplus K_{11}$ = 01111011101000010111100000110100001011100010001
S-box outputs 01110011000001011101000100000001
$f(R_{10},K_{11})$ = 11100001000001001111101000000010
$L_{12} = R_{11}$ = 11000101011110000011110001111000
$E(R_{11})$ = 01100000101010111111000000011111100000111111000
K_{12} = 01110101011100011111010110010100011001111110100
$E(R_{11}) \oplus K_{12}$ = 00010101110110100000010110001011111001000001100
S-box outputs 01110011000001011101000100000001
$f(R_{11},K_{12})$ = 11000010011010001100111111101010
$L_{13} = R_{12}$ = 01110101101111010001100001011000
$E(R_{12})$ = 00111010101111011111101010001111000000101111000
K_{13} = 10010111110001011101000111111010101110100100000
$E(R_{12}) \oplus K_{13}$ = 10101101011110000010101101110101101110001011000
Sbox outputs 10011010110100011000101101001111
$f(R_{12},K_{13})$ = 11011101101110110010100100100010

$L_{14} = R_{13} = 00011000110000110001010101011010$

$E(R_{13}) = 000011110001011000000110100010101010101011110100$ $K_{13} = 010111110100001110110111111100101110011100111010$ $E(R_{13}) \oplus K_{14} = 010100000101010110110001011110000100110111001110$ S-box outputs 01100100011110011001101011110001 $f(R_{13},K_{14}) = 10110111001100011000111001010101$ $L_{15} = R_{14} = 11000010100011001001011000001101$
$E(R_{14}) = 11100000010101000101100101001010110000000001011011$ $K_{15} = 10111111100100011000110100111101001111110000101 0$ $E(R_{14}) \oplus K_{15} = 01011111110001011101010001110111111111111010100 01$ S-box outputs 10110010111010001000110100111100 $f(R_{14},K_{15}) = 01011011100000010010011101101110$ $R_{15} = 01000011010000100011001000110100$
$E(R_{15}) = 00100000011010100000010000011010010000011010 1000$ $K_{16} = 11001011001111011000101100001110000101111111 0101$ $E(R_{15}) \oplus K_{16} = 11101011010101111000111100010100010101100101 1101$ S-box outputs 10100111100000110010010000101001 $f(R_{15},K_{16}) = 11001000110000000100111110011000$ $R_{16} = 00001010010011001101100110010101$

Cuối cùng, áp dụng IP^{-1} vào L_{16},R_{16} ta nhận được bản mã hexa là:

85E813540F0AB405

2.8.3. Một số ý kiến thảo luận về DES

Khi DES được đề xuất như một chuẩn mật mã, đã có rất nhiều ý kiến phê phán. Một lý do phản đối DES có liên quan đến các hộp *S*. Mọi tính toán liên quan đến DES ngoại trừ các hộp *S* đều tuyến tính, tức việc tính phép hoặc loại trừ của hai đầu ra cũng giống như phép hoặc loại trừ của hai đầu vào rồi tính toán đầu ra. Các hộp *S* - chứa đựng thành phần phi tuyến của hệ mật là yếu tố quan trong nhất đối với độ mật của hệ thống (Ta đã thấy là các hệ mật tuyến tính - chẳng hạn như Hill - có thể dễ dàng bị mã thám khi bị tấn công bằng bản rõ đã biết). Tuy nhiên, tiêu chuẩn xây dựng các hộp S không được biết đầy đủ. Một số người đã gợi ý là các hộp S phải chứa các "cửa sập" được dấu kín, cho phép Cục An ninh Quốc gia Mỹ (NSA) giải mã được các thông báo nhưng vẫn giữ được mức độ an toàn của DES. Dĩ nhiên ta không thể bác bỏ được khẳng định này, tuy nhiên không có một chứng cớ nào được đưa ra để chứng tỏ rằng trong thực tế có các cửa sập như vậy.

Năm 1976 NSA đã khẳng định rằng, các tính chất sau của hộp *S* là tiêu chuẩn thiết kế:

- Mỗi hàng trong mỗi hộp *S* là một hoán vị của các số nguyên 0, 1,... , 15.
- Không một hộp *S* nào là một hàm Affine hoặc tuyến tính các đầu vào của nó.
- Việc thay đổi một bit vào của S phải tạo nên sự thay đổi ít nhất là hai bit ra.
- Đối với hộp *S* bất kỳ và với đầu vào x bất kỳ $S(x)$ và $S(x \oplus 001100)$ phải khác nhau tối thiểu là hai bit (trong đó x là xâu bit độ dài 6).
- Hai tính chất khác nhau sau đây của các hộp *S* có thể coi là được rút ra từ tiêu chuẩn thiết kế của NSA.

- Với hộp *S* bất kỳ, đầu vào x bất kỳ và với $e,f \in \{0,1\} : S(x) \neq S(x \oplus 1 \text{lef} 00)$.
- Với hộp *S* bất kỳ, nếu cố định một bit vào và xem xét giá trị của một bit đầu ra cố định thì các mẫu vào để bit ra này bằng 0 sẽ xấp xỉ bằng số mẫu ra để bit đó bằng 1. (Chú ý rằng, nếu cố định giá trị bit vào thứ nhất hoặc bit vào thứ 6 thì có 16 mẫu vào làm cho một bit ra cụ thể bằng 0 và có 16 mẫu vào làm cho bit này bằng 1. Với các bit vào từ bit thứ hai đến bit thứ 5 thì điều này không còn đúng nữa. Tuy nhiên, phân bố kết quả vẫn gần với phân bố đều. Chính xác hơn, với một hộp *S* bất kỳ, nếu ta cố định giá trị của một bit vào bất kỳ thì số mẫu vào làm cho một bit ra cố định nào đó có giá trị 0 (hoặc 1) luôn nằm trong khoảng từ 13 đến 19).

Người ta không biết rõ là liệu có còn một chuẩn thiết kế nào đầy đủ hơn được dùng trong việc xây dựng hộp S hay không.

Sự phản đối xác đáng nhất về DES chính là kích thước của không gian khoá: 2^{56} là quá nhỏ để đảm bảo an toàn thực sự. Nhiều thiết bị chuyên dụng đã được đề xuất nhằm phục vụ cho việc tấn công với bản rõ đã biết. Phép tấn công này chủ yếu thực hiện tìm khoá theo phương pháp vét cạn. Tức với bản rõ x 64 bit và bản mã y tương ứng, mỗi khoá đều có thể được kiểm tra cho tới khi tìm được một khoá k thoả mãn $e_k(x) = y$. (Cần chú ý là có thể có nhiều hơn một khoá k như vậy).

Ngay từ năm 1977, Diffie và Hellman đã gợi ý rằng có thể xây dựng một chip VLSI (mạch tích hợp mật độ lớn) có khả năng kiểm tra được 10^6 khoá /giây. Một máy có thể tìm toàn bộ không gian khoá cỡ 10^6 trong khoảng 1 ngày. Họ ước tính chi phí để tạo một máy như vậy khoảng 2.10^7\$.

Trong cuộc hội thảo tại hội nghị CRYPTO'93, Michael Wiener đã đưa ra một thiết kế rất cụ thể về máy tìm khoá. Máy này xây dựng trên một chip tìm khoá, có khả năng thực hiện đồng thời 16 phép mã và tốc độ tới 5×10^7 khoá/giây. Với công nghệ hiện nay, chi phí chế tạo khoảng 10,5\$/chip. Giá của một khung máy chứa 5760 chip vào khoảng 100.000\$ và như vậy nó có khả năng tìm ra một khoá của DES trong khoảng 1,5 ngày. Một thiết bị dùng 10 khung máy như vậy có giá chừng 10^6 \$ sẽ giảm thời gian tìm kiếm khoá trung bình xuống còn 3,5 giờ.

2.8.4. DES trong thực tế

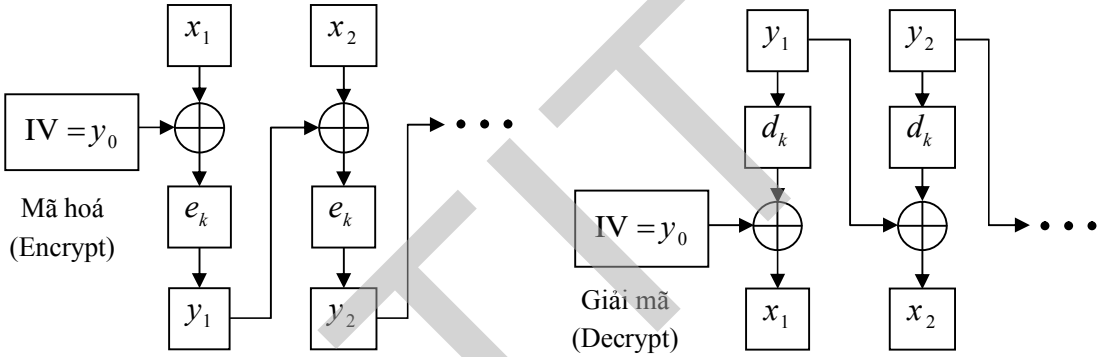
Mặc dù việc mô tả DES khá dài dòng song người ta có thể thực hiện DES rất hữu hiệu bằng cả phần cứng lẫn phần mềm. Các phép toán duy nhất cần được thực hiện là phép hoặc loại trừ các xâu bit. Hàm mở rộng *E*, các hộp *S*, các hoán vị IP và *P* và việc tính toán các giá trị $K_1,...,K_{16}$ đều có thể thực hiện được cùng lúc bằng tra bảng (trong phần mềm) hoặc bằng cách nối cứng chúng thành một mạch.

Các ứng dụng phần cứng hiện thời có thể đạt được tốc độ mã hoá cực nhanh. Công ty Digital Equipment đã thông báo tại hội nghị CRYPTO'92 rằng họ đã chế tạo một chip có 50 ngàn tranzistor có thể mã hoá với tốc độ 1 Gbit/s bằng cách dùng nhịp có tốc độ 250MHz. Giá của chip này vào khoảng 300\$. Tới năm 1991 đã có 45 ứng dụng phần cứng và chương trình cơ sở của DES được Uỷ ban tiêu Chuẩn quốc gia Mỹ (NBS) chấp thuận.

Một ứng dụng quan trọng của DES là trong giao dịch ngân hàng Mỹ - (ABA) DES được dùng để mã hoá các số định danh cá nhân (PIN) và việc chuyển tài khoản bằng máy thủ quỹ tự động (ATM). DES cũng được Hệ thống chi trả giữa các nhà băng của Ngân hàng hối đoái (CHIPS) dùng để xác thực các giao dịch vào khoảng trên $1{,}5\times10^{12}$ USA/tuần. DES còn được sử dụng rộng rãi trong các tổ chức chính phủ. Chẳng hạn như Bộ năng lượng, Bộ Tư pháp và Hệ thống dự trữ liên bang.

2.8.4.1. Các chế độ hoạt động của DES

Có 4 chế độ làm việc đã được phát triển cho DES: Chế độ quyển mã điện tử (ECB), chế độ phản hồi mã (CFB), chế độ liên kết khối mã (CBC) và chế độ phản hồi đầu ra (OFB). Chế độ ECB tương ứng với cách dùng thông thường của mã khối: với một dãy các khối bản rõ cho trước $x_1, x_2, \ldots$ (mỗi khối có 64 bit), mỗi x_i sẽ được mã hoá bằng cùng một khoá k để tạo thành một chuỗi các khối bản mã $y_1, y_2, \ldots$ theo quy tắc $y_i = e_k(y_{i-1} \oplus x_i)$. Việc sử dụng chế độ CBC được mô tả trên Hình 2.16.



Hình 2.16. Chế độ CBC

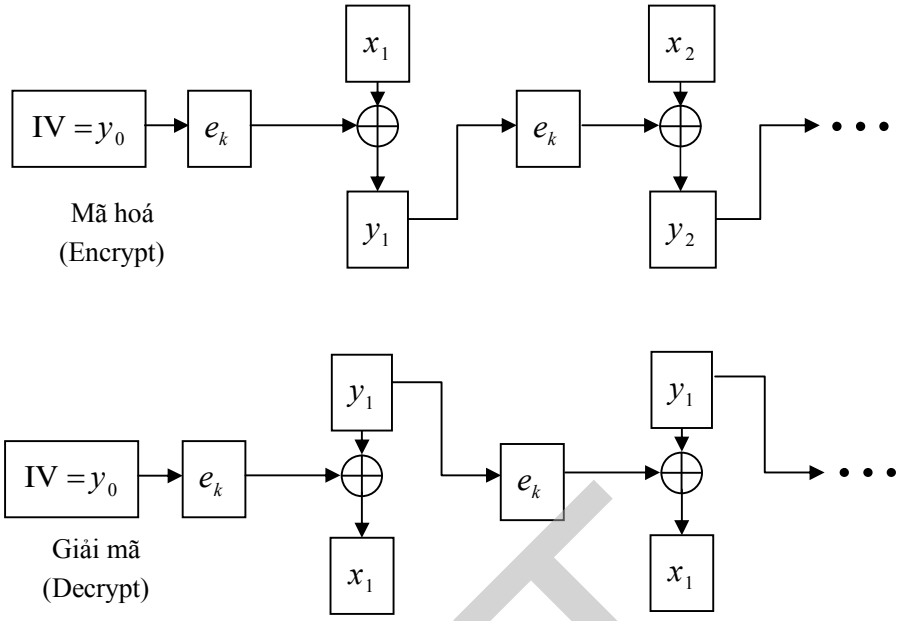
Trong các chế độ OFB và CFB dòng khoá được tạo ra sẽ được cộng mod 2 với bản rõ (tức là nó hoạt động như một hệ mã dòng, xem phần 3.8). OFB thực sự là một hệ mã dòng đồng bộ: dòng khoá được tạo bởi việc mã lặp vector khởi tạo 64 bit (vector IV). Ta xác định $z_0 = \text{IV}$ và rồi tính dòng khoá $z_1, z_2, \ldots$ theo quy tắc $z_i = e_k(z_{i-1}), i \geq 1$. Dãy bản rõ $x_1, x_2, \ldots$ sau đó sẽ được mã hoá bằng cách tính $y_i = x_i \oplus z_i, i \geq 1$.

Trong chế độ CFB, ta bắt đầu với $y_0 = \text{IV}$ (là một vector khởi tạo 64 bit) và tạo phần tử z_i của dòng khoá bằng cách mã hoá khối bản mã trước đó. Tức $z_i = e_k(y_{i-1}), i \geq 1$. Cũng như trong chế độ OFB: $y_i = x_i \oplus z_i, \ i \geq 1$. Việc sử dụng CFB được mô tả trên Hình 2.17 (chú ý rằng hàm mã DES e_k được dùng cho cả phép mã và phép giải mã ở các chế độ CFB và OFB).

Cũng còn một số biến thể của OFB và CFB được gọi là các chế độ phản hồi k bit $(1 < k < 64)$. Ở đây, ta đã mô tả các chế độ phản hồi 64 bit. Các chế độ phản hồi 1 bit và 8 bit thường được dùng trong thực tế cho phép mã hoá đồng thời 1 bit (hoặc byte) số liệu.

Bốn chế độ công tác có những ưu, nhược điểm khác nhau. Ở chế độ ECB và OFB, sự thay đổi của một khối bản rõ x_i 64 bit sẽ làm thay đổi khối bản mã y_i tương ứng, nhưng các

khối bản mã khác không bị ảnh hưởng. Trong một số tình huống, đây là một tính chất đáng mong muốn. Ví dụ, chế độ OFB thường được dùng để mã khi truyền vệ tinh.



Hình 2.17. Chế độ CFB

Mặt khác ở các chế độ CBC và CFB, nếu một khối bản rõ x_i bị thay đổi thì y_i và tất cả các khối bản mã tiếp theo sẽ bị ảnh hưởng. Như vậy các chế độ CBC và CFB có thể được sử dụng rất hiệu quả cho mục đích xác thực. Đặc biệt hơn, các chế độ này có thể được dùng để tạo mã xác thực bản tin (MAC - message authentication code). MAC được gắn thêm vào các khối bản rõ để thuyết phục Bob tin rằng, dãy bản rõ đó thực sự là của Alice mà không bị Oscar giả mạo. Như vậy MAC đảm bảo tính toàn vẹn (hay tính xác thực) của một bản tin (nhưng tất nhiên là MAC không đảm bảo độ mật).

Ta sẽ mô tả cách sử dụng chế độ BCB để tạo ra một MAC. Ta bắt đầu bằng vector khởi tạo IV chứa toàn số 0. Sau đó dùng chế độ CBC để tạo các khối bản mã $y_1,...,y_n$ theo khoá K . Cuối cùng ta xác định MAC là y_n . Alice sẽ phát đi dãy các khối bản rõ $x_1,...,x_n$ cùng với MAC. Khi Bob thu được $x_1,...,x_n$ anh ta sẽ khôi phục lại $y_1,...,y_n$ bằng khoá K bí mật và xác minh xem liệu y_n có giống với MAC mà mình đã thu được hay không?

Nhận thấy Oscar không thể tạo ra một MAC hợp lệ do anh ta không biết khoá K mà Alice và Bob đang dùng. Hơn nữa Oscar thu chặn được dãy khối bản rõ $x_1,...,x_n$ và thay đổi ít nhiều nội dung thì thì chắc chắn là Oscar không thể thay đổi MAC để được Bob chấp nhận.

Thông thường ta muốn kết hợp cả tính xác thực lẫn độ bảo mật. Điều đó có thể thực hiện như sau: Trước tiên Alice dùng khoá K_1 để tạo MAC cho $x_1,...,x_n$. Sau đó Alice xác định x_{n+1} là MAC rồi mã hoá dãy $x_1,...,x_{n+1}$ bằng khoá thứ hai K_2 để tạo ra bản mã $y_1,...,y_{n+1}$. Khi Bob thu được $y_1,...,y_{n+1}$, trước tiên Bob sẽ giải mã (bằng K_2) và kiểm tra xem x_{n+1} có phải là MAC đối với dãy $x_1,...,x_n$ dùng K_1 hay không.

Ngược lại, Alice có thể dùng K_1 để mã hoá $x_1,...,x_n$ và tạo ra được $y_1,...,y_n$, sau đó dùng K_2 để tạo MAC y_{n+1} đối với dãy $y_1,...,y_n$. Bob sẽ dùng K_2 để xác minh MAC và dùng K_1 để giải mã $y_1,...,y_n$.

2.8.4.2. Mã nguồn DES (Xem phụ lục 1)

2.8.5. Chuẩn mã dữ liệu tiên tiến (AES)

Vào 1997, Viện tiêu chuẩn và công nghệ quốc gia (NIST) Của Mỹ đã phát động cuộc thi nhằm xây dựng một chuẩn mã dữ liệu mới thay thế cho chuẩn mã dữ liệu cũ DES đã được đưa ra năm 1974. Qua quá trình tuyển chọn vào tháng 10 năm 2000, NIST đã công bố chuẩn mã dữ liệu mới được lựa chọn là thuật toán Rijndael. Đây là một mật mã khối đối xứng với ba kích thước khóa có thể lựa chọn (128 bit, 192 bit và 256 bit). Sau đây ta sẽ mô tả thuật toán AES này.

2.8.5.1. Cơ sở toán học của AES

Trong AES các phép toán cộng và nhân được thực hiện trên các byte trong trường hữu hạn $\mathrm{GF}(2^8)$.

Phép cộng:

Phép cộng giữa hai phần tử (các byte) trong trường hữu hạn được thực hiện bằng cách cộng theo modulo 2 các bit tương ứng trong biểu diễn của các byte này. Phép cộng các byte A và B với:

$$A = (a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8)$$
$$B = (b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7 \quad b_8)$$

là $C = A + B$ với $C = (c_1 \quad c_2 \quad c_3 \quad c_4 \quad c_5 \quad c_6 \quad c_7 \quad c_8)$

trong đó $C_i = a_i + b_i \bmod 2$ với $i = \overline{1,8}$

Các phần tử của trường hữu hạn còn có thể được biểu diễn dưới dạng đa thức. Ví dụ tổng của $A = 73_H$ và $B = 4E_H$ (viết dưới dạng cơ số 16 - hexa) là:

$$73_H + 4E_H = 3D_H$$

Viết dưới dạng nhị phân:

$$01110011 + 01001110 = 00111101$$

Viết dưới dạng đa thức:

$$(x^6 + x^5 + x^4 + x + 1) + (x^6 + x^3 + x^2 + x) = (x^5 + x^4 + x^3 + x^2 + 1)$$

Phép nhân:

Phép nhân được thực hiện trên $GF(2^8)$ bằng cách nhân hai đa thức rút gọn theo modulo của một đa thức bất khả quy $m(x)$.

Trong AES đa thức bất khả quy này là $m(x) = x^8 + x^4 + x^3 + x + 1$

Ví dụ 2.18: $A = \text{C3}_H$, $B = 85_H$ tương ứng với:

$$a(x) = x^7 + x^6 + x + 1 \text{ và } b(x) = x^7 + x^2 + 1$$

Khi đó C = A.B

$$c(x) = a(x)\,b(x) \bmod \left(x^8 + x^4 + x^3 + x + 1\right)$$

$$c(x) = x^7 + x^5 + x^3 + x^2 + x$$

hay $C = AE_H = 10101110$

2.8.5.2. Thuật toán AES

AES mã hóa một khối bản rõ M 128 bit thành một khối bản mã C 128 bit bằng cách dùng một khóa mã K có độ dài 128 bit (hoặc 192 hoặc 256 bit) tương ứng với $AES-128$ (hoặc $AES-192$ hoặc $AES-256$). Thuật toán thực hiện trên các byte và kích thước khối đối với đầu vào đầu ra và khóa được biểu thị bằng các từ 32 bit (4 byte).

AES sẽ thực hiện một số vòng mã hóa N_r phụ thuộc vào độ dài khóa được sử dụng (Bảng 2.10)

Bảng 2.10. Số các vòng mã hóa của AES

Thuật toán AES	Độ dài đầu vào/đầu ra	Độ dài khóa N_k	Số vòng N_r
$AES-128$	4 từ	4 từ	10 vòng
$AES-192$	4 từ	6 từ	12 vòng
$AES-256$	4 từ	8 từ	14 vòng

Mã hóa AES:

Mõi vòng gồm 4 phép biến đổi mật mã theo byte

- Thay thế byte
- Dịch các hàng của mảng trạng thái (State Array)
- Trộn dữ liệu trong một cột của State Array
- Cộng khóa vòng vào State Array

Phép thay thế byte: SubBytes( )

Phép biến đổi AES đầu tiên là một phép thay thế byte phi tuyến gọi là phép biến đổi SubBytes( ), nó hoạt động độc lập trên mỗi byte. Trước tiên nó sẽ tính nghịch đảo của phép nhân trong $GF\left(2^8\right)$, sau đó sử dụng một phép biến đổi trên nghịch đảo này.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

trong đó b_i biểu thị bit thứ i của byte b

Dịch các hàng của State Array; Phép biến đổi ShiftRows( )

Phép biến đổi tiếp theo của AES là dịch các hàng của State Array. Lượng dịch $\text{Shift}(r, N_b)$ phụ thuộc vào số hàng r. Các khối đầu vào (bản rõ) vào các khoái đầu ra (bản mã) là các khối 128 bit gồm $N_b = 4$ từ 32 bit

Phép biến đổi ShiftRows( ) được biểu thị như sau:

$$s'_{r,c} = s_r\left(c + shift\left(r, N_b\right)\right) \bmod N_b$$

trong đó $0 \le c \le N_b$

Hàng đầu tiên sẽ không dịch, tức là $\text{shift}(0, N_b = 4) = 0$

Với các hàng còn lại lượng dịch sẽ tùy theo số hàng

$$\text{shift}(0,4) = 0$$
$$\text{shift}(1,4) = 1$$
$$\text{shift}(2,4) = 2$$
$$\text{shift}(3,4) = 3$$

Trộn dữ liệu trong một cột State Array: Phép biến đổi Mixcolumns( )

Phép biến đổi Mixcolumns( ) được dùng để trộn dữ liệu trong một cột của ma trận trạng thái. Các cột được xem như các đa thức trong $GF\left(2^8\right)$. Đầu ra của Mixcolumns( ) là $s'(x)$ được tạo bằng cách nhân cột với $s(x)$ với đa thức $a(x)$ và rút gọn theo $\bmod\left(X^4 + 1\right)$

$$s'(x) = a(x)\, s(x) \bmod\left(x^4 + 1\right)$$

trong đó: $a(x) = 03_H x^3 + 01_H x + 02_H$

Ở dạng ma trận phép biến đổi này có thể viết như sau:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02_H & 03_H & 01_H & 01_H \\ 01_H & 02_H & 03_H & 01_H \\ 01_H & 01_H & 02_H & 03_H \\ 03_H & 01_H & 01_H & 02_H \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Ở đây $0 \le c < N_b$

Mở rộng khóa AES: KeyExpansion( )

Thuật toán AES sẽ tạo từ khóa mã 128 bit (hoặc 192 hoặc 256 bit) một tập khởi tạo N_b từ 32 bit và N_b từ 32 bit cho mỗi vòng bao gồm $N_b\left(N_r + 1\right)$ từ 32 bit. Chương trình giải mã KeyExpansion( ) chứa các SubWord( ) và RotWord( ).

Hàm SubWord( ) là một phép thay thế (hộp S) một từ vào 4 byte bằng một từ ra 4 byte.

Hàm RotWord( ) thực hiện phép hoán vị vòng các byte trong một từ 4 byte (32 bit) W_i:

$$RotWord(a_0,a_1,a_2,a_3)=(a_1,a_2,a_3,a_0)$$

KeyExpansion $\left(byte\,key\left[4_*N_k\right],word\,w\left[N_{b*}\left(N_r+1\right)\right],N_k\right)$

Begin

$i=0$

while $\left(i<N_k\right)$

$w[i]=word\left[key\left[4_*i\right],key\left[4_*i+1\right],key\left[4_*i+2\right],key\left[4_*i+3\right]\right]$

$i=i+1$

end while

$i \square N_k$

while $\left(i<N_{b*}\left(N_r+1\right)\right)$

word $temp=w[i-1]$

if $\left(i \bmod N_k=0\right)$

$temp=\text{SubWord}\left(\text{RotWord}\left(temp\right)\right)xor\ Rconw\left[i/N_k\right]$

else if $\left(N_k=8\ and\ i \bmod N_k=4\right)$

$temp=\text{SubWord}\left(temp\right)$

end if

$w[i] \square w\left[i-N_k\right]=xor\ temp$

$i=i+1$

end while

end

Chương trình giải mã của AES

Cipher $\left(byte\,in\left[4_*N_b\right],byte\,out\left[4_*N_b\right],word\,w\left[N_{b*}\left(N_r+1\right)\right]\right)$

Begin byte state $\left[4,N_b\right]$ state = in AddRoundKey(state,w)

for round = 1 step 1 to N_r-1

SubBytes (state), ShifRows (state),

Mixcolumns(state), AddRoundKey(state,w+round $_*N_b$)

end for

SubBytes (state), ShifRows (state)

AddRoundKey(state,w+$N_{r*}N_b$)

out = state

end

2.9. ƯU VÀ NHƯỢC ĐIỂM CỦA MẬT MÃ KHÓA BÍ MẬT

2.9.1. Ưu điểm

- Mật mã khóa bí mật (mật mã cổ điển) nói chung đơn giản, tức là các yêu cầu về phần cứng không phức tạp, thời gian tính toán nhanh.
- Mật mã khóa bí mật có tính hiệu quả, thông thường tốc độ mã $R_{mã} = 1$ (số bit đầu ra mã hóa bằng với số bit đầu vào).

Từ các ưu điểm này cho thấy mật mã cổ điển dễ sử dụng cho các dịch vụ nhạy cảm với độ trễ và các dịch vụ di động.

2.9.2. Nhược điểm

- Với mật mã khóa bí mật phải dùng kênh an toàn để truyền khóa, điều này dẫn đến chi phi sẽ cao hơn và việc thiết lập kênh an toàn khó khăn hơn.
- Việc tạo khóa và giữ bí mật khóa phức tạp. Khi làm việc trên mạng nếu dùng mật mã khóa bí mật sẽ phải tạo và lư trữ số lượng khóa nhiều.
- Nếu sử dụng mật mã khóa bí mật sẽ khó xây dựng các dịch vụ an toàn khác như các dịch vụ đảm bảo tính toàn vẹn của dữ liệu, dịch vụ xác thực và chữ ký số. Các dịch vụ này sẽ được thực hiện bởi mật mã khóa công khai.

BÀI TẬP CHƯƠNG 2

Bài 2.1: Thám mã thu được bản mã sau:

PSZI QIERW RIZIV LEZMRK XS WEC CSY EVI WSVVC

Biết rằng đây là bản mã của mật Caesar với khoá k chưa biết. Hãy dùng phương pháp tìm khoá vét cạn để tìn được bản rõ tiếng Anh tương ứng.

Ghi chú: Phương pháp tìm khoá vét cạn là phương pháp thử giải mã bằng mọi khoá có thể có.

Bài 2.2: Thám mã thu được bản mã sau:

_EHOHWSI_ON_E_TREVADYC_YQNOREUGNIOS_ EMAEFH
R_SATONEL_NRA DEEHTES_ERCO_TL_FEFI

Hãy chỉ ra rằng đây là một hệ mật hoán vị và thực hiện thám mã bằng phương pháp tìm khoá vét cạn. (Ký hiệu (_) là khoảng trắng (space)).

Bài 2.3: Thực hiện thám mã các bản mã sau của một hệ mật mã dịch vòng với khoá k chưa biết, bằng các phương pháp tìm khóa vét cạn và Thống kê, biết rằng các ký tự có xác suất xuất hiện lớn trong tiếng Anh được sắp xếp theo thứ tự sau:

_,E,T,A,H,O,N

Với giả sử ‘‘khoảng trống’’ (_) được xem là 1 ký tự

a) XMQIDMWDQSVIDZEPYEFPIDXLERDQSRIBDBSYDGERDKIXDQ
SVIDQSRIBDFYXDBSYDGERRSXDKIXDQSVIDXMQI

b) YMJEKTTQNXMERFSEXJJPXEMFUUNSJXXENSEYMJEI
NXYFSHJEYMJEANXJELWTAXENYEZSIJWEMNXEKJJY

c) ZNKFZX_KFYOMTFULFOTZKRROMKTIKFOYFTUZFQTUBRKJMKFH_ZFOSG
MOTGZOUT

d) IDRIZIVDORS_DXLIDPSZIDSJDSYVDTEVIRXWDJSVDYWDXM
PPD_IDLEZIDFIGSQIDTEVIRXW

Bài 2.4: Dưới đây là 4 bản mã thu được từ mã thay thế. Một bản thu được từ mã thay thế, một từ mã Vigenère, một từ mật mã Affine và một bản chưa xác định. Nhiệm vụ ở đây là xác định bản rõ trong mỗi trường hợp.

Hãy mô tả các bước cần thực hiện để giải mã mỗi bản mã (bao gồm tất cả các phân tích thống kê và các tính toán cần thực hiện).

Hai bản rõ đầu lấy từ cuốn "**The Diary of Samuel Marchbanks**" của Robertson Davies, Clack Iriwin,1947; bản rõ thứ tư lấy từ "**Lake Wobegon Days**" của Garrison Keillor, Viking Penguin, 1985.

a) Mã thay thế.

EMGLOSUDCGDNCUSWYSFHNSFCYKDPUMLWGYICOXYSIPJCK

QPKUGKMGOUCGINCGACKSNISACYKZSCKXEOCKSHYSXCG

OIDPKZCNKSHICGIWYGKKGKGOLDSILKGOIUSIGLEDSPWZU

GFZCCNDGYYSFUSZCNXEOJNCGYEOWEUPXEZGACGNFGLKNS

ACIGOIYCKXOUOUZCFZCCNDGYYSFEUEKUZCSOCFZCCNC

IACZEJNCSHFZEJZEGMXCYHCIUMGKUSY

Chỉ dẫn: **F** sẽ giải mã thành **w**.

b) Hệ mã Vigenère

KCCPKBGUFDPHQTYAVINRRTMVGRKDNBVFĐETDGILTXRGUD

DKOTFMBPVGEGLTGCKQRACQCWDNAWCRXLZAKFTLEWRPTVC

QKYVXCHKFTPONCQQRHJVAJUWETMCMSPKQDYHJVDAHCTRL

SVSKCGCZQọDZXGSFRLSWCWSJTBHAFSLASPRJAHKJRJUMV

GKMITZHFPDLSPZLVLGWTFPLKKEBDPGCEBSHCTJRWXBAFS

PEZQNRWXCVYCGAONWDDKACKAWBBIKFTLOVKCGGHJVLNHI

FFSQESVYCLACNVRWBBIREPBBVFEXOSCDYGZWPFDTKFQLY

CWHJVTNHIQ/BTKH/VNPIST

c) Hệ mã Affine.

KQEREJEBCPPCJCRKIEACUZBKRVPKRBCIBQCARBJCVFCUP

KRLOFKPACUZQEPBKRXPEIIEABDKPBCPFCDCCAFIEABĐKP

BCPFEQPKAZBKRHALBKAPCCIBURCCDKDCCJC/DFUIXPAFF

ERBICZDFKABICBBENEFCUPLCVKABPCYDCCDPKBCOCPERK

IVKSCPICBRKLJPKABL

d) Hệ mã chưa xác định được.

BNVSNSIHQCEELSSKKYERIFJKXUMBGVKAMQLJTYAVFBKVT

DVBPVVRJYYLAOKYMPQSCGDLFSRLLPROYGESEBUUALRWXM

MASAZLGLEĐFJBZAVVPXWI CGJXASCBYEHOSNMULKCEAHTQ

OKMFLEBKFXLRRFDTZXCIWBJSICBGAWDVYDHAVFJXZIBKC

GJIWEAHTTOEWTUHKRQVVRGZBXYIREMMASCSPBNLHJMBLR

FFJELHWEYLWISTFVVYFJCMHYUYRUFSFMGESIGRLWALSVVM

NUHSIMYYITCCQPZSICEHBCCMZFEGVJYOCDEMMPGHVAAUM

ELCMOEHVLTIPSUYILVGFLMVWDVYDBTHFRAYISYSGKVSUU

HYHGGCKTMBLRX

Bài 2.5:

a) Có bao nhiêu ma trận khả nghịch cấp 2×2 trên $\mathbf{Z}_{26}$?

b) Giả sử p là số nguyên tố. Hãy chứng tỏ số các ma trận khả nghịch cấp 2×2 trên $\mathbf{Z}_p$ là $\left(p^2-1\right)\left(p^2-p\right)$.

Chỉ dẫn Vì p là số nguyên tố nên $\mathbf{Z}_p$ là một trường. Hãy sử dụng khẳng định sau: Một ma trận trên một trường là khả nghịch khi và chỉ khi các hàng của nó là các véc tơ độc lập tuyến tính (tức không tồn tại một tổ hợp tuyến tính các hàng khác 0 mà tổng của chúng là một véc tơ toàn số 0).

c) Với p là số nguyên tố và m là một số nguyên $m\geq 2$. Hãy tìm công thức tính số các ma trận khả nghịch cấp $m\times m$ trên $\mathbf{Z}_p$.

Bài 2.6: Giả sử ta đã biết rằng bản rõ "**conversation**" sẽ tạo nên bản mã "**HIARRTNUYTUS**" (được mã theo hệ mã Hill nhưng chưa xác định được m). Hãy xác định ma trận mã hoá.

Bài 2.6: Hệ mã Affine - Hill là hệ mã Hill được sửa đổi như sau: Giả sử m là một số nguyên dương và $P=C=\left(\mathbf{Z}_{26}\right)^m$. Trong hệ mật này, khoá K gồm các cặp $\left(L,b\right)$, trong đó L là một ma trận khả nghịch cấp $m\times m$ trên $\mathbf{Z}_{26}$ và $b\in\left(\mathbf{Z}_{26}\right)^m$ theo công thức $y=xL+b$. Bởi vậy, nếu $L=\left(l_{ij}\right)$ và $b=\left(b_1,...,b_m\right)$ thì:

$$\left(y_1,...,y_m\right)=\left(x_1,...,x_m\right)\begin{pmatrix} l_{1,1} & l_{1,2} & \dots & l_{1,m} \\ l_{2,1} & l_{2,2} & \dots & l_{2,m} \\ \vdots & \vdots & & \vdots \\ l_{m,1} & l_{m,2} & \dots & l_{m,m} \end{pmatrix}+\left(b_1,...,b_m\right)$$

Giả sử Oscar đã biết bản rõ 1à ***"adisplayedequation"*** và bản mã tương ứng là "**DSRMSIOPLXLJBZULLM**". Oscar cũng biết $m=3$. Hãy tính khoá và chỉ ra tất cả các tính toán cần thiết?

Bài 2.7: Sau đây là cách thám mã hệ mã Hill sử dụng phương pháp tấn công chỉ với bản mã. Giả sử ta biết $m=2$. Chia các bản mã thành các khối có độ dài 2 kí tự (các bộ đôi). Mỗi bộ đôi này là bản mã của một bộ đôi của bản rõ nhờ dùng một ma trận mã hoá chưa biết. Hãy nhặt ra các bộ đôi thường gặp nhất trong bản mã và coi rằng đó là mã của một bộ đôi thường gặp trong danh sách ở bảng 1.1 (ví dụ TH và ST). Với mỗi giả định, hãy thực hiện phép tấn công với bản rõ đã biết cho tới. khi tìm được ma trận giải mã đúng.

Sau đây là một ví dụ về bản mã để bạn giải mã theo phương pháp đã nêu:

LMQETXYEAGTXCTUIEWNCTXLZEWUAISPZYVAPEWLMGQWVA

XFTGMSQCADAGTXLMDXNXSNPJQSYVAPRIQSMHNOCVAXFV.

Bài 2.8: Ta sẽ mô tả một trường hợp đặc biệt của mã hoán vị. Giả sử m, n là các số nguyên dương. Hãy viết bản rõ theo thành từng hàng thành một hình chữ nhật $m \times n$. Sau đó tạo ra bản mã bằng cách lấy các cột của hình chữ nhật này Ví dụ, nếu $m = 4, n = 3$ thì ta sẽ mã hoá bản rõ "**cryptography**" bằng cách xây dựng hình chữ nhật :

cryp

togr

aphy

Bản mã sẽ là: "**CTAROPYGHPRY**"

a) Hãy mô tả cách Bob giải mã một bản mã (với m, n đã biết).

b) Hãy giải mã bản mã sau: (nhận được theo phương pháp đã nêu):

MYAMRARUYIQTENCTORAHROYWĐSOYEOUARRGĐERNOGW

Bài 2.9: Hãy chứng minh rằng phép giải mã DES có thể thực hiện bằng cách áp dụng thuật toán mã hoá DES cho bản rõ với bảng khoá đảo ngược.

Bài 2.10: Cho $DES(x, K)$ là phép mã hoá DES của bản rõ x với khoá K. Giả sử $y = DES(x, K)$ và $y' = DES\left(c(x), c(K)\right)$ trong đó $c(.)$ kí hiệu là phần bù theo các bit của biến. Hãy chứng minh rằng $y = c(y)$ (tức là nếu lấy phần bù của bản rõ và khoá thì bản mã kết quả cũng là phần bù của bản mã ban đầu). Chú ý rằng kết quả trên có thể chứng minh được chỉ bằng cách sử dụng mô tả "mức cao" của DES - cấu trúc thực tế của các hộp S và các thành phần khác của hệ thống không ảnh hưởng tới kết quả này.

Bài 2.11: Mã kép là một cách để làm mạnh thêm cho DES: với hai khóa K_1 và K_2 cho trước, ta xác định $y = e_{K_2}(e_{K_1}(x))$ (dĩ nhiên đây chính là tích của DES với chính nó). Nếu hàm mã hoá e_{K_2} giống như hàm giải mã d_{K_1} thì K_1 và K_2 được gọi là các khoá đối ngẫu (đây là trường hợp không mong muốn đối với phép mã kép vì bản mã kết quả lại trùng với bản rõ). Một khoá được gọi là tự đối ngẫu nếu nó đối ngẫu với chính nó.

a) Hãy chứng minh rằng nếu C_0 gồm toàn các số 0 hoặc gồm toàn các số 1 và D_0 cũng vậy thì K là tự đối ngẫu.

b) Hãy tự chứng minh rằng các khoá sau (cho ở dạng hexa) là tự đối ngẫu:

0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

F E E F E F E F E F E F E F E F

1 F 1 F 1 F 1 F 0 F 0 F 0 F 0 F

E 0 E 0 E 0 E 0 F 1 F 1 F 1 F 1

c) Hãy chứng tỏ rằng nếu $C_0 = 0101...01$ hoặc $1010...10$ (ở dạng nhị phân) thì XOR các xâu bit C_i và C_{17-i} là $111...11$, với $1 \le i \le 16$ (khẳng định tương tự cũng đúng đối với D_i).

d) Hãy chứng tỏ các cặp khoá sau là đối ngẫu:

E001E001F101F101

FE1FFE1FF0EFE0E

E01FE01FFF10FF10

01E001E001F101F1

1FEFE1FFE0EFE0EFE

1FE01FE00EF10EF1

CHƯƠNG 3. MẬT MÃ KHOÁ CÔNG KHAI

Trước khi tìm hiểu hệ mật khóa công khai chúng ta ôn tập lại một số kiến thức về lý thuyết số.

3.1. SỐ HỌC MODULO

3.1.1. Số nguyên

Tập các số nguyên:

$$\{\ldots,-3,-2,-1,0,1,2,3,\ldots\}=\mathbf{Z} \tag{3.1}$$

Định nghĩa 3.1:

Cho $a,b\in\mathbf{Z}$, a là ước của b nếu $\exists c\in\mathbf{Z}:\ b=ac$. Ký hiệu $a\,|b$

Các tính chất chia hết

$\forall a,b,c\in\mathbf{Z}$ ta có:

(i) $a\,|a$.

(ii) Nếu $a\,|b$ và $b\,|c$ thì $a\,|c$

(iii) Nếu $a\,|b$ và $a\,|c$ thì $a\,|(bx+cy)$ với $\forall x,y\in\mathbf{Z}$

(iv) Nếu $a\,|b$ và $b\,|a$ thì $a=\pm b$

Định nghĩa 3.2:

Thuật toán chia đối với các số nguyên:

Nếu a và b là các số nguyên với $b\geq 1$ thì $a=qb+r,\ \ 0\leq r<b$

q và r là duy nhất.

Phần dư của phép chia a và b được ký hiệu $a \bmod b=r$

Thương của phép chia a và b được ký hiệu $a\,div\,b=q$

Ta có $a\,div\,b=\left[\frac{a}{b}\right]$, $\quad a \bmod b=a-b\left[\frac{a}{b}\right]$

Ví dụ 3.1: $a=73,b=17\Rightarrow 73 div\,17=4,\quad 73 \bmod 17=5$

Định nghĩa 3.3: Ước chung.

c là ước chung của a và b nếu $c\,|a\ \&\ c\,|b$

Định nghĩa 3.4: Ước chung lớn nhất (ƯCLN)

Số nguyên dương d là ƯCLN của các số nguyên a và b (Ký hiệu$d=(a,b)$) nếu:

(i) d là ước chung của a và b.

(ii) Nếu có $c \mid a$ và $c \mid b$ thì $c \mid d$.

Như vậy (a,b) là số nguyên dương lớn nhất ước của cả a và b không kể $(0,0) = 0$.

Ví dụ 3.2:

Các ước chung của 12 và 18 là $\{\pm 1, \pm 2, \pm 3, \pm 6\} \rightarrow (12,18) = 6$

Định nghĩa 3.5: Bội chung nhỏ nhất (BCNN)

Số nguyên dương d là bội chung nhỏ nhất (BCNN) của các số nguyên a và b (Ký hiệu $d = BCNN(a,b)$) nếu:

(i) $a \mid d$, $b \mid d$.

(ii) Nếu có $a \mid c$, $b \mid c$ thì $d \mid c$.

Như vậy d là số nguyên dương nhỏ nhất là bội của cả a và b.

Tính chất

$$BCNN(a,b) = \frac{a.b}{(a,b)} \tag{3.2}$$

Ví dụ 3.3: $(12,18) = 6 \quad \Rightarrow \quad BCNN(12,18) = \frac{12.18}{6} = 36$

Định nghĩa 3.6:

Hai số nguyên dương a và b được gọi là nguyên tố cùng nhau nếu: $(a,b) = 1$

Định nghĩa 3.7:

Số nguyên $p \geq 2$ được gọi là số nguyên tố nếu các ước dương của nó chỉ là 1 và p. Ngược lại p được gọi là hợp số.

Định lý 3.1: (**Định lý cơ bản của số học**)

Với mỗi số nguyên $\boldsymbol{n} \geq 2$ ta luôn phân tích được dưới dạng tích của luỹ thừa của các số nguyên tố.

$$\boldsymbol{n} = \boldsymbol{p}_1^{e_1} \boldsymbol{p}_2^{e_2} \ldots \boldsymbol{p}_k^{e_k} \tag{3.3}$$

Trong đó p_i là các số nguyên tố khác nhau và e_i là các số nguyên dương. Hơn nữa phân tích trên là duy nhất.

Định nghĩa 3.8:

Với $n \geq 2$, hàm $\varphi(n)$ được xác định là số các số nguyên trong khoảng $[1,n]$ nguyên tố cùng nhau với n.

Các tính chất của hàm $\varphi(n)$

Nếu p là số nguyên tố thì $\varphi(p) = p - 1$.

Nếu $(m, n) = 1$ thì $\varphi(m.n) = \varphi(m).\varphi(n)$.

Nếu $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ là phân tích ra thừa số nguyên tố của n thì:

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\dots\left(1 - \frac{1}{p_k}\right) \tag{3.4}$$

Định lý 3.2: Với $\forall n \geq 5$:

$$\varphi(n) > \frac{n}{6 \ln \ln n} \tag{3.5}$$

3.1.2. Các thuật toán trong Z

Cho a và b là các số nguyên không âm và nhỏ hơn hoặc bằng $a \times b = b \times a$. Cần chú ý rằng số các bit trong biểu diễn nhị phân của n là $\lfloor \lg n \rfloor + 1$ và số này xấp xỉ bằng $\lg n$. Số các phép toán bit đối với bốn phép toán cơ bản trên các số là cộng, trừ, nhân và chia sử dụng các thuật toán kinh điển được tóm lược trên bảng sau. Các kỹ thuật tinh tế hơn đối với các phép toán nhân và chia sẽ có độ phức tạp nhỏ hơn.

Bảng 3.1. Độ phức tạp bit của các phép toán cơ bản trong Z

Phép toán		Độ phức tạp bit
Cộng	$a + b$	$0(\lg a + \lg b) = 0(\lg n)$
Trừ	$a - b$	$0(\lg a + \lg b) = 0(\lg n)$
Nhân	$a.b$	$0\left((\text{lga}).(\text{lgb})\right) = 0\left((\lg n)^2\right)$
Chia	$a = qb + r$	$0\left((\text{lga}).(\text{lgb})\right) = 0\left((\lg n)^2\right)$

ƯCLN của 2 số nguyên a và b có thể được tính theo định lý sau:

Định lý 3.3:

Nếu $a = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, $b = p_1^{f_1} p_2^{f_2} \dots p_k^{f_k}$ trong đó $e_i \geq 0$, $f_i \geq 0$

thì $UCLN(a, b) = p_1^{\min(e_1, f_1)} p_2^{\min(e_2, f_2)} \dots p_k^{\min(e_k, f_k)}$

và $BCNN(a, b) = p_1^{\max(e_1, f_1)} p_2^{\max(e_2, f_2)} \dots p_k^{\max(e_k, f_k)}$

Ví dụ 3.4: Cho $a = 4864 = 2^8.19$, $b = 3458 = 2.7.13.19$. Khi đó

$$UCLN(a, b) = (4864, 3458) = 2.19 = 38$$
$$BCNN(a, b) = (4864, 3458) = 2^8.7.13.19 = 442624$$

Định lý 3.4: Nếu a và b là các số nguyên dương với $a > b$ thì:

$$UCLN(a,b) = UCLN(b, a \bmod b) \tag{3.6}$$

Thuật toán Euclide sau sẽ cho ta cách tính ƯCLN rất hiệu quả mà không cần phải phân tích ra thừa số nguyên tố.

Thuật toán Euclide

Tính UCLN của 2 số nguyên

VÀO : Hai số nguyên không âm a và b với $a > b$

RA : ƯCLN của a và b.

(1) While $b \neq 0$ do

$$r \leftarrow a \bmod b \ , \ a \leftarrow b \ , \ b \leftarrow r$$

(2) Return (a).

Định lý 3.5: Thuật toán trên có thời gian chạy chừng $0((\lg n)^2)$ các phép toán bit.

Ví dụ 3.5: Sau đây là các bước chia của thuật toán trên khi tính:

$$(4864, 3458) = 38$$
$$4864 = 1.3458 + 1406$$
$$3458 = 2.1406 + 646.$$
$$1406 = 2.646 + 76$$
$$646 = 5.114 + 38$$
$$76 = 2.38 + 0$$

Thuật toán trên có thể được mở rộng để không những chỉ tính được ƯCLN của 2 số nguyên a và b mà còn tính được các số nguyên x và y thoả mãn $ax + by = d$.

Thuật toán Euclide mở rộng

VÀO : Hai số nguyên không âm a và b với $a \geq b$

RA : $d = UCLN(a,b)$ và các số nguyên x và y thoả mãn $ax + by = d$.

Nếu $b = 0$ thì đặt $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$ và $return(d,x,y)$

(1) Đặt $x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$, $y_1 \leftarrow 1$

(2) While $b > 0$ do

2.1 $q \leftarrow \lfloor a/b \rfloor, r \leftarrow a - qb, x \leftarrow x_2 - qx_1, y \leftarrow y_2 - qy_1$

2.2 $a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$, $y_1 \leftarrow y$

(3) Đặt $d \leftarrow a$, $x \leftarrow x_2$, $y \leftarrow y_2$ và $return(d,x,y)$

Định lý 3.6:

Thuật toán trên có thời gian chạy cỡ $0((\lg n)^2)$ các phép toán bit.

Ví dụ 3.6: Bảng 3.2 sau chỉ ra các bước của thuật toán trên với các giá trị vào $a = 4864$ và $b = 3458$

Bảng 3.2. Thuật toán Euclide mở rộng với các đầu vào $a = 4864$ và $b = 3458$

Q	r	x	y	a	b	x_1	x_2	y_2	y_1
–	–	–	–	4864	3458	1	0	0	1
1	1406	1	–1	3458	1406	0	1	1	–1
2	646	–2	3	1406	646	1	–2	–1	3
2	114	5	–7	646	114	–2	5	3	7
5	76	–27	38	114	76	5	–27	–7	38
1	38	32	–45	76	38	–27	32	38	–45
2	0	–91	128	38	0	32	–91	–45	128

Bởi vậy ta có:

$UCLN\left(4864,3458\right) = 38$ và $\left(4864\right)\left(32\right)+\left(3458\right)\left(–45\right) = 38$

3.1.3. Các số nguyên modulo n

Định nghĩa 3.9:

Nếu a và b là các số nguyên thì a được gọi là đồng dư với b theo modulo (ký hiệu là $a \equiv b \bmod n$) nếu $n \left|(a-b)\right.$.

Số nguyên n được gọi là modulo đồng dư.

Ví dụ 3.7: $24 \equiv 9 \bmod 5$ vì $24 – 9 = 3.5$

$–11 \equiv 17 \bmod 7$ vì $–11 – 17 = –4.7$

Các tính chất

Đối với $a,a_1,b,b_1,c \in \mathbf{Z}$ ta có:

(1) $a \equiv b\left(\bmod n\right)$ nếu và chỉ nếu a và b cũng có phần dư khi chia cho n .

(2) Tính phản xạ : $a \equiv a\left(\bmod n\right)$.

(3) Tính đối xứng : Nếu $a \equiv b\left(\bmod n\right)$ thì $b \equiv a\left(\bmod n\right)$

Tính bắc cầu : Nếu $a \equiv b\left(\bmod n\right)$ và $b \equiv c\left(\bmod n\right)$ thì $a \equiv c\left(\bmod n\right)$

(4) Nếu $a \equiv a_1\left(\bmod n\right)$ và $b \equiv b_1\left(\bmod n\right)$ thì

$a+b \equiv a_1+b_1\left(\bmod n\right)$ và $ab \equiv a_1b_1\left(\bmod n\right)$

Lớp tương đương của một số nguyên a là tập các số nguyên đồng dư với a modulo n. Từ các tính chất (2), (3) và (5) ở trên ta có thể thấy rằng đối với n cố định, quan hệ đồng dư theo modulo n sẽ phân hoạch $\mathbf{Z}$ thành các lớp tương đương.

Nếu $a = qn + r$ với $0 \le r \le n$ thì $a \equiv r \left(\mathrm{mod} n\right)$.

Bởi vậy mỗi số nguyên a là đồng dư theo modulo n với một số nguyên duy nhất nằm trong khoảng từ 0 tới $n-1$, số này được gọi là thặng dư tối thiểu của $a \bmod n$. Như vậy a và r có thể được dùng để biểu thị cho lớp tương đương này.

Định nghĩa 3.10:

Các số nguyên modulo n (ký hiêu $\mathbf{Z}_n$) là tập (các lớp tương đương) của các số nguyên $\{0,1,2,\ldots,n-1\}$. Các phép cộng, trừ, nhân trong $\mathbf{Z}_n$ được thực hiện theo modulo n.

Ví dụ 3.8: $\mathbf{Z}_{25} = \{0,1,\ldots,24\}$. Trong $\mathbf{Z}_{25}$ ta có:

$$13 + 16 = 4 \text{ vì } 13 + 16 = 29 \equiv 4 \left(\mathrm{mod}\, 25\right)$$

Tương tự $13.16 = 8$ trong $\mathbf{Z}_{25}$.

Định nghĩa 3.11: (Phần tử nghịch đảo)

Cho $a \in Z_n$, phần tử nghịch đảo (ngược theo phép nhân) của $a \bmod n$ là một số nguyên $x \in Z_n$ sao cho: $ax \equiv 1 \left(\mathrm{mod} n\right)$

Nếu x tồn tại thì nó là duy nhất, a được gọi là khả nghịch. Phần tử nghịch đảo của a được ký hiệu là a^{-1}.

Định nghĩa 3.12:

Phép chia của với a cho $b \bmod n$ là tích của a và $b^{-1} \bmod n$ tích này được xác định nếu b là phần tử khả nghịch.

Định lý 3.7:

Cho $a \in \mathbf{Z}_n$, khi đó a là khả nghịch nếu và chỉ nếu: $(a,n) = 1$

Ví dụ 3.9: Các phần tử khả nghịch trong $\mathbf{Z}_9$ là 1, 2, 4, 5, 7 và 8. Chẳng hạn $4^{-1} = 7$ vì $4.7 \equiv 1 \left(\mathrm{mod} 9\right)$.

Định lý 3.8:

Cho $d = (a,n)$, phương trình đồng dư $ax \equiv b \left(\mathrm{mod} n\right)$ có nghiệm x nếu và chỉ nếu $d \mid b$, trong trường hợp này có đúng d nghiệm nằm giữa 0 và $n-1$, những nghiệm này là tất cả các đồng dư theo modulo $n \mid d$.

Định lý 3.9: **(Phần dư China)**

Nếu các số nguyên $n_1,n_2,\ldots,n_k$ là nguyên tố cùng nhau từng đôi một thì hệ các phương trình đồng dư:

$$x \equiv a_1 \left(\mathrm{mod} n_1\right)$$
$$x \equiv a_2 \left(\mathrm{mod} n_2\right)$$
$$\ldots\ldots\ldots\ldots$$
$$x \equiv a_k \left(\mathrm{mod} n_k\right)$$

sẽ có nghiệm duy nhất theo modulo n $\left(n = n_1.n_2 \ldots n_k\right)$

Thuật toán Gausse

Nghiệm x của hệ phương trình đồng dư trong định lý phần dư China có thể đươc tính bằng:

$$x = \sum_{i=1}^{k} a_i N_i M_i \bmod n \tag{3.7}$$

Trong đó $N_i = n / n_i$ và $M_i = N_i^{-1} \bmod n_i$

Các tính toán này có thể được thực hiện trong $0\left(\left(\lg n\right)^2\right)$ các phép toán trên bit.

Ví dụ 3.10: Cặp phương trình đồng dư $x \equiv 3\left(\mathrm{mod}\, 7\right), x \equiv 7\left(\mathrm{mod} 13\right)$ có nghiệm duy nhất $x \equiv 59\left(\mathrm{mod} 91\right)$

Định lý 3.10:

Nếu $\left(n_1,n_2\right) = 1$ thì cặp phương trình đồng dư.

$$x \equiv a\left(\mathrm{mod} n_1\right), x \equiv a\left(\mathrm{mod} n_2\right)$$

có một nghiệm duy nhất $x \equiv a\left(\mathrm{mod} n_1,n_2\right)$

Định nghĩa 3.13:

Nhóm nhân của $\mathbf{Z}_n$ là $\mathbf{Z}_n^* = \left\{a \in Z_n \,\middle|\, \left(a,n\right) = 1\right\}$

Đặc biệt, nếu n là số nguyên tố thì $\mathbf{Z}_n^* = \left\{a \,\middle|\, 1 \le a \le n-1\right\}$

Định nghĩa 3.14:

Cấp của $\mathbf{Z}_n^*$ là số các phần tử trong $\mathbf{Z}_n^*$ (ký hiệu $\left|\mathbf{Z}_n^*\right|$)

Theo định nghĩa của hàm Phi-Euler ta thấy:

$$\left|\mathbf{Z}_n^*\right| = \varphi\left(n\right) \tag{3.8}$$

Chú ý: nếu $a \in \mathbf{Z}_n^*$ và $b \in \mathbf{Z}_n^*$ thì $a,b \in \mathbf{Z}_n^*$ và bởi vậy $\mathbf{Z}_n^*$ là đóng đối với phép nhân.

Định lý 3.11:

Cho p là một số nguyên tố:

(1) *Định lý Euler:* Nếu $a \in \mathbf{Z}_n^*$ thì $a^{\varphi(n)} \equiv 1(\mathrm{mod}\, n)$.

(2) Nếu n là tích của các số nguyên khác nhau và nếu $r \equiv s(\mathrm{mod}\, \varphi(n))$ thì $a^r \equiv a^s (\mathrm{mod}\, n)$ đối với mọi số nguyên a. Nói một cách khác khi làm việc với modulo n thì các số mũ có thể được rút gọn theo modulo $\varphi(n)$.

Định lý 3.12:

Cho p là một số nguyên tố:

(1) Định lý Ferma: Nếu $(a,p) = 1$ thì $a^{p-1} \equiv 1(\mathrm{mod}\, p)$.

(2) Nếu $r \equiv s(\mathrm{mod}\, p-1)$ thì $a^r \equiv a^s (\mathrm{mod}\, p)$ đối với mọi số nguyên a. Nói một cách khác khi làm việc với modulo của một số nguyên tố p thì các luỹ thừa có thể được rút gọn theo modulo $p-1$.

(3) Đặc biệt $a^p \equiv a(\mathrm{mod}\, p)$ với mọi số nguyên a.

Định nghĩa 3.15:

Cho $a \in \mathbf{Z}_n^*$. Cấp của a (ký hiệu là $ord(a)$) là số nguyên dương nhỏ nhất t sao cho $a^t \equiv 1(\mathrm{mod}\, n)$.

Định nghĩa 3.16:

Cho $a \in \mathbf{Z}_n^*$, $ord(a) = t$ và $a^s \equiv 1(\mathrm{mod}\, n)$ khi đó t là ước của s.

Đặc biệt $t \mid \varphi(n)$.

Ví dụ 3.11: Cho $n = 21$, khi đó $\mathbf{Z}_{21}^* = \{1,2,4,5,8,10,11,13,16,17,19,20\}$

Chú ý rằng $\varphi(21) = \varphi(7).\varphi(3) = 12 = |\mathbf{Z}_{21}^*|$. Cấp của các phần tử trong $\mathbf{Z}_{21}^*$ được nêu trong bảng sau:

Bảng 3.3. Cấp của các phần tử trong $\mathbf{Z}_{21}^*$

$a \in Z_{21}^*$	1	2	4	5	8	10	11	13	16	17	19	20
$ord(a)$	1	6	3	6	2	6	6	2	3	6	6	2

Định nghĩa 3.17:

Cho $\alpha \in \mathbf{Z}_n^*$. Nếu cấp của α là $\varphi(n)$ thì α được gọi là phần tử sinh hay phần tử nguyên thuỷ của $\mathbf{Z}_n^*$. Nếu $\mathbf{Z}_n^*$ có một phần tử sinh thì $\mathbf{Z}_n^*$ được gọi là cyclic.

Các tính chất của các phần tử sinh của $\mathbf{Z}_n^*$

(1) $\mathbf{Z}_n^*$ có phần tử sinh nếu và chỉ nếu $n = 2, 4, p^k$ hoặc là $2p^k$, trong đó p là một số nguyên tố lẻ và $k \geq 1$. Đặc biệt, nếu p là một số nguyên tố thì $\mathbf{Z}_n^*$ có phần tử sinh.

(2) Nếu α là một phần tử sinh của $\mathbf{Z}_n^*$ thì:

$$\mathbf{Z}_n^* = \{\alpha^i \bmod n \mid 0 \leq i \leq \Phi(n) - 1\} \tag{3.9}$$

(3) Giả sử rằng α là một phần tử sinh của $\mathbf{Z}_n^*$ khi đó $b = \alpha^i \bmod n$ cũng là một phần tử sinh của Z_n^* nếu và chỉ nếu $(i, \varphi(n)) = 1$. Từ đó ta rút ra rằng nếu $\mathbf{Z}_n^*$ là cyclic thì số các phần tử sinh là $\varphi(\varphi(n))$.

(4) $\alpha \in \mathbf{Z}_n^*$ là một phần tử sinh của $\mathbf{Z}_n^*$ nếu và chỉ nếu $\alpha^{\varphi(n)/p} \neq 1 (\bmod n)$ đối với mỗi nguyên tố p của $\varphi(n)$

Ví dụ 1.12: $\mathbf{Z}_{21}^*$ không là cyclic vì nó không chứa một phần tử có cấp $\Phi(21) = 12$ (Chú ý rằng 21 không thoả mãn điều kiện (1) ở trên).

$\mathbf{Z}_{25}^*$ là cyclic và có một phần tử sinh $\alpha = 2$

Định nghĩa 3.18:

Cho $a \in \mathbf{Z}_n^*$, a được gọi là thặng dư bậc hai modulo n (hay bình phương của modulo n) nếu tồn tại $x \in \mathbf{Z}_n^*$ sao cho $x^2 \equiv a (\bmod n)$. Nếu không tồn tại x như vậy thì a được gọi là thặng dư không bậc hai $\bmod n$. Tập tất cả các thặng dư bậc hai modulo n được ký hiệu là Q_n, còn tập tất cả các thặng dư không bậc hai được ký hiệu là $\overline{Q_n}$. Cần chú ý rằng theo định nghĩa $0 \notin \mathbf{Z}_n^*$. Bởi vậy $0 \notin Q_n$ và $0 \notin \overline{Q_n}$.

Định lý 3.13:

Cho p là một số nguyên tố và α là một phần tử sinh của $\mathbf{Z}_p^*$. Khi đó $a \in \mathbf{Z}_p^*$ là một thặng dư bậc hai modulo p nếu và chỉ nếu $a = \alpha^i \bmod p$, trong đó i là một số *nguyên chẵn*. Từ đó rút ra:

$$|Q_p| = \frac{(p-1)}{2} \text{ và } |\overline{Q_p}| = \frac{(p-1)}{2} \tag{3.10}$$

tức là một nửa số phần tử trong $\mathbf{Z}_p^*$ là các thặng dư bậc hai và nửa còn lại thặng dư không bậc hai.

Ví dụ 3.12: $\alpha = 6$ là một phần tử sinh của $\mathbf{Z}_{13}^*$. Các luỹ thừa của α được liệt kê ở bảng 3.4. sau đây:

Bảng 3.4.

i	1	2	3	4	5	6	7	8	9	10	11	12
$\alpha^i \bmod 13$	6	10	8	9	2	12	7	3	5	4	11	1

Bởi vậy $Q_{13} = \{1,3,4,9,10,12\}$, $\overline{Q}_{13} = \{2,5,6,7,8,11\}$

Định lý 3.14:

Cho n là tích của hai số nguyên tố lẻ khác nhau q và p, $n = pq$, khi đó $a \in \mathbf{Z}_n^*$ là một thặng dư bậc hai modulo n nếu và chỉ nếu $a \in Q_p$ và $a \in Q_q$. Điều đó dẫn tới:

$$|Q_n| = |Q_q|.|Q_p| = \frac{(p-1)(q-1)}{4}$$

và

$$|\overline{Q}_n| = \frac{3(p-1)(q-1)}{4}$$

Ví dụ 3.13: Cho $n = 21$. Khi đó :

$$Q_{21} = \{1,4,16\} \quad \overline{Q}_{21} = \{2,5,8,10,11,13,17,19,20\}$$

Định nghĩa 3.19:

Cho $a \in Q_n$, nếu $x \in Z_n^*$ thoả mãn $x^2 \equiv a \pmod{n}$ thì x được gọi là căn bậc hai của $a \bmod n$.

Định lý 3.15: (Số các căn bậc hai).

Nếu p là một số nguyên tố lẻ và $a \in Q_n$ thì a được gọi là căn bậc hai theo modulo p.

Tổng quát hơn, cho $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$, trong đó p_i là các số nguyên tố lẻ phân biệt và $e_i \geq 1$. Nếu $a \in Q_n$ thì có đúng 2^k căn bậc hai khác nhau theo modulo n.

Ví dụ 3.14:

- Các căn bậc 2 của $12 \bmod 37$ là 7 và 30.

- Các căn bậc 2 của $121 \bmod 315$ là 11, 74, 101, 151, 164, 214, 241 và 304.

3.1.4. Các thuật toán trong $\mathbf{Z}_n$

Cho n là một số nguyên dương. Các phần tử của $\mathbf{Z}_n$ sẽ được biểu thị bởi các số nguyên $Q_{21} = \{0,1,2,...,n-1\}$.

Ta thấy rằng, nếu $a,b \in Z_n$ thì

$$(a+b) \bmod n = \begin{cases} a+b & a+b<n \\ a+b-r & a+b \geq n \end{cases} \tag{3.11}$$

Bởi vậy phép cộng (và trừ) theo modulo có thể thực hiện được mà không cần phép chia dài. Phép nhân modulo của a và b có thể được thực hiện bằng cách nhân các số nguyên thông thường rồi lấy phần dư của kết quả sau khi chia cho n. Các phần tử nghịch đảo trong $\mathbf{Z}_n$ có thể được tính bằng cách dùng thuật toán Euclide mở rộng được mô tả dưới đây.

3.1.4.1. Thuật toán (Tính các nghịch đảo trong $\mathbf{Z}_n$)

VÀO: $a \in Z_n$

RA : $a^{-1} \bmod n$ (nếu tồn tại).

(1) Dùng thuật toán Euclide mở rộng để tìm các số nguyên x và y sao cho $ax+ny=d$ trong đó $d=(a,n)$.

(2) Nếu $d>1$ thì $a^{-1} \bmod n$ không tồn tại.

Ngược lại $return\ (x)$.

Phép luỹ thừa theo modulo có thể được thực hiện có hiệu quả bằng thuật toán nhân và bình phương có lặp. Đây là một thuật toán rất quan trọng trong nhiều thủ tục mật mã. Cho biểu diễn nhị phân của k là:

$\sum_{i=0}^{t} k_i 2^i$ trong đó mỗi $k_i \in \{0,1\}$ khi đó

$$a^k = \prod_{i=0}^{t} a^{k_i 2^i} = \left(a^{2^0}\right)^{k_0} \left(a^{2^1}\right)^{k_1} \ldots \left(a^{2^t}\right)^{k_t} \tag{3.12}$$

3.1.4.2. Thuật toán nhân và bình phương có lặp để lấy luỹ thừa trong $\mathbf{Z}_n$

VÀO: $a \in Z_n$ và số nguyên k, $(0 \leq k < n)$ có biểu diễn nhị phân:

$$k = \sum_{i=0}^{t} k_i 2^i$$

RA : $a^k \bmod n$

(1) Đặt $b \leftarrow 1$. Nếu $k=0$ thì $return\ (b)$

(2) Đặt $A \leftarrow a$.

(3) Nếu $k_0 = 1$ thì đặt $b \leftarrow a$.

(4) For i from 1 to t do

(4.1). Đặt $A \leftarrow A^2 \bmod n$.

(4.2). Nếu $k_i = 1$ thì đặt $b \leftarrow A\,b \bmod n$

(5) Return (b)

Ví dụ 3.15: Bảng 3.5 sau chỉ ra các bước tính toán $5^{596} \bmod 1234 = 1013$

Bảng 3.5. Tính 5^{596} mod1234

i	0	1	2	3	4	5	6	7	8	9
k_i	0	0	1	0	1	0	1	0	0	1
A	5	25	625	681	1011	369	421	779	947	925
b	1	1	625	625	67	67	1059	1059	1059	1013

Số các phép toán bit đối với phép toán cơ bản trong $\mathbf{Z}_n$ được tóm lược trong bảng 3.6 dưới đây.

Bảng 3.6. Độ phức tạp bit của các phép toán cơ bản trong Z_n

Phép toán		Độ phức tạp bit
Cộng modulo	$a+b$	$0(\lg n)$
Trừ modulo	$a-b$	$0(\lg n)$
Nhân modulo	$a.b$	$0\left((\lg n)^2\right)$
Nghịch đảo modulo	$a^{-1} \bmod n$	$0\left((\lg n)^2\right)$
Luỹ thừa modulo	$a^k \bmod n$, $k < n$	$0\left((\lg n)^3\right)$

3.1.5. Các ký hiệu Legendre và Jacobi

Ký hiệu Legendre là một công cụ hữu ích để xem xét liệu một số nguyên a có là một thặng dư bậc hai theo modulo của một số nguyên tố p hay không.

3.1.5.1. Ký hiệu Legendre

Định nghĩa 3.20:

Cho p là một số nguyên tố lẻ và a là một số nguyên. Ký hiệu Legendre $\left(\frac{a}{p}\right)$ được xác định như sau:

$$\left(\frac{a}{p}\right)=\begin{cases}0 & p \mid a\\ 1 & a \in Q_p\\ -1 & a \in \overline{Q}_p\end{cases} \tag{3.13}$$

3.1.5.2. Các tính chất của ký hiệu Legendre

Cho p là một số nguyên tố lẻ và $a,b \in \mathbf{Z}$. Khi đó ký hiệu Legendre có các tính chất sau:

(1) $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p$. Đặc biệt $\left(\frac{1}{p}\right)=1$ và $\left(-\frac{1}{p}\right)=(-1)^{(p-1)/2}$ Bởi vậy $-1 \in Q_p$ nếu $p \equiv 1 \pmod 4$ và $-1 \in \overline{Q}_p$ nếu $p \equiv 3 \pmod 4$

(2) $\left(\frac{ab}{p}\right) \equiv \left(\frac{a}{p}\right).\left(\frac{b}{p}\right)$. Bởi vậy nếu $a \in Z_p^*$ thì $\left(\frac{a^2}{p}\right)=1$.

(3) Nếu $a \equiv b \pmod p$ thì $\left(\frac{a}{p}\right)=\left(\frac{b}{p}\right)$.

(4) $\left(\frac{2}{p}\right)=(-1)^{(p^2-1)/8}$. Bởi vậy $\left(\frac{2}{p}\right)=1$ nếu $p \equiv 1$ hoặc $7 \pmod 8$ và $\left(\frac{2}{p}\right)=-1$ nếu $p \equiv 3$ hoặc $5 \pmod 8$.

(5) Luật thuận nghịch bậc 2:

Giả sử p là một số nguyên tố lẻ khác với q, khi đó:

$$\left(\frac{p}{q}\right)=\left(\frac{q}{p}\right)(-1)^{(p-1)(q-1)/4}$$

Nói một cách khác $\left(\frac{p}{q}\right)=\left(\frac{q}{p}\right)$ trừ phi cả p và q là đồng dư với $3 \pmod 4$, trong trường hợp này $\left(\frac{p}{q}\right)=-\left(\frac{q}{p}\right)$.

Dấu hiệu Jacobi là tổng quát hoá của ký hiệu Legendre đối với các số nguyên lẻ n không nhất thiết là một số nguyên tố.

3.1.5.3. Ký hiệu Jacobi

Định nghĩa 3.21:

Cho $n \geq 3$ là các số nguyên tố có phân tích $n = p_1^{e_1}.p_2^{e_2}\ldots p_k^{e_k}$. Khi đó ký hiệu Jacobi $\left(\frac{a}{n}\right)$ được định nghĩa là

$$\left(\frac{a}{n}\right)=\left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2}\ldots\left(\frac{a}{p_k}\right)^{e_k} \tag{3.14}$$

Ta thấy rằng nếu n là một số nguyên tố thì ký hiệu Jacobi chính là ký hiệu Legendre.

3.1.5.4. Các tính chất của ký hiệu Jacobi

Cho $n \geq 3$ là các số nguyên tố $a,b \in \mathbf{Z}$. Khi đó ký hiệu Jacobi có các tính chất sau:

(1) $\left(\frac{a}{n}\right)=0,1$ hoặc -1. Hơn nữa $\left(\frac{a}{n}\right)=0$ nếu và chỉ nếu $UCLN\left(a,n\right)\neq 1$.

(2) $\left(\frac{ab}{n}\right)\equiv\left(\frac{a}{n}\right).\left(\frac{b}{n}\right)$. Bởi vậy $a \in Z_n^*$ thì $\left(\frac{a^2}{n}\right)=1$

(3) $\left(\frac{a}{m.n}\right)\equiv\left(\frac{a}{m}\right).\left(\frac{a}{n}\right)$.

(4) Nếu $a \equiv b\left(\mathrm{mod}n\right)$ thì $\left(\frac{a}{n}\right)=\left(\frac{b}{n}\right)$.

(5) $\left(\frac{1}{n}\right)=1$

(6) $\left(-\frac{1}{n}\right)=\left(-1\right)^{(n-1)/2}$. Bởi vậy: $\left(-\frac{1}{n}\right)=1$ nếu $n \equiv 1\left(\mathrm{mod}\,4\right)$

$\left(-\frac{1}{n}\right)=-1$ nếu $n \equiv 3\left(\mathrm{mod}\,4\right)$

(7) $\left(\frac{2}{n}\right)=\left(-1\right)^{\left(n^2-1\right)/8}$. Bởi vậy: $\left(\frac{2}{n}\right)=1$ nếu $n \equiv 1$ hoặc $7\left(\mathrm{mod}\,8\right)$

$\left(\frac{2}{n}\right)=-1$ nếu $n \equiv 3$ hoặc $5\left(\mathrm{mod}\,8\right)$

(8) $\left(\frac{m}{n}\right)=\left(\frac{n}{m}\right)\left(-1\right)^{(m-1)(n-1)/4}$

Nói một cách khác $\left(\frac{m}{n}\right)=\left(\frac{n}{m}\right)$ trừ phi cả hai số m và n đều đồng dư với $3(\mathrm{mod}\,4)$, trong trường hợp này $\left(\frac{m}{n}\right)=-\left(\frac{n}{m}\right)$.

Từ các tính chất của ký hiệu Jacobi ta thấy rằng n lẻ và $a=2^e a_1$ trong đó a_1 là một số lẻ thì:

$$\left(\frac{a}{n}\right)=\left(\frac{2^e}{n}\right)\left(\frac{a_1}{n}\right)=\left(\frac{2}{n}\right)^e\left(\frac{n \bmod a_1}{a_1}\right)(-1)^{(a_1-1)(n-1)/4}$$

Từ công thức này ta có thể xây dựng thuật toán đệ quy sau để tính $\left(\frac{a}{n}\right)$ mà không cần phải phân tích n ra các thừa số nguyên tố .

3.1.5.5. Thuật toán tính toán ký hiệu Jacobi (và ký hiệu Legendre)

JACOBI (a,n)

VÀO: Số nguyên lẻ $n \geq 3$ số nguyên a, $(0 \leq a \leq n)$

RA : Ký hiệu Jacobi $\left(\frac{a}{n}\right)$ (Sẽ là ký hiệu Legendre khi n là số nguyên tố)

(1) Nếu $a=0$ thì $return\,(0)$

(2) Nếu $a=1$ thì $return\,(1)$

(3) Viết $a=2^e a_1$, trong đó a_1 là một số lẻ

(4) Nếu e chẵn thì đặt $s \leftarrow 1$. Ngược lại hãy đặt $s \leftarrow 1$ nếu $n=1$ hoặc $7(\mathrm{mod}\,8)$

(5) Nếu $n \equiv 3(\mathrm{mod}\,4)$ và $a_1 \equiv 3(\mathrm{mod}\,4)$ thì đặt $s \leftarrow -s$

(6) Đặt $r_1 \leftarrow n \bmod a_1$

(7) Return $(s\,JACOBI\,(n_1,a_1))$

Thuật toán trên có thời gian chạy chừng $0((\lg n)^2)$ các phép toán bit.

3.1.5.6. Nhận xét (tìm các thặng dư bậc hai theo modulo của số nguyên tố p)

Cho p là một số nguyên tố lẻ. Mặc dù đã biết rằng một nửa các phần tử trong $\mathbf{Z}_p^*$ là các thặng dư không bậc hai theo modulo p nhưng không có một thuật toán xác định theo thời gian đa thức nào được biết để tìm.

Một thuật toán ngẫu nhiên tìm một thặng dư không bậc hai là chọn ngẫu nhiên các số nguyên $a \in \mathbf{Z}_p^*$ cho tới khi số đó thoả mãn $\left(\frac{a}{p}\right) = -1$. Phép lặp đối với số được chọn trước khi tìm được một thặng dư bậc hai là 2 và bởi vậy thuật toán được thực hiện theo thời gian đa thức.

3.1.5.7. Ví dụ tính toán ký hiệu Jacobi

Cho $a = 158$ và $n = 235$. Thuật toán trên tính $\left(\frac{158}{235}\right)$ như sau:

$$\left(\frac{158}{235}\right) = \left(\frac{2}{235}\right)\left(\frac{79}{235}\right) = (-1)\left(\frac{235}{79}\right)(-1)^{78.234/4} = \left(\frac{77}{79}\right)$$
$$= \left(\frac{77}{79}\right)(-1)^{76.78/4} = \left(\frac{2}{77}\right) = -1$$

Khác với ký hiệu Legendre, ký hiệu Jacobi $\left(\frac{a}{n}\right)$ không cho biết liệu a có phải là một thặng dư bậc 2 theo modulo n hay không. Sự thực là nếu $a \in Q_n$ thì $\left(\frac{a}{n}\right) = 1$ Tuy nhiên $\left(\frac{a}{n}\right) = 1$ thì không có nghĩa là $a \in Q_n$.

3.1.5.8. Ví dụ (Các thặng dư bậc 2 và không bậc 2)

Bảng 3.7. Các ký hiệu Jacobi của các phần tử trong $\mathbf{Z}_{21}^*$

$a \in \mathbf{Z}_{21}^*$	1	2	4	5	8	10	11	13	16	17	19	20
$a^2 \bmod n$	1	4	16	4	1	16	16	1	4	16	4	1
$\left(\frac{a}{3}\right)$	1	−1	1	−1	−1	1	−1	1	1	−1	1	−1
$\left(\frac{a}{7}\right)$	1	1	1	−1	1	−1	1	−1	1	−1	−1	−1
$\left(\frac{a}{21}\right)$	1	−1	1	1	−1	−1	−1	−1	1	1	−1	−1

Bảng 1.6 liệt kê các phần tử trong $\mathbf{Z}_{21}^*$ và các ký hiệu Jacobi của chúng. Từ ví dụ trong phần c ta có $Q_{21} = \{1,4,16\}$. Ta thấy rằng $\left(\frac{5}{21}\right) = 1$ nhưng $5 \notin Q_{21}$.

Định nghĩa 3.22:

Cho $n \geq 3$ là các số nguyên tố lẻ và cho $J_n = \left\{a \in Z_n^* \middle| \left(\frac{a}{n}\right) = 1\right\}$ tập các thặng dư giả bậc 3 theo modulo *n* (Ký hiệu $\hat{Q}_n$) được định nghĩa là tập $J_n - Q_n$.

Định lý 3.16:

Cho $n = pq$ là tích của hai số nguyên tố lẻ khác nhau. Khi đó:

$$\left|Q_n\right| = \left|\tilde{Q}_n\right| = (p-1)(q-1)/4 \tag{3.15}$$

tức là một nửa các phần tử trong J_n là các thặng dư giả bậc hai.

3.1.6. Các số nguyên Blum

Định nghĩa 3.23:

Số nguyên Blum là một hợp số có dạng $n = pq$, trong đó p và q là các số nguyên tố khác nhau và thoả mãn:

$$\begin{aligned} p &\equiv 3 \bmod 4 \\ q &\equiv 3 \bmod 4 \end{aligned} \tag{3.16}$$

Định lý 3.17:

Cho $n = pq$ là một số nguyên Blum và cho $a \in Q_n$. Khi đó a có đúng 4 căn bậc hai modulo n và chỉ có một số nằm trong Q_n .

Định nghĩa 3.24:

Cho n là một số nguyên Blum và cho $a \in Q_n$. Căn bậc hai duy nhất của a nằm trong Q_n được gọi là căn bậc hai chính $a \bmod n$.

3.1.7. Ví dụ (Số nguyên Blum)

Đối với số nguyên Blum $n = 21$. Ta có $J_n = \{1,4,5,16,17,20\}$ và $\tilde{Q}_n = \{5,17,20\}$. Bốn căn bậc 2 của $a = 4$ là 2, 5, 16 và 19, trong đó chỉ có 16 là cũng nằmg trong Q_n . Bởi vậy 16 là căn bậc 2 chính của $4 \bmod 21$.

Định lý 3.18:

Nếu $n = pq$ là một số nguyên Blum thì ánh xạ.

$f : Q_n \to Q_n$ được xác định bởi $f(x) = x^2 \bmod n$ là một phép hoán vị.

Ánh xạ ngược của f là: $f^{-1}(x) = x^{((p-1)(q-1)+4/8)} \bmod n$.

3.2. GIỚI THIỆU VỀ MẬT MÃ KHOÁ CÔNG KHAI

Trong mô hình mật mã cổ điển trước đây mà hiện nay đang được nghiên cứu Alice (người gửi) và Bob (người nhận) chọn một cách bí mật khoá *K*. Sau đó dùng *K* để tạo luật mã hoá e_k và luật giải mã d_k. Trong hệ mật này d_k hoặc giống e_k hoặc dễ dàng nhận được từ nó (ví dụ trong hệ DES quá trình giải mã hoàn toàn tương tự như quá trình mã nhưng thủ tục khoá ngược lại). Các hệ mật thuộc loại này được gọi là hệ khoá bí mật, nếu để lộ e_k thì làm cho hệ thống mất an toàn.

Nhược điểm của hệ mật này là nó yêu cầu phải có thông tin trước về khoá *K* giữa Alice và Bob qua một kênh an toàn trước khi gửi một bản mã bất kỳ. Trên thực tế điều này rất khó đảm bảo. Chẳng hạn khi Alice và Bob ở cách xa nhau và họ chỉ có thể liên lạc với nhau bằng thư tín điện tử (E.mail). Trong tình huống đó Alice và Bob không thể tạo một kênh bảo mật với giá phải chăng.

Ý tưởng xây dựng một hệ mật khoá công khai (hay dùng chung) là tìm một hệ mật không có khả năng tính toán để xác định d_k khi biết e_k. Nếu thực hiện được như vậy thì quy tắc mã e_k có thể được công khai bằng cách công bố nó trong một danh bạ (bởi vậy nên có thuật ngữ *hệ mật khoá công khai*). Ưu điểm của hệ mật khoá công khai là ở chỗ Alice (hoặc bất kỳ ai) có thể gửi một bản tin đã mã cho Bob (mà không cần thông tin trước về khoá mật) bằng cách dùng mật mã công khai e_k. Người nhận A sẽ là người duy nhất có thể giải được bản mã này bằng sử dụng luật giải bí mật d_k của mình.

Có thể hình dung hệ mật này tương tự như sau. Alice đặt một vật vào một hộp kim loại và rồi khoá nó lại bằng một khoá số do Bob để lại. Chỉ có Bob là người duy nhất có thể mở được hộp vì chỉ có anh ta mới biết tổ hợp mã của khoá số của mình.

Ý tưởng về một hệ mật khoá công khai được Diffie và Hellman đưa ra vào năm 1976. Còn việc hiện thực hoá nó thì do Rivesrt, Shamir và Adleman đưa ra lần đầu tiên vào năm 1977, họ đã tạo nên hệ mật nổi tiếng RSA (sẽ được nghiên cứu trong chương này).

Một chú ý quan trọng là một hệ mật khoá công khai không bao giờ có thể đảm bảo được độ mật tuyệt đối (an toàn vô điều kiện). Sở dĩ như vậy vì đối phương khi nghiên cứu một bản mã, anh ta có thể *mã lần lượt các bản tin rõ* bằng luật mã hoá công khai e_k cho tới khi anh ta tìm được bản rõ duy nhất M đảm bảo $C = e_k(M)$. Bản rõ này chính là kết quả giải mã. Bởi vậy, ta chỉ nghiên cứu độ mật về mặt tính toán của các hệ mật này.

Một khái niệm có ích khi nghiên cứu hệ mật khoá công khai là khái niệm về *hàm cửa sập một chiều*. Ta sẽ định nghĩa khái niệm này một cách không hình thức.

Hàm mã khoá công khai e_k của Bob phải là một hàm dễ tính toán. Song việc tìm hàm ngược (hàm giải mã) rất khó khăn (đối với bất kỳ ai không phải là Bob). Đặc tính dễ tính toán hàm ngược thường được gọi là đặc tính một chiều. Bởi vậy điều kiện cần thiết là e_k phải là hàm một chiều (tính thuận đơn giản, nhưng tính ngược rất phức tạp).

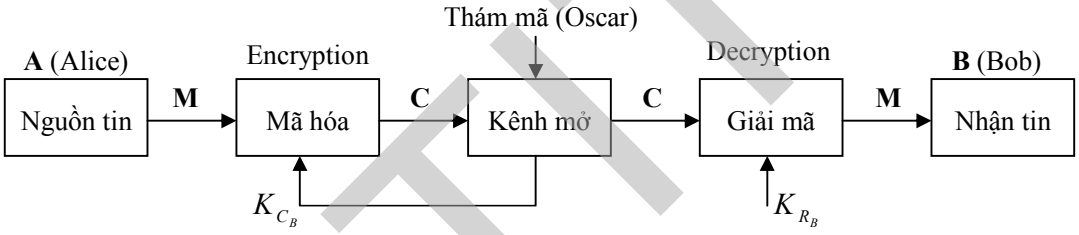
Các hàm một chiều đóng vai trò quan trọng trong mật mã học, chúng rất quan trọng trong các hệ mật khoá công khai và trong nhiều lĩnh vực khác. Đáng tiếc là mặc dù có rất

nhiều hàm được coi là hàm một chiều nhưng cho đến nay vẫn không tồn tại một hàm nào có thể chứng minh được là hàm một chiều.

Sau đây là một ví dụ về một hàm được coi là hàm một chiều. Giả sử n là tích của hai số nguyên tố lớn p và q, giả sử b là một số nguyên dương. Khi đó ta xác định ánh xạ $f: \mathrm{Z}_n \to \mathrm{Z}_n$ là $f(x) = x^b \bmod n$ (với b và n đã được chọn thích hợp thì đây chính là hàm mã RSA, sau này ta sẽ nói nhiều hơn về nó).

Để xây dựng một hệ mật khoá công khai thì việc tìm được một hàm một chiều vẫn chưa đủ. Ta không muốn e_k là hàm một chiều đối với Bob vì anh ta phải có khả năng giải mã các bản tin nhận được một cách hiệu quả. Điều cần thiết là Bob phải có một cửa sập chứa thông tin bí mật cho phép dễ dàng tìm hàm của e_k. Như vậy Bob có thể giải mã một cách hữu hiệu vì anh ta có một hiểu biết tuyệt mật nào đó về K. Bởi vậy một hàm được gọi là cửa sập một chiều nếu nó là một hàm một chiều và nó trở nên dễ tính ngược nếu biết một cửa sập nhất định.

3.3. SƠ ĐỒ CHỨC NĂNG CỦA HỆ MẬT KHÓA CÔNG KHAI



Hình 3.1. Sơ đồ chức năng hệ mật khóa công khai

Sơ đồ truyền tin bí mật từ A đến B sử dụng mật mã khóa công khai được mô ta trong Hình 3.1, trong đó:

+ M : bản tin rõ

+ C : Bản mã

+ K_{C_B} là khóa công khai của B (khóa mã hóa) được lấy trên kênh mở.

+ K_{R_B} là khóa bí mật của B (Khóa giải mã).

Hàm *mã hóa* là một ánh xạ 1:1:

$$C = E\left(M, K_{C_B}\right) \tag{3.17}$$

Và hàm *giải mã*:

$$M = E^{-1}\left(C, K_{R_B}\right) \tag{3.18}$$

Theo sơ đồ của hệ mật khóa công khai ta thấy các ưu điểm là:

- Không cần hai khóa bí mật.
- Không cần kênh an toàn riêng

- Biết khóa mã hóa trên kênh mở nhưng rất khó giải mã, tức là biết K_{C_B} nhưng rất khó suy ra được K_{R_B} (độ khó ở đây chính là độ phức tạp tính toán hoặc tài nguyên và thời gian tính toán)

Các nghiên cứu trên thế giới từ năm 1976 cho đến nay đã đưa ra được 5 bài toán một chiều như sau:

- Bài toán logarit rời rạc
- Bài toán phân tích thừa số (gắn với hệ mật nổi tiếng RSA)
- Bài toán xếp ba lô
- Bài toán mã sửa sai
- Bài toán xây dựng hệ mật trên đường cong elliptic.

Trong phần tiếp theo dưới sẽ lần lượt tìm hiểu từng bài toán một chiều và ứng dụng trong các hệ mật liên quan.

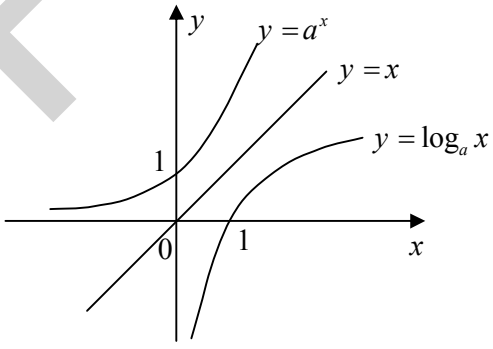
3.4. BÀI TOÁN LOGARIT RỜI RẠC VÀ CÁC HỆ MẬT LIÊN QUAN

3.4.1. Bài toán logarit rời rạc

3.4.1.1. Bài toán logarit trên trường số thực R

+ *Bài toán thuận*: Hàm số $y = a^x$ với $a, x \in R$, việc tính toán hàm mũ này có thể được thực hiện dễ dàng bằng thuật toán nhân và bình phương.

+ *Bài toán ngược*: như ta đã biết phép tính ngược của hàm mũ chính là hàm logarit $y = \log_a x$, việc tính toán hàm ngược logarit này sẽ khó khăn hơn nhiều so với hàm thuận. Tuy nhiên, cả hai phép hãm mũ và logarit đều là các hàm đồng biến cho nên có thể xác định giá trị tương đối của hàm logarit được (như Hình 3.2).



Hình 3.2. Đồ thị hàm $y = a^x$ và $y = \log_a x$

Một số tính chất của hàm logarit.

+ $y = \log_a bc = \log_a b + \log_a c$

+ $y = \log_a \frac{b}{c} = \log_a b - \log_a c$

+ $\log_a 1 = 0$

+ $y = \log_a x^{-1} = -\log_a x$

3.4.1.2. Bài toán logarit trên trường hữu hạn

Xét vành $\mathbf{Z}_p$ với p là số nguyên tố. Theo định lý 2.7 nếu p là nguyên tố thì $¢_p$ là một trường ($¢_p = \mathrm{GF}(2)$).

Tập tất cả các phần tử khác không của trường sẽ tạo nên một nhóm nhân xyclic $¢_p^*$.

$¢_p^* = ¢_p / \{0\} = \{1, 2, ..., p-1\}$

+ ***Bài toán thuận:*** $\boxed{y = a^x \bmod p,\ (a, x \in ¢_p^*)}$

Ví dụ 3.16: Xét $p = 19, a = 2$ ta có các giá trị $y = a^x$ như trong Bảng 3.8

Bảng 3.8. Các giá trị của $y = 2^x \bmod 19$ trên $¢_{19}^*$

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^x	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1

Chú ý:

+ *Nếu a là một phần tử nguyên thủy thì a^x sẽ đi qua tất cả các phần tử của nhóm.*
+ *Nếu a là phần tử nguyên thủy thì a^i cũng là nguyên thủy với $(i, p-1) = 1$ (p là số nguyên tố).*

Trong ví dụ 3.16 các giá trị của i thỏa mãn $(i, 18) = 1$ là $i = (1, 5, 7, 11, 13, 17)$. Số lượng các giá trị của i bằng giá trị hàm $\varphi(p-1)$.

$$N_i = \varphi(p-1) = \varphi(18) = 6$$

Cách tính hàm Phi-Euler φ như trình bày tại mục 1.1.12.

Như vậy trong nhóm $¢_{19}^*$ có 6 phần tử nguyên thủy:

$$2 = 2^1;\quad 13 = 2^5;\quad 14 = 2^7;\quad 15 = 2^{11};\quad 3 = 2^{13};\quad 10 = 2^{17} \qquad (3.19)$$

Các phần tử nguyên thủy này tạo thành các cặp nghịch đảo như sau:

$(2, 10) \leftrightarrow 2 = 10^{-1}$

$(13, 3) \leftrightarrow 13 = 3^{-1}$

$(14, 15) \leftrightarrow 14 = 15^{-1}$

+ ***Bài toán ngược:*** $\boxed{y = \log_a x,\ (a, x \in ¢_p^*)}$

Từ bảng 3.8 ta tính được hàm ngược $\log_2 x$ như trong bàng 3.9.

Bảng 3.9. Giá trị $\log_2 x \pmod{19}$ trên $\not{c}^*_{19}$

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^x	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
$\log_2 x$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

Chú ý: vì $2^{18} = 1$ nên $\log_2 1 = 18$.

Một số tính chất của hàm logarit rời rạc.

+ $y = \log_a bc = (\log_a b + \log_a c) \bmod p - 1$

+ $y = \log_a \frac{b}{c} = (\log_a b - \log_a c) \bmod p - 1$

+ $\log_{a^{-1}} x = -\log_a x = p - 1 - \log_a x$

+ $\log_a 1 = 0 = p - 1$ (coi $0 = p - 1$)

Nhận xét: Từ hai Bảng 3.8 và Bảng 3.9 ta thấy hai hàm thuận và ngược đều không phải hảm đồng biến, khi biết bài toán thuận thì mời tim được bài toán ngược. Do đó việc giải bài toán ngược giống bài toán vét cạn, phải thử lần lượt các trường hợp.

Việc xác định logarit của một phần tử bất kỳ trong trường là bài toán khó giải.

+ Bài toán logarit rời rạc

Cho $\not{c}^*_p$ với p là số nguyên tố, α là một phần tử nguyên thủy $\alpha \in \not{c}^*_p$.

Yêu cầu tìm $y = \log_\alpha x$ với $\alpha, x \in \not{c}^*_p$.

Nhận xét: $\forall x \in \not{c}^*_p$ thì:

- Bài toán có nghiệm khi α là phần tử nguyên thủy.

- Bài toán có thể không có nghiệm khi α bất kỳ.

Ví dụ 3.17: Với trường hợp $p = 19$ ta đa tính được 6 phần tử nguyên thủy như trong (3.19) ta sẽ đi tính bài toan logarit rời rạc với cơ số là 6 phần tử nguyên thủy này.

Tuy nhiên ta có thể áp dụng tính chất của hàm logarit rời rạc:

$\log_{a^{-1}} x = -\log_a x = p - 1 - \log_a x$, hay $\log_{a^{-1}} x + \log_a x = p - 1$

để tính logarit với cơ số là các cặp số nghịch đảo.

Tức là $(2,10)$ là cặp số nghịch đảo, khi đó $\log_{10} x = p - 1 - \log_2 x = 18 - \log_2 x$ .Tương tự $(13,3)$ và $(14,15)$ là các cặp nghịch đảo nên $\log_3 x = 18 - \log_{13} x$ và $\log_{15} x = 18 - \log_{14} x$.

Với quy tắc như thế có thể tính được các giá trị logarit như trong Bảng 3.10.

Bảng 3.10. Bài toán logarit rời rạc trên $\mathbb{Z}_{19}^*$

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^x	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
$\log_2 x$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9
$\log_{10} x$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9
13^x	13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
$\log_{13} x$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9
$\log_3 x$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9
14^x	14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
$\log_{14} x$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	4	9
$\log_{15} x$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	14	9

Có thể tính 13^x thông qua 2^x, ta thấy rằng $13 = 2^5$ do đó $13^x = 2^{5x}$, tương tự như thế $14^x = 2^{7x}$.

3.4.2. Một số hệ mật xây dựng trên bài toán logarit rời rạc

3.4.2.1. Trao đổi và thỏa thuận khóa Diffie-Hellman

Giả sử A và B muốn liên lạc sử dụng hệ mật khoá bí mật. Để thoả thuận một khoá K chung cho cả hai bên qua một kênh không an toàn mà không ai khác có thể biết được, A và B có thể dùng thủ tục thoả thuận khoá Diffie -Hellman sau:

Chọn trước một số nguyên tố p thích hợp và một phần tử nguyên thủy $\alpha \in \mathbb{Z}_p^*$.

Cặp số (p, α) tạo thành khóa công khai.

Quá trình trao đổi khóa được thực hiện như sau:

A	**B**
+ A chọn một số nguyên x bí mật thỏa mãn $1 < x < p-1$ và tính:	+ B chọn một số nguyên y bí mật thỏa mãn $1 < y < p-1$ và tính:
$\boxed{\alpha^x \bmod p}$ Gửi cho B →	← Gửi cho A $\boxed{\alpha^y \bmod p}$
Sau đó A gửi giá trị này cho B.	Sau đó B gửi giá trị này cho A.
+ A nhận $\alpha^y \bmod p$ và tính ra khóa dùng chung:	+ B nhận $\alpha^x \bmod p$ và tính ra khóa dùng chung:

$K = (a^y)^x \bmod p = \alpha^{xy} \bmod p$	$K = (a^x)^y \bmod p = \alpha^{xy} \bmod p$

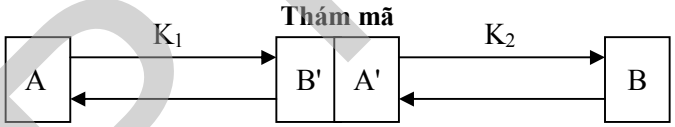
Ví dụ 3.18: Giả sử A và B chọn $p = 19$ và $\alpha = 2$

A	B
+ A chọn một số nguyên $x = 3$ và tính:	+ B chọn một số nguyên $y = 5$ và tính:
$2^3 \bmod 19 = 8$, sau đó A gửi 8 cho B	$2^5 \bmod 19 = 13$, B gửi giá trị 13 cho A.
+ A nhận 13 và tính ra khóa dùng chung:	+ B nhận 8 và tính ra khóa dùng chung:
$K = 13^3 \bmod 19 = 12$	$K = 8^5 \bmod 19 = 12$

Nhận xét:

+ Thám mã biết (α, p) nhưng không biết x, y, nếu muốn biết khóa dùng chung $K = \alpha^{xy} \bmod p$ thì thám mã phải giải bài toán logarit (bài toán ngược) để tìm x và y.

+ Tuy nhiên việc thỏa thuận theo phương thức này sẽ chịu phép tấn công "*kẻ đứng giữa*" (Man in the midle). Thám mã đứng giữa sẽ giả mạo B để thỏa thuận khóa K_1 dùng chung với A, đồng thời thám mã giả danh A để thỏa thuận khóa K_2 dùng chung với B. Thám mã liên lạc với A bằng khóa K_1, giải mã để lấy cắp thông tin của A, sau đó lại mã hóa thông tin của A bằng khóa K_2 để liên lạc với B (và tương tự như thế theo chiều từ B đến A). Hai bên A và B vẫn nhận đúng thông tin tưởng là liên lạc đúng với nhau, nhưng thực tế là liên lạc với thám mã. Đây là điểm yếu của phương pháp thỏa thuận khóa kiểu này, để khắc phục người ta sử dụng các phương pháp xác thực (sẽ được trình bày sau).



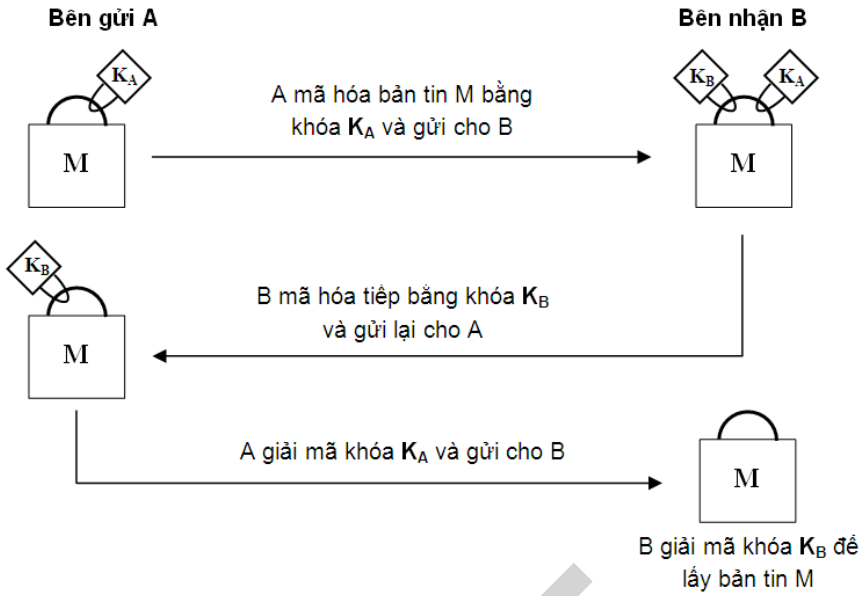
Hình 3.3. Phép tấn công kẻ đứng giữa

3.4.2.2. Hệ mật Omura-Massey

a) Sơ đồ hệ mật Omura – Massey

Ý tưởng của hệ mật Omura – Massey được mô tả trong Hình 3.4.

Hai bên liên lạc A và B sử dụng hai khóa bí mật khác nhau là K_A và K_B, đầu tiên bản tin M được A mã hóa bằng khóa bí mật của A là K_A và gửi bản mã cho B, tất nhiên trên đường truyền thám mã không thể có được bản tin M vì không có khóa K_A. Bên B nhận được bản mã lại thực hiện mã hóa một lần nữa bằng khóa K_B và gửi lại bản mã mới cho A. Khi A nhận lại bản mã thì tiến hành giải mã bằng khóa K_A, lúc này bản tin M chỉ được mã hóa bằng khóa K_B. A gửi bản mã này cho B, bên B nhận được và tiến hành giải mã bằng khóa K_B để lấy lại bản tin M.



Hình 3.4. Lưu đồ hệ mật Omura - Massey

Hệ mật này về cơ bản rất an toàn, tuy nhiên nó có một nhược điểm là bản mã truyền giữa A và B phải được thực hiện 3 lần, tức là tốc độ mã thấp hay dung lượng thông tin cần truyền sẽ tăng lên.

b) Hệ mật Omura – Massey xây dựng trên bài toán logarit rời rạc

Hai bên liên lạc A và B chọn trước $\not\in_p$, với p là số nguyên tố.

Tạo khóa: Hai bên liên lạc A và B tạo cho mình khóa bí mật như sau:

A chọn m,n ngẫu nhiên thỏa mãn $mn \equiv 1 \bmod p-1$. Khóa bí mật của A là m,n

B chọn u,v ngẫu nhiên thỏa mãn $uv \equiv 1 \bmod p-1 \rightarrow$ Khóa bí mật của B là u,v

Giả sử A cần gửi một bản tin M cho B, quá trình truyền tin bảo mật sử dụng hệ mật Omura – Massey được mô tả như Hình 3.5:

$\mathbf{A}_{(m,n)}$	$\mathbf{B}_{(u,v)}$
+ A tính $\boxed{M^m \bmod p}$ và gửi cho B $\longrightarrow$	+ B nhận $M^n \bmod p$ và tính
	$\boxed{(M^m)^u \bmod p}$
	B gửi giá trị này cho A
+ A nhận $(M^m)^u \bmod p$ và tính:	
$\boxed{(M^{mu})^n \bmod p = M^u \bmod p}$ và gửi cho B $\longrightarrow$	+ B nhận $M^u \bmod p$ và giải mã ra bản tin M: $\boxed{(M^u)^v \bmod p = M}$

Hình 3.5. Mô tả hệ mật Omura – Massey sử dụng bài toán logarit rời rạc.

CHƯƠNG 4. HÀM BĂM, XÁC THỰC VÀ CHỮ KÝ SỐ

4.1. CÁC HÀM BĂM VÀ TÍNH TOÀN VẸN CỦA DỮ LIỆU

4.1.1. Khái niệm về hàm băm

Các hàm băm đóng vai trò cơ bản trong mật mã hiện đại. Hàm băm sẽ tạo ra một đầu ra có độ dài cố định từ bản tin đầu vào. Đầu ra này được định nghĩa là *mã băm* (kết quả băm, giá trị băm).

Nói một cách chính xác hơn, hàm băm h sẽ tạo ra ánh xạ các xâu bit có độ dài hữu hạn tuỳ ý thành các xâu bit có độ dài n cố định.

Xét xâu bit $x \in (Z_2)^n$ và một ánh xạ:

$$h : x \to y \in (Z_2)^m \tag{4.1}$$

Trong đó: $n > m$ nhận giá trị bất kỳ, còn $m = const$

Hàm $y = h(x)$ được gọi là hàm băm (hay mã băm, tóm lược thông báo)

Hàm băm h là một ánh xạ đầu ra có độ dài m cố định $h : D \to R$ và $|D| > |R|$ điều này có nghĩa là không thể tránh khỏi các va chạm (tức là cùng một giá trị đầu ra có thể có nhiều bộ giá trị vào khác nhau). Nếu hàm h là ngẫu nhiên theo nghĩa tất cả các đầu ra là đồng xác suất thì có chừng 2^{n-m} các đầu vào ánh xạ tới mỗi đầu ra (n : số bit đầu vào, m : số bit đầu ra, $n > m$) và 2 đầu vào được chọn ngẫu nhiên sẽ có cùng đầu ra với xác suất 2^{-m} (không phụ thuộc vào n).

Ý tưởng cơ bản của việc sử dụng các hàm băm trong mật mã là sử dụng chúng như một ảnh biểu diễn rút gọn (đôi khi còn được gọi là vết, dấu tay số hay tóm lược thông báo) của một xâu vào và có thể được dùng như thể nó chính là xâu vào đó.

Các hàm băm được dùng cho các sơ đồ chữ ký số kết hợp với việc đảm bảo tính toàn vẹn của dữ liệu, khi đó bản tin trước hết được băm và rồi giá trị băm (được xem như đại diện cho bản tin) sẽ được ký thay cho vị trí bản tin gốc.

Một lớp các hàm băm được gọi là *các mã xác thực thông báo* (MAC - Message Authentication Codes) sẽ cho phép xác thực thông báo bằng kỹ thuật đối xứng (mật mã cổ điển).

Các thuật toán MAC sử dụng 2 đầu vào (bao gồm bản tin và một khoá bí mật) để tạo ra một đầu ra có kích cỡ cố định (n bit) với ý đồ đảm bảo rằng nếu không biết khoá thì việc tạo ra cùng một đầu ra là không khả thi. MAC có thể được dùng để đảm bảo tính toàn vẹn của dữ liệu, xác thực tính nguyên bản của số liệu cũng như định danh trong sơ đồ mật mã cổ điển.

Một ứng dụng điển hình của hàm băm (không dùng khoá) để đảm bảo tính toàn vẹn của dữ liệu có thể được mô tả như sau:

Giá trị băm tương ứng với một bản tin riêng x sẽ được tính ở thời điểm T_1 . Tính toàn vẹn của giá trị băm này (chứ không phải là bản thân bản tin) sẽ được bảo vệ theo một cách

nào đó. Ở thời điểm tiếp theo sau T_2 phép kiểm tra sau sẽ được tiến hành để xác định xem liệu thông báo có bị sửa đổi hay không, tức là xem liệu bản tin x' có giống bản tin gốc hay không. Giá trị băm của x' sẽ được tính toán và so sánh với giá trị băm đã được bảo vệ, nếu chúng bằng nhau thì bên thu sẽ chấp nhận rằng x và x' là như nhau và như vậy có nghĩa là bản tin đã không bị sửa đổi. Như vậy vấn đề đảm bảo tính vẹn toàn của một bản tin lớn sẽ được gui về đảm bảo cho một giá trị băm có kích cỡ cố định (và nhỏ).

Ứng dụng trên thường được gọi là *mã phát hiện sự sửa đổi (MDC - Manipulation Detection Codes).*

4.1.2. Các định nghĩa, tính chất cơ bản và phân loại hàm băm

4.1.2.1. Định nghĩa hàm băm

Định nghĩa 4.1: Hàm băm là một ánh xạ $h(x)$ thỏa mãn hai tính chất:

a) Tính chất nén: h sẽ ánh xạ một đầu vào x có độ dài bit hữu hạn tuỳ ý tới một đầu ra $h(x)$ có độ dài bit m hữu hạn.

b) Tính chất dễ dàng tính toán: Với h cho trước và một đầu vào x , có thể dễ dàng tính được $h(x)$.

4.1.2.2. Một số tính chất của các hàm băm không có khoá

Giả sử h là một hàm băm không có khoá, x và x' là các đầu vào và y và y' là các đầu ra tương ứng. Ngoài hai tính chất cơ bản trên hàm băm mật mã còn có 3 tính chất sau:

a) Tính khó tính toán nghịch ảnh:

Đối với hầu hết các đầu ra được xác định trước, không có khả năng tính toán để tìm một đầu vào bất kỳ mà khi băm sẽ cho ra đầu ra tương ứng (Tức là tìm một nghịch ảnh x sao cho $h(x)=y$ với y cho trước và không biến đầu vào tương ứng).

b) Khó tìm nghịch ảnh thứ hai:

Không có khả năng tính toán để tìm một đầu vào đã cho trước: Tức là với x cho trước phải tìm $x' \neq x$ sao cho $h(x)=h(x')$

c) Tính khó va chạm. Không có khả năng về tính toán để tìm hai đầu vào khác nhau bất kỳ $x' \neq x$ để $h(x)=h(x')$.

4.1.2.3. Định nghĩa hàm băm một chiều (OWHF - oneway hash function)

Định nghĩa 4.2: Định nghĩa hàm băm một chiều (OWHF - oneway hash function) là một hàm băm (ngoài hai tính chất cơ bản) có tính chất bổ sung là:

- Khó tìm nghịch ảnh
- Khó tìm nghịch ảnh thứ hai.

4.1.2.4. Định nghĩa hàm băm khó va chạm (CRHF: Collision resistant HF)

Định nghĩa 4.3: Định nghĩa hàm băm khó va chạm (CRHF: Collision resistant HF) là một hàm băm (ngoài hai tính chất cơ bản) có tính chất bổ sung là:

- Khó tìm nghịch ảnh thứ hai
- Khó và chạm

4.1.2.5. Chú ý về các thuật ngữ

Khó tìm nghịch ảnh ≡ Một chiều

Khó tìm nghịch ảnh thứ hai ≡ Khó va chạm yếu.

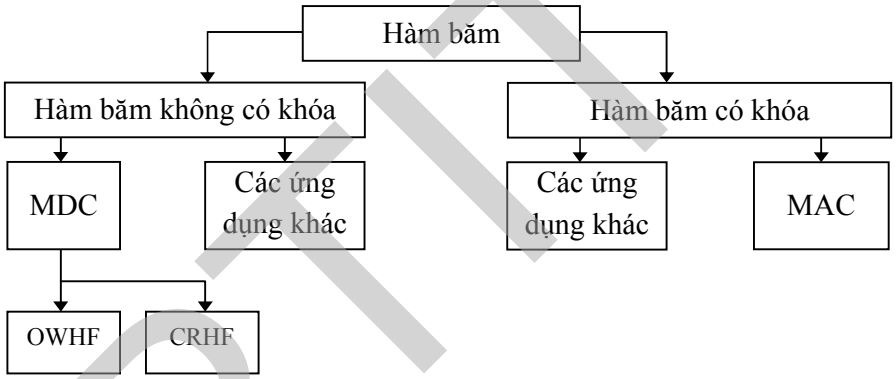
Khó va chạm ≡ Khó va chạm mạnh

OWHF ≡ Hàm băm một chiều yếu.

CRHF ≡ Hàm băm một chiều mạnh.

Ví dụ: r bit kiểm tra của một mã xyclic (n,k) với $k > r$ có thể coi là một hàm băm thoả mãn hai tính chất cơ bản (dễ tính toán và nén). Tuy nhiên nó không thoả mãn tính chất khó tìm nghịch ảnh thứ hai.

4.1.2.6. Phân loại các hàm băm mật mã và ứng dụng



Hình 4.1. Phân loại hàm băm

MDC: Manipullation Detection Code – Mã phát hiện sửa đổi

MAC: Message Authentication Code – Mã xác thực thông báo.

4.1.3. Các hàm băm không có khóa (MDC)

Các hàm băm không khóa dựa trên mật mã khối

Định nghĩa 4.4:

Mật mã khối (n,r) là một mã khối xác định một hàm khả nghịch từ các bản rõ n bit sang các bản mã n bit bằng cách sử dụng một khoá r bit. Nếu E là một phép mã hoá như vậy thì $E_k(x)$ ký hiệu cho phép mã hoá x bằng khoá k.

Định nghĩa 4.5:

Cho h là một hàm băm có lặp h được xây dựng từ một mật mã khối với hàm nén f thực hiện s phép mã hoá khối để xử lý từng khối bản tin n bit. Khi đó tốc độ của là $1/s$.

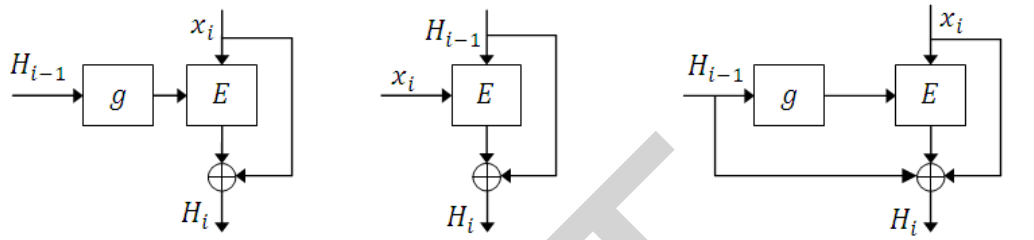
4.1.3.1. MDC độ dài đơn

Ba sơ đồ trong Hình 4.2 dưới đây có liên quan chặt chẽ với các hàm băm độ dài đơn, xây dựng trên các mật mã khối. Các sơ đồ này có sử dụng các thành phần được xác định trước như sau:

Một mật mã khối n bit khởi sinh E_k được tham số hoá bằng một khoá đối xứng k.

Một hàm g ánh xạ n bit vào thành khoá k sử dụng cho E (Nếu các khoá cho E cũng có độ dài n thì g có thể là hàm đồng nhất).

Một giá trị ban đầu cố định IV (Init Vector) thích hợp để dùng với E.



a) Sơ đồ Matyas-Mayer–Oseas *b) Sơ đồ Davies-Mayer* *c) Sơ đồ Miyaguchi - Preneel*

Hình 4.2. Một số phương pháp xây dựng hàm băm đơn

Thuật toán băm Matyas - Mayer – Oseas (M-M-O)

VÀO: Xâu bit x

RA : Mã băm n bit của x

Đầu vào x được phân chia thành các khối n bit và được độn nếu cần thiết nhằm tạo khối cuối cùng hoàn chỉnh. Ta được t khối n bit: $x = (x_1, x_2, ..., x_t)$. Phải xác định trước một giá trị ban đầu n bit (ký hiệu IV).

Đầu ra là H_t được xác định như sau:

$$\begin{cases} H_0 = IV \\ H_i = E_{g(H_i - 1)}(x_i) \oplus x_i, \ 1 \le i \le t \end{cases}$$

Thuật toán băm Davies - Mayer (D-M)

VÀO: Xâu bit x

RA : Mã băm n bit của x

Đầu vào x được phân thành các khối k bit (k là kích thước khoá) và được độn nếu cần thiết để tạo khối cuối cùng hoàn chỉnh. Biểu thị thông báo đã độn thành t khối k bit: $x = (x_1, x_2, ..., x_t)$. Xác định trước một giá trị ban đầu n bit (ký hiệu IV).

Đầu ra là H_t được xác định như sau:

$$\begin{cases} H_0 = IV \\ H_i = E_{x_i}(H_i) \oplus H_{i-1}, \ 1 \le i \le t \end{cases}$$

Thuật toán băm Miyaguchi - Preneel

Sơ đồ này tương tự như M-M-O ngoại trừ H_{i-1} (đầu ra ở giai đoạn trước) được cộng modulo 2 với tín hiệu ra ở giai đoạn hiện thời. Như vậy:

$$\begin{cases} H_0 = IV \\ H_i = E_{g(H_i-1)}(x_i) \oplus x_i \oplus H_{i-1},\ 1 \le i \le t \end{cases}$$

Nhận xét: Sơ đồ D-M có thể coi là sơ đồ đối ngẫu với sơ đồ M - M - O theo nghĩa x_i và H_{i-1} đổi lẫn vai trò.

4.1.3.2. MDC độ dài kép: MDC-2 và MDC- 4

MDC-2 và MDC- 4 là các mã phát hiện sự sửa đổi yêu cầu tương ứng là 2 và 4 phép toán mã hoá khối trên mỗi khối đầu vào hàm băm. Chúng sử dụng 2 hoặc 4 phép lặp của sơ đồ M - M - O để tạo ra hàm băm có độ dài kép. Khi dùng DES chúng sẽ tạo ra mã băm 128 bit. Tuy nhiên trong cấu trúc tổng quát có thể dùng các hệ mật mã khối khác MDC-2 và MDC4 sử dụng các thành phần xác định như sau:

DES được dùng làm mật mã khối E_k có đầu vào/ ra 64 bit và được tham số hoá bằng khoá k 56 bit.

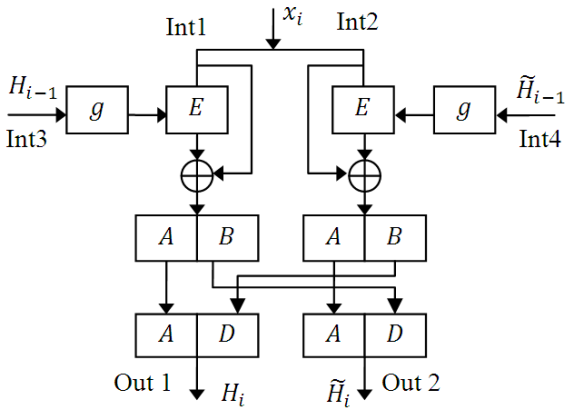
Hai hàm g và $\tilde{g}$ ánh xạ các giá trị 64 bit U thành các khoá DES 56 bit như sau:

Cho $U = u_1 u_2 .. u_{64}$, xoá mọi bit thứ 8 bắt đầu từ u_8 và đặt các bit thứ 2 và thứ 3 về "10" đối với g và "01" đối với $\tilde{g}$.

$$g(U) = u_1 10 u_4 u_5 u_6 u_7 u_9 u_{10} .. u_{63}$$
$$\tilde{g}(U) = u_1 01 u_4 u_5 u_6 u_7 u_9 u_{10} .. u_{63}$$

Đồng thời điều này cũng phải đảm bảo rằng chúng không phải là các khoá DES yếu hoặc nửa yếu vì các khoá loại này có bit thứ hai bằng bit thứ ba. Đồng thời điều này cũng đảm bảo yêu cầu bảo mật là $g(IV) \neq \tilde{g}(IV)$.

Thuật toán MDC -2 có thể được mô tả theo sơ đồ sau:



Hình 4.3. Sơ đồ hàm băm MDC – 2

Thuật toán MDC - 2

VÀO: Xâu bit x có độ dài $r = 64t$ với $t \geq 2$.

RA : Mã băm 128 bit của x

Phân x thành các khối 64 bit $x_i : (x_1, x_2, ..., x_t)$.

Chọn các hằng số không bí mật IV và $\widetilde{\text{IV}}$ từ một tập các giá trị khuyến nghị đã được mô tả trước. Tập ngầm định các giá trị cho trước này là (ở dạng HEXA)

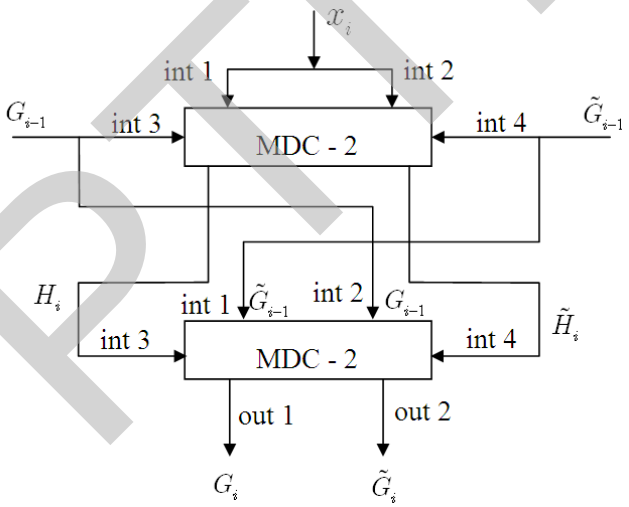
$$\text{IV} = 0\text{x}5252525252525252$$
$$\widetilde{\text{IV}} = 0\text{x}2525252525252525$$

Ký hiệu $||$ là phép ghép và .. là các nửa 32 bit phải và trái của C_i

Đầu ra $h(x) = H_t \,||\, \tilde{H}_t$ được xác định như sau: (với $1 \leq i \leq t$)

$$H_0 = IV,\ k_i = g(H_{i-1}),\ C_i = E_{k_i}(x_i) \oplus x_i,\ H_i = C_i^L \,||\, \tilde{C}_i^R$$
$$\tilde{H}_0 = \tilde{IV},\ \tilde{k}_i = \tilde{g}(H_{i-1}),\ \tilde{C}_i = E_{\tilde{k}_i}(x_i) \oplus x_i,\ \tilde{H}_i = \tilde{C}_i^L \,||\, C_i^R$$

Thuật toán MDC - 4 có thể được mô tả theo sơ đồ sau:



Hình 4.4. Sơ đồ hàm băm MDC – 4

4.1.4. Các hàm băm có khoá (MAC)

Định nghĩa 4.6: Định nghĩa thuật toán mã xác thực thông báo (MAC).

Thuật toán MAC là một họ các hàm h_k (được tham số hoá bằng một khoá bí mật k) có các tính chất sau:

a) Dễ dàng tính toán: Với h_k đã biết và giá trị k cho trước và một đầu vào x, $h_k(x)$ có thể được tính dễ dàng ($h_k(x)$ được gọi là giá trị MAC hay MAC).

b) Nén: h_k ánh xạ một đầu vào x có độ dài bit hữu hạn tuỳ tới một đầu ra $h_k(x)$ có độ dài bit n cố định.

c) Khó tính toán: Với các cặp giá trị $(x_i, h_k(x_i))$ không có khả năng tính một cặp $(x_i, h_k(x_i))$ với $x \neq x_i$ (kể cả có khả năng $h_k(x) = h_k(x_i)$ với một i nào đó).

Nếu tính chất *c* không thoả mãn thì thuật toán được coi là giả mạo MAC.

Các hàm băm có khoá được sử dụng để xác thực thông báo và thường được gọi là *các thuật toán tạo mã xác thực thông báo (MAC).*

MAC dựa trên các mật mã khối.

Thuật toán

VÀO: Dữ liệu x , mật mã khối E, khoá MAC bí mật k của E.

RA : n bit MAC trên x (n là độ dài khối của E)

1) Độn và chia khối: Độn thêm các bit vào x nếu cần. Chia dữ liệu đã độn thành từng khối n bit: $x_1, x_2, ..., x_t$.

2) Xử lý theo chế độ CBC.

Ký hiệu E_k là phép mã hoá E với khoá k .

Tính khối H_t như sau:

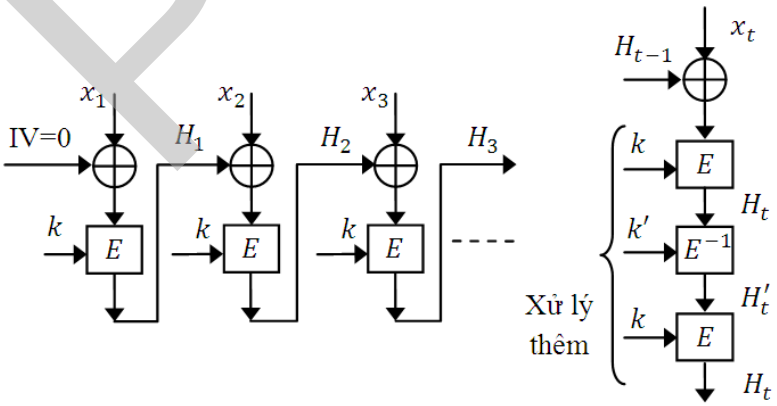
$$H_1 \leftarrow E_k(x_1)$$
$$H_i \leftarrow K_k(H_{i-1} \oplus x_i)\ 2 \le i \le t$$

3) Xử lý thêm để tăng sức mạnh của MAC

Dùng một khoá bí mật thứ hai $k' \neq k$. Tính

$$H_t' \leftarrow E_{k'}^{-1}(H_t),\ H_t = E_k(H_t')$$

4) Kết thúc: MAC là khối n bit H_t



Hình 4.5. Thuật toán MAC dùng CBC

4.1.5. Tính toàn vẹn của dữ liệu và xác thực thông báo

Định nghĩa 4.7:

Tính toàn vẹn của dữ liệu là tính chất đảm bảo dữ liệu không bị sửa đổi một cách bất hợp pháp kể từ khi dữ liệu được tạo ra, được phát hoặc được lưu giữ bởi một nguồn được xác định.

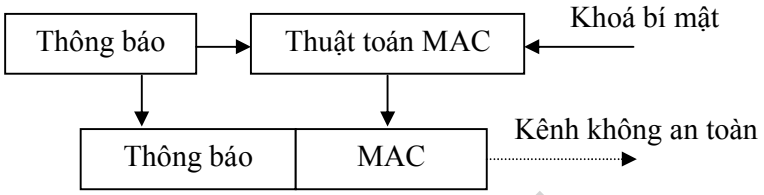
Định nghĩa 4.8:

Xác thực tính nguyên bản của dữ liệu là một kiểu xác thực đảm bảo một bên liên lạc được chứng thực là nguồn thực sự tạo ra dữ liệu đó ở một thời điểm nào đó trong quá khứ.

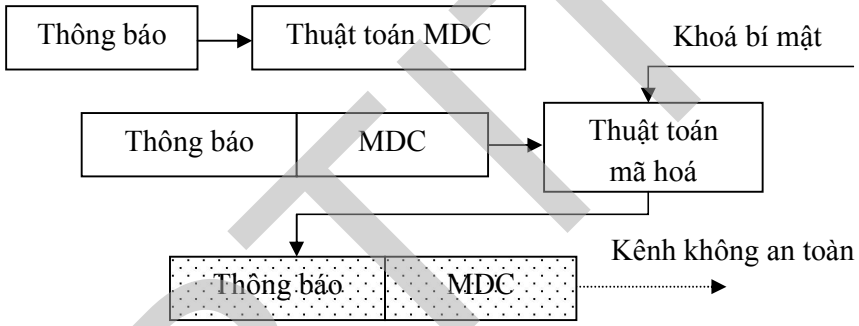
Xác thực thông báo là một thuật ngữ được dùng tương đương với xác thực nguyên gốc của dữ liệu.

Có ba phương pháp cung cấp tính toàn vẹn của dữ liệu bằng cách dùng các hàm băm.

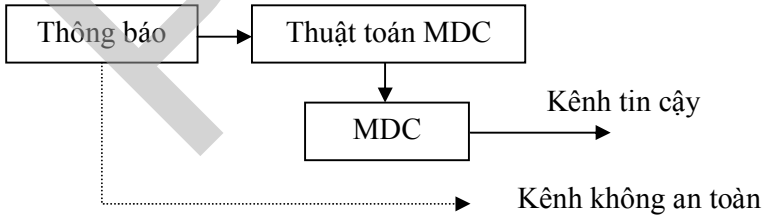
a) Chỉ dùng MAC:



b) Dùng MDC và mã hoá:



c) Sử dụng MDC và kênh tin cậy



Các phương pháp đảm bảo xác thực tính nguyên vẹn của dữ liệu.

- Dùng MAC.
- Dùng các sơ đồ chữ ký số.
- Gắn (trước khi mã hoá) một giá trị thẻ xác thực bí mật vào văn bản được mã.

4.2. CHỮ KÝ SỐ

Trong quá trình làm việc với các văn bản bằng giấy chúng ta sử dụng chữ ký tươi, các yêu cầu cơ bản quan trọng của chữ ký tươi bao gồm:

+ Ngắn gọn (ngắn hơn văn bản cần ký)

+ Là đại diện duy nhất cho người ký.

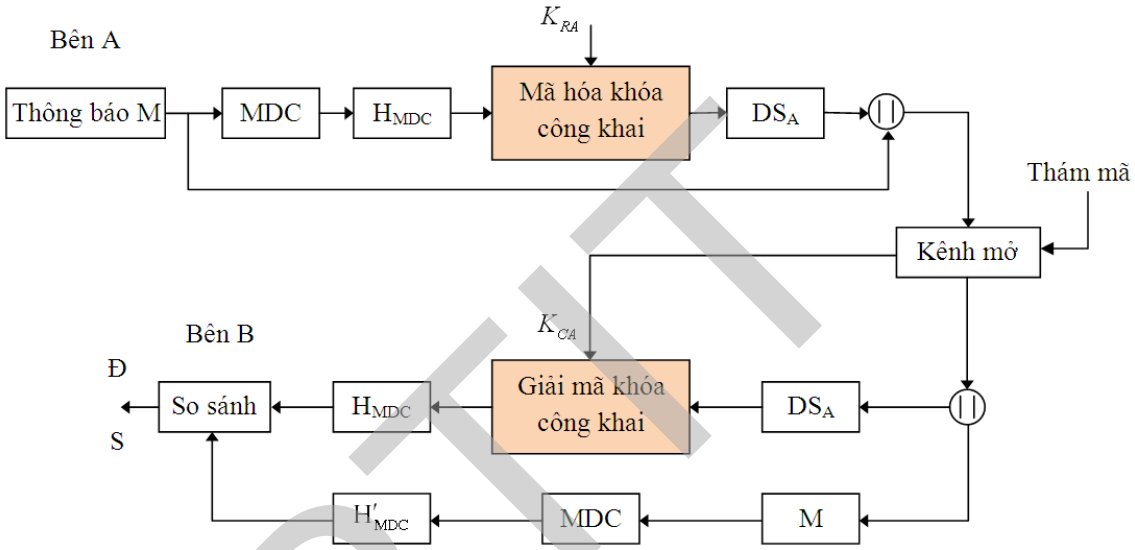
+ Khó bắt chước.

Khi làm việc qua mạng, do không gặp trực tiếp nên phải dùng chữ ký điện tử và nó cũng thỏa mãn các yêu cầu như chữ ký tươi. Tuy nhiên chữ ký số gắn với từng văn bản.

Nếu dùng hàm băm không khóa để làm chữ ký thì ai cũng làm được, còn dùng hàm băm có khóa thì phải trao đổi khóa, có thể giả mạo được.

4.2.1. Sơ đồ chữ ký số

Để thực hiện được chữ ký số, người ta xây dựng trên cơ sở kết hợp mã hoá khoá công khai với hàm băm như được mô tả trong Hình 4.6.



Hình 4.6. Sơ đồ chữ ký số

Trong sơ đồ Hình 4.6, quá trình tạo chữ ký số của A như sau:

Bên A cần tạo chữ ký số cho thông báo M (văn bản cần ký), đầu tiên A băm M bằng hàm băm không khóa (MDC) được mã băm đầu ra H_{MDC}, mã băm này được mã hóa bằng một hệ mật khóa công khai bằng *khóa bí mật* K_{RA} của A, kết quả được chữ ký số của A là DS_A. Cuối cùng chữ ký số này được ghép với thông báo M và gửi đến cho B.

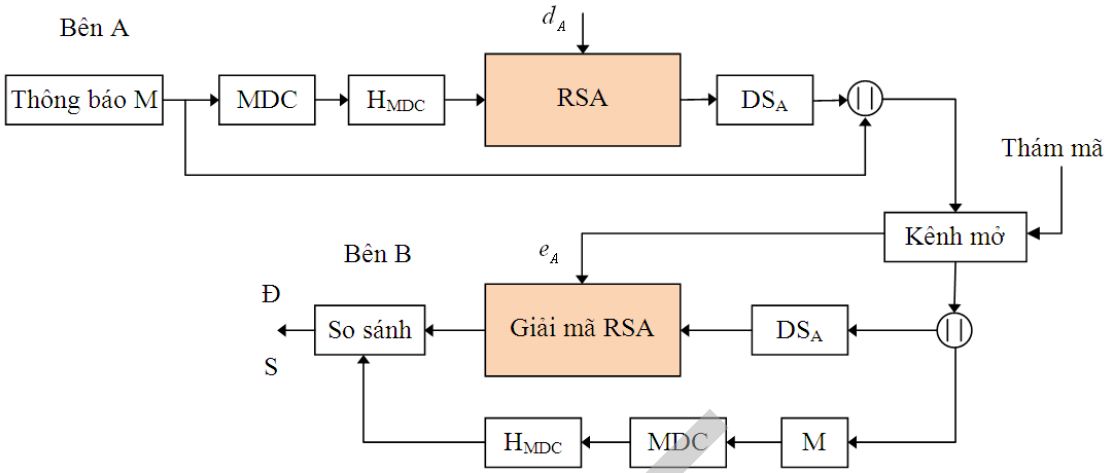
Bên nhận B tiến hành kiểm tra chữ ký số:

- Tách chữa ký số DS_A khỏi thông báo M.
- Dùng khóa *công khai của A là* K_{CA} để giải mã DS_A thu được mã băm H_{MDC} của thông báo M của A khi chưa truyền qua kênh mở.
- Tiến hành băm thông báo M theo đúng thuật toán của A để có được mã băm H'_{MDC} của thông báo M đã truyền qua kênh mở.
- So sánh hai mã băm H_{MDC} và H'_{MDC} để xác thực thông báo M.

Trong sơ đồ chữ ký số ta nhận thấy rằng việc sử dụng hàm băm là để xác thực nội dung thông báo M, còn hệ mật khóa công khai dùng để xác thực chủ thể nội dung (Bên A)

4.2.2. Sơ đồ chữ ký số RSA.

Có thể coi bài toán xác thực là bài toán "đối ngẫu" với bài toán bảo mật. Vì vậy, sử dụng ngược thuật toán RSA ta có thể có được một sơ đồ chữ ký số RSA như sau:



Hình 4.7. Sơ đồ chữ ký số RSA không bảo mật

Tạo chữ ký số:

+ Băm thông báo M: $H_{MDC} = h(M)$

+ Mã hóa mã băm bằng khóa bí mật d_A để tạo chữ ký số: $DS_A = \left[h(M)\right]^{d_A} \bmod n_A$

+ Ghép chữ ký số với thông báo: $M \,||\, DS_A$

Kiểm tra chữ ký số:

+ Tách DS_A khỏi thông báo M

+ Giải mã DS_A bằng khóa công khai của A (e_A) để tìm mã băm:

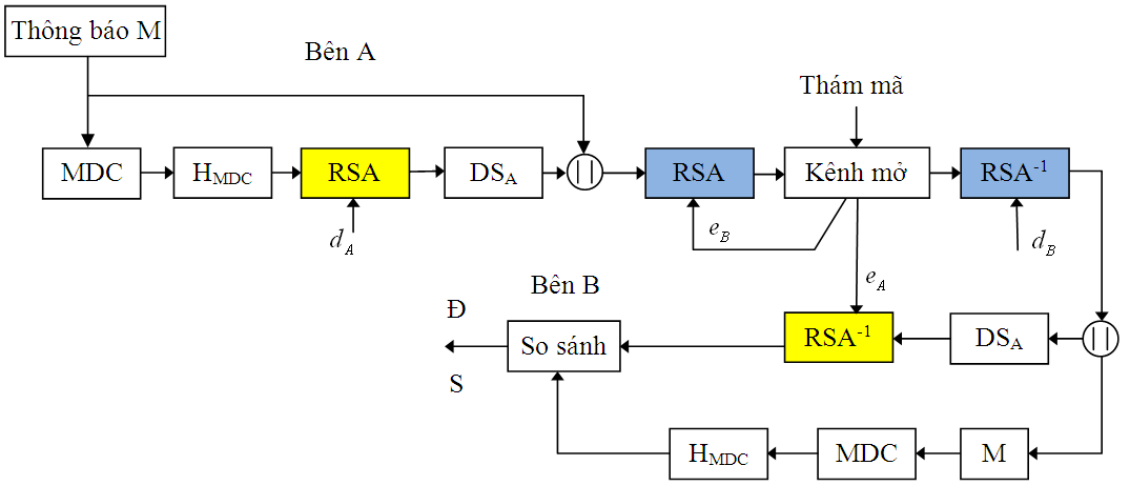
$$\mathrm{H}_{\mathrm{MDC}} = h(M) = \left(h(\mathrm{M})^{d_A}\right)^{e_A} \bmod n_A$$

+ Băm thông báo nhận được: $H'_{MDC} = h(M)$

+ So sánh hai mã băm $\mathrm{H}_{\mathrm{MDC}}$ và $\mathrm{H}'_{\mathrm{MDC}}$ để kiểm tra chữ ký số.

Sơ đồ chữ ký số trong Hình 4.7 là sơ đồ không bảo mật, khi cần bảo mật cả thông báo M người ta sử dụng sơ đồ Hình 4.8.

Trong sơ đồ Hình 4.8, sau khi A tạo được chữ ký số DS_A và ghép với thông báo M, thì chúng được mã hóa bằng hệ mật RSA với khóa công khai của B (e_B) và sau đó gửi đến B. Bên B sẽ tiến hành giải mã bằng khóa bí mật của B (d_B) trước khi kiểm tra chữ ký số.



Hình 4.8. Sơ đồ chữ ký số RSA có bảo mật

4.3. HỆ MẬT DỰA TRÊN ĐỊNH DANH

4.3.1. Ý tưởng cơ bản

Hệ mật dựa trên định danh do Shamin đề xuất [16] là một hệ mật bất đối xứng trong đó thông tin định danh của thực thể (tên riêng) đóng vai trò khoá công khai của nó. Trung tâm xác thực T được sử dụng để tính khoá riêng tương ứng của thực thể này. Trong các hệ mật khoá công khai thông thường mỗi người sử dụng có một cặp khoá (s,P) trong đó s là khoá bí mật (chỉ có người dùng này biết) còn P là khoá công khai mà mọi người đều có thể biết. Như vậy, các khoá công khai không cần phải giữ kín mà cần công bố rộng rãi. Tuy nhiên tính công khai này lại trở thành đối tượng cho các tấn công tích cực như việc thay khoá công khai giả vào vị trí khoá công khai thực trong danh bạ. Bởi vậy, ngoài cặp khoá (s,P) ta cần phải có chuỗi định danh I và không một dấu hiệu đảm bảo G để biết rằng P thực sự là khoá công khai của người dùng I và không phải là một kẻ giả mạo. Khi ta sử dụng các hệ mật dựa trên định danh, khoá công khai sẽ tương đương với định danh $(P=I)$. Còn dấu hiệu đảm bảo sẽ tương đương với khoá bí mật (tức là $G=s$). Hệ thống này có nhiều đặc tính tốt do không phải lưu trữ chứng chỉ để kiểm tra.

Sau khi tính khoá riêng của một người dùng T sẽ chuyển khoá riêng cho người dùng đó trên một kênh riêng an toàn. khoá riêng này được tính không chỉ từ thông tin định danh của thực thể mà còn phải là một hàm của một thông tin riêng nào đó chỉ có T mới biết (Khoá riêng của T). Đây là điều cần thiết nhằm tránh giả mạo và bắt chước. Điều chủ yếu là chỉ T mới có khả năng tạo các khoá riêng hợp lệ phù hợp với thông tin định danh.

4.3.2. Sơ đồ trao đổi khoá Okamoto-Tanaka

Phần này mô tả tóm lược sơ đồ trao đổi khoá Okamoto-Tanaka [17] là một hệ thống phân phối khoá dựa trên định danh. Sơ đồ này gồm 3 pha sau:

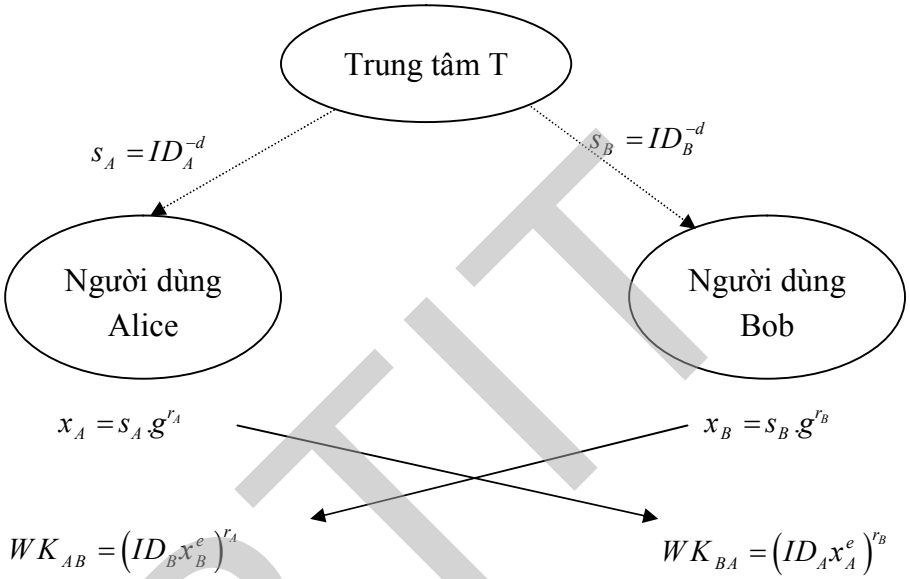
Pha chuẩn bị.

Trung tâm xác thực tin cậy chọn 2 số nguyên tố p và q và đưa công khai các giá trị n, g và e, trong đó $n = pq$, g là phần tử sinh của cả Z_p^* và Z_q^*, còn $e \in Z_{\lambda(n)}^*$. Ở đây, hàm Carmichael của n được xác định như sau:

$$\lambda(n) = BCNN\left(p-1, q-1\right)$$

Cho $d \in Z_{\lambda(n)}^*$ là khoá bí mật của trung tâm thỏa mãn điều kiện:

$$ed \equiv 1 \bmod \lambda(n)$$



Hình 4.9. Sơ đồ trao đổi khoá Okamoto-Tanaka

Pha tham gia của người dùng.

Cho ID_i là thông tin định danh của người dùng thứ i $\left(i = A, B, C, ...\right)$. Cho s_i là khoá bí mật của người dùng i thoả mãn:

$$s_i \equiv ID_i^{-d} \bmod n$$

Sau đó trung tâm T sẽ công bố $\left(e, n, g, ID_i\right)$ và phân phát s_i tới mỗi người dùng i qua một kênh an toàn (hoặc bằng cách dùng thẻ).

Pha tạo khoá chung.

Ta giả sử ở đây rằng hai người dùng Alice và Bob muốn chia sẻ một khoá chung (chẳng hạn để dùng cho một hệ mật khoá bí mật).

Trước tiên Alice tạo một số ngẫu nhiên r_A và tính:

$$x_A \equiv s_A g^{r_A} \bmod n$$

và gửi nó cho Bob.

Tương tự, Bob tạo một số ngẫu nhiên r_B và tính:

$$x_B \equiv s_B g^{r_B} \bmod n$$

và gửi nó cho Alice.

Tiếp theo, Alice tính:

$$WK_{AB} = \left(ID_B x_B^e\right)^{r_A} \bmod n$$

Tương tự, Bob tính

$$WK_{BA} = \left(ID_A x_A^e\right)^{r_B} \bmod n$$

WK_{AB} và WK_{BA} sẽ dùng làm khoá chung vì:

$$\begin{aligned} WK_{AB} &= \left(ID_B x_B^e\right)^{r_A} \bmod n \\ &= \left(ID_B (s_B g^{r_B})^e\right)^{r_A} \\ &= \left(ID_B (ID_B^{-d})^e . g^{r_B e}\right)^{r_A} \\ &= g^{e r_B r_A} \\ &= WK_{BA} \bmod n \end{aligned}$$

Ví dụ 3.19:

Hai bên liên lạc A và B chọn $p = 19$.

Tạo khóa: + A chọn $(m, n) = (11, 5)$ thỏa mãn $11 \times 5 = 55 \equiv 1 \bmod 18$.

+ B chọn $(u, v) = (7, 13)$ thỏa mãn $7 \times 13 = 91 \equiv 1 \bmod 18$

Giả sử A cần gửi bản tin $M = 6$ cho B. Quá trình truyền tin thực hiện như sau:

$\mathbf{A}_{(11,5)}$	$\mathbf{B}_{(7,13)}$
+ A tính $6^{11} \bmod 19 = 17$ và gửi cho B →	+ B nhận 17 và tính:
	$17^7 \bmod 19 = 5$
+ A nhận 5 và tính:	B gửi 5 cho A ←
$5^5 \bmod 19 = 9$ và gửi 9 cho B →	+ B nhận 9 và giải mã ra bản tin M:
	$5^{13} \bmod 19 = 6 = M$

Nhận xét:

- Để tìm bản tin M thám mã phải giải bài toán logarit rời rạc, với trường hợp p lớn thì đây là hệ mật an toàn.
- Để truyền bản tin M thì phải thực hiện truyền 3 lần, do đó tốc độ mã $R_{mã} = 1/3$ (thấp). Vì vậy hệ mật này chỉ thích hợp khi truyền các bản tin ngắn hoặc truyền khóa, phân phối khóa cho hệ mật khóa bí mật.

3.4.2.3. Hệ mật Elgamal

a) Tạo khóa

Mỗi bên liên lạc A,B tạo cho mình một cặp khóa công khai – bí mật theo các bước sau:

Bước 1: Chọn một số nguyên tố p lớn và α là một phần tử nguyên thủy $\left(\alpha \in \not{\mathbb{C}}_p^*\right)$

Bước 2: Chọn một số nguyên a ngẫu nhiên với $1 < a < p - 1$ và tính

$\alpha^a \bmod p$

Bước 3: + Khóa công khai là bộ 3 số: $\left(p, \alpha, \alpha^a\right)$

+ Khóa bí mật là: a

b) Mã hóa

Giả sử B cần gửi bản tin M cho A, B sẽ thực hiện các bước sau:

Bước 1: B nhận khóa công khai của A: $\left(p, \alpha, \alpha^a\right)$

Bước 2: B chọn số nguyên k ngẫu nhiên với $1 < k < p - 1$ và tính:

$$\begin{cases} \gamma = \alpha^k \bmod p \\ \delta = M\left(\alpha^a\right)^k \bmod p \end{cases} \quad (3.20)$$

Giả sử bản tin đã được biểu thị dưới dạng một số nguyên M trong dải $\{1,\ldots,p-1\}$. Phép tính mũ trong (3.20) được tính bằng thuật toán nhân và bình phương theo modulo.

Bước 3: B gửi bản mã $C = (\gamma, \delta)$ cho A.

Ta thấy bản mã C được ghép từ γ và δ nên nó có độ dài bit bằng 2 lần độ dài của M, đây là nhược điểm của hệ mật này.

c) Giải mã

A nhận bản mã C từ B và tiến hành giải mã theo các bước sau:

Bước 1: A sử dụng khóa bí mật a để tính:

$$\gamma^{p-1-a} \bmod p = \alpha^{-ak} \bmod p .$$

(Chú ý $\gamma^{p-1-a} = \gamma^{-a} = (\alpha^k)^{-a} = \alpha^{-ak}$)

Bước 2: A khôi phục bản rõ bằng cách tính:

$$\delta\gamma^{p-1-a} \bmod p = M\alpha^{ak}\alpha^{-ak} \bmod p = M$$

Ví dụ 3.20:

Tạo khoá.

Bước 1: A chọn $p = 17$ và phần tử nguyên thủy $\alpha = 3$ của $\not{c}_{17}^{*}$.

Bước 2: A chọn khóa bí mật $a = 6$ và tính $\alpha^a \bmod p = 3^6 \bmod 17 = 15$

Bước 3: + Khóa công khai của A là bộ 3 số: $\left(p, \alpha, \alpha^a\right) = (17,3,15)$

+ Khóa bí mật của A là: $a = 6$

Mã hoá

Giả sử B cần gửi bản tin $M = 7$ cho A.

Bước 1: B nhận khóa công khai của A: $\left(p, \alpha, \alpha^a\right) = (17,3,15)$.

Bước 2: B chọn số nguyên $k = 4$ và tính:

$$\begin{cases} \gamma = \alpha^k \bmod p = 3^4 \bmod 17 = 13 \\ \delta = M\left(\alpha^a\right)^k \bmod p = 7.(15)^4 \bmod 17 = 10 \end{cases}$$

Bước 3: B gửi bản mã $C = (\gamma, \delta) = (13,10)$ cho A.

Giải mã

A nhận bản mã $C = (\gamma, \delta) = (13,10)$ và tiến hành giải mã.

Bước 1: A sử dụng khóa bí mật $a = 6$ để tính:

$$\gamma^{p-1-a} \bmod p = 13^{10} \bmod 17 = 16$$

Bước 2: A khôi phục bản rõ bằng cách tính:

$$\delta\gamma^{p-1-a} \bmod p = 10.16 \bmod 17 = 7 = M$$

Nhận xét:

- Để tìm khóa bí mật a (từ α^a) thám mã phải giải bài toán logarit rời rạc (tính $a = \log_\alpha \alpha^a$), với trường hợp p lớn thì không thể giải được, hệ mật là an toàn.
- Hiệu quả truyền tin thấp, vì tốc độ mã chỉ đạt $R_{mã} = 1/2$

3.5. BÀI TOÁN PHÂN TÍCH THỪA SỐ VÀ HỆ MẬT RSA

3.5.1. Bài toán phân tích thừa số

Theo định lý 1.1, với mỗi số nguyên $n \geq 2$ ta luôn phân tích n được dưới dạng tích của luỹ thừa của các số nguyên tố và phân tích này là duy nhất như sau:

$$n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k} \tag{3.21}$$

Trong đó p_i là các số nguyên tố khác nhau và e_i là các số nguyên dương.

Nếu n là tích của hai số nguyên tố $n = pq$, trong đó p,q là hai số nguyên tố *lớn có độ lớn xấp xỉ nhau*, thì việc phân tích n thành tích của p và q là bài toán khó. Hay là, nếu ta biết p,q thì tính n rất đơn giản, nhưng ngược lại biết n thì rất khó tìm p và q.

3.5.2. Hệ mật RSA (Rivest – Shamir – Adleman)

3.5.2.1. Tạo khóa

Mỗi bên liên lạc A và B tự tạo cho mình một cặp khóa công khai – bí mật theo các bước sau đây:

Bước 1: Chọn 2 số nguyên tố lớn p và q có độ lớn tương đương.

Bước 2: Tính $n = pq$ và $\varphi(n) = \varphi(pq) = (p-1)(q-1)$

Bước 3: Chọn một số nguyên e ngẫu nhiên với $1 < e < \varphi(n)$ thỏa mãn: $(e, \varphi(n)) = 1$

Bước 4: Tính số nguyên duy nhất d, $1 < d < \varphi(n)$ thỏa mãn $ed \equiv 1 \bmod \varphi(n)$.

(Có thể sử dụng thuật toán Euclide mở rộng để tìm d)

Bước 5: + Khóa công khai công khai của A là cặp số: (n,e)

+ Khóa bí mật của A là số d.

Chú ý: Các số nguyên e và d trong thuật toán tạo khoá RSA được gọi là số mũ mã hoá và số mũ giải mã còn số n được gọi là modulus. Ta thấy e và d là hai số nghịch đảo nên vai trò của chúng là giống nhau, tức là nếu khóa công khai là (n,e) thì khóa bí mật là d, ngược lại nếu khóa công khai là (n,d) thì khóa bí mật sẽ là e

3.5.2.2. Mã hóa

Giả sử B cần gửi bản tin M cho A (với $M \le n$), B thực hiện các bước sau:

Bước 1: B nhận khóa công khai của A (n,e).

Bước 2: B tính (mã hóa):

$$C \equiv M^e \bmod n \tag{3.22}$$

Bước 3: B gửi bản mã C cho A.

3.5.2.3. Giải mã

A nhận bản mã C từ B và tiến hành giải mã:

$$C^d \bmod n = M^{ed} \bmod n = M \tag{3.23}$$

Nhận xét:

- Thám mã có e muốn biết d thì phải biết $\varphi(n)$, mà muốn biết $\varphi(n)$ thì thám mã phải thực hiện bài toán phân tích thừa số $n = pq$ để biết p và q. Khi p và q là các số nguyên tố lớn thì thám mã không thể giải được.
- Hiệu quả truyền tin của RSA cao ($R_{mã} \approx 1$).
- Thuật toán RSA có tính bền vững, vì các lý do trên mà RSA là một trong các hệ mật được sử dụng rộng rãi trong vòng hơn 30 năm qua.

Chú ý (Số mũ vạn năng)

Số $\lambda = BCNN(p-1, q-1)$ đôi khi được gọi là số mũ vạn năng của n, λ có thể được dùng thay cho $\varphi(p-1)(q-1)$ khi tạo khoá RSA. Cần chú ý rằng λ là ước thực sự của φ. Sử dụng λ có thể thu được số mũ giải mã d nhỏ hơn (làm cho giải mã nhanh hơn). Tuy nhiên, nếu p và q được chọn ngẫu nhiên thì $UCNN(p-1, q-1)$ sẽ khá nhỏ và bởi vậy Φ và λ sẽ là các số có kích thước xấp xỉ.

3.5.2.4. Một số ví dụ

Ví dụ 3.21:

Xây dựng các tham số cho hệ mật RSA (Tạo khoá)

Bước 1: A chọn $p = 7$ và $q = 17$.

Bước 2: A tính:

+ $n = pq = 7 \times 17 = 119$

+ $\varphi(n) = (p-1)(q-1) = 6 \times 16 = 96$

Bước 3: A chọn $e = 5$ thỏa mãn $(5,96) = 1$ vì $(5,6) = 1; (5,16) = 1$

Bước 4: Tính d thỏa mãn:

$$ed \equiv 1 \bmod \varphi(n) \Rightarrow 5d \equiv 1 \bmod 96 \quad (3.24)$$

Với các giá trị p,q còn nhỏ ta có thể giải phương trình đồng dư để tìm d. Từ biểu thức (3.24) ta có phương trình sau:

$$5d = 1 + k96$$

Với $k = 4$ tìm được: $d = \dfrac{1+4.96}{5} = 77$

Bước 5: + Khóa công khai của A là: $(119,5)$

+ Khóa bí mật của A là: 77

Cũng có thể chọn khóa công khai là $(119,77)$ và khóa bí mật là 5.

Ví dụ 3.22:

Xây dựng một hệ mật RSA để gửi bản tin **M = CRYPTOGRAPH** từ B đến A.

***Tạo khóa*:**

Bước 1: A chọn $p = 43, q = 59$.

Bước 2: A tính $n = pq = 43 \times 59 = 2537$

Và $\varphi(n) = (p-1)(q-1) = 42 \times 58 = 2436$

Bước 3: A chọn $e = 1357$

Bước 4: A tính ra $d = e^{-1} = 1357^{-1} = 517$

Bước 5: + Khóa công khai của A: $(n,e) = (2537,1357)$

+ Khóa bí mật của A: $d = 517$

Mã hóa:

Bên B biến đổi bản tin **M = CRYPTOGRAPH** như sau:

Biểu diễn dạng Hexa của M:

M =	**C**	**R**	**Y**	**P**	**T**	**O**	**G**	**R**	**A**	**P**	**H**
M_{HEX} =	43	52	59	50	54	4F	47	52	41	50	48

Các giá trị hexa của các ký tự lấy từ bảng mã ASCII, ví dụ chữ cái C có mã hexa tương ứng là 43.

Sau đó chuyển đổi các số hexa thành số nhị phân, ví dụ số 43 gồm 2 số hexa là 4 và 3, khi chuyển sang nhị phân ta chuyển từng số một: số $4_{Hex} \leftrightarrow 0100_{Bin}$, số $3_{Hex} \leftrightarrow 0011_{Bin}$ và như thế số 43 sẽ chuyển thành 8 bit nhị phân tương ứng là: $43_{Hex} \leftrightarrow 0100.0011_{Bin}$. Tiến hành chuyển đổi toàn bộ 11 chữ cái của M ta được biểu diễn nhị phân tương ứng của M như sau:

$$M_{Bin} = \underbrace{01000011}_{43}.\underbrace{01010010}_{52}.\underbrace{01011001}_{59}...\underbrace{01010000}_{50}.\underbrace{01001000}_{48} \quad (8\times 11 = 88bit)$$

Có một chú ý là khi mã hóa thì giá trị $M < n$, với n tính được ở trên ta thấy:

$n = 2537 > 2048 = 2^{11}$ như vậy mỗi lần chỉ mã hóa được tối đa 11 bit thông tin.

Chuỗi bit của bản tin rõ gồm 88 bit, vậy B chia thành 8 khối, mỗi khối 11 bit như sau:

$$M_{Bin} = \underbrace{01000011010}_{M_1}.\underbrace{10010010110}_{M_2}...\underbrace{00001001000}_{M_8}$$

Tiếp theo B sẽ chuyển 8 khối 11 bit $(M_1 \div M_8)$ thành giá trị thập phân tương ứng:

$$M_{Dec} = \underbrace{538}_{M_1}.\underbrace{1174}_{M_2}.\underbrace{672}_{M_3}.\underbrace{1348}_{M_4}.\underbrace{1955}_{M_5}.\underbrace{1353}_{M_6}.\underbrace{42}_{M_7}.\underbrace{72}_{M_8}$$

B tiến hành mã hóa từng khối M_i để có các khối bản mã C_i tương ứng như sau:

$$\begin{cases} C_1 = M_1^e \bmod n = 538^{1357} \bmod 2537 = 905 \\ C_2 = M_2^e \bmod n = 1174^{1357} \bmod 2537 = 1307 \\ C_3 = M_3^e \bmod n = 672^{1357} \bmod 2537 = 1040 \\ C_4 = M_4^e \bmod n = 1348^{1357} \bmod 2537 = 1987 \\ C_5 = M_5^e \bmod n = 1955^{1357} \bmod 2537 = 750 \\ C_6 = M_6^e \bmod n = 1353^{1357} \bmod 2537 = 1567 \\ C_7 = M_7^e \bmod n = 42^{1357} \bmod 2537 = 1590 \\ C_8 = M_8^e \bmod n = 72^{1357} \bmod 2537 = 1093 \end{cases}$$

Cuối cùng B sẽ biến đổi các bản mã C_i thành các bit nhị phân và ghép lại thành chuỗi 88 bit và truyền chuỗi bit này đến A.

***Giải mã*:**

A nhận chuỗi các bit bản mã do B gửi lại tách thành các khối 11 bit và biến đổi thành giá trị thập phân tương ứng để có các khối bản mã C_i. Tiếp theo A sẽ giải mã cho từng khối bản mã C_i để có các khối M_i theo nguyên tắc giải mã:

$$M_i = (C_i)^d \bmod n$$

Ví dụ: $M_1 = (C_1)^d \bmod n = 905^{517} \bmod 2537 = 538$

Sau khi đã giải mã hết 8 khối C_i, A có 8 khối M_i và A sẽ thực hiện theo quy trình ngược lại bên B để có bản tin rõ cuối cùng là **M = CRYPTOGRAPH**

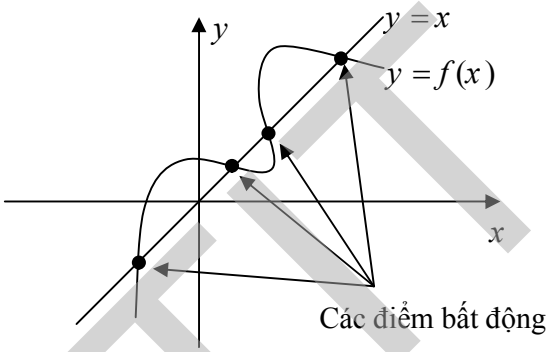
Việc mã hóa và giải mã của hệ mật RSA đều dựa vào hàm mũ, trong trường hợp các số mũ lớn, để tính toán được người ta sử dụng thuật toán nhân và bình phương để tính hàm lũy thừa modulo. Thuật toán này đã được trình bày trong mục 3.1.4.2.

3.5.3. Vấn đề điểm bất động trong RSA

Xét một ánh xạ $y = f(x)$ (phép mã hóa trong hệ mật), tồn tại các điểm x mà sau khi mã hóa bản tin không thay đổi, tức là:

$$\exists x : x = f(x)$$

Các điểm này gọi là các điểm bất động.



Hình 3.6. Các điểm bất động của $y = f(x)$

Hệ mật RSA cũng tồn tại các điểm bất động.

Ví dụ 3.23:

Xét hệ mật RSA với các tham số: $(n,e) = (35,17)$ với $p = 5, q = 7$. Giả sử bản tin $M = 8$, ta thấy:

$$C = M^e \bmod n = 8^{17} mod 35 = 8$$

Tức là $M = C$ như vậy là không che giấu được thông tin.

Số các điểm bất động của một hệ mật RSA được xác định theo định lý sau đây:

Định lý 3.19:

Với hệ mật RSA có tham số khóa công khai là (n,e) với $n = pq$ thì số bản tin không thể che giấu được (số điểm bất động) tính như sau:

$$\boxed{N = \left[1 + UCLN\left(e-1, p-1\right)\right]\left[1 + UCLN\left(e-1, q-1\right)\right]} \quad (3.25)$$

Ví dụ 3.24:

+ Với hệ mật RSA có $(n,e) = (35,3)$ với $p = 5, q = 7$ thì số bản tin không che giấu được là:

$$N = \left[1 + UCLN\left(2,4\right)\right]\left[1 + UCLN\left(2,6\right)\right] = 9$$

Các điểm bất động là: $M = \{0,1,6,14,15,20,21,29,34\}$

Như vậy xác suất gặp phải các bản tin không che giấu được khoảng $1/4$.

+ Vẫn hệ hệ mật RSA như trên nhưng ta thay đổi sốe : $(n,e) = (35,17)$ thì lúc này số bản tin không che giấu được là:

$$N = \left[1 + UCLN\left(16,4\right)\right]\left[1 + UCLN\left(16,6\right)\right] = 15$$

Các điểm bất động là: $M = \{0,1,6,7,8,13,14,15,20,21,22,27,28,29,34\}$

Và xác suất gặp phải các bản tin không che giấu được khoảng gần $1/2$.

3.5.4. Hệ mật Rabin

3.5.4.1. Tạo khoá

Mỗi bên liên lạc tạo cho mình một khoá công khai và một khoá bí mật tương ứng theo các bước sau:

Bước 1: Tạo 2 số nguyên tố lớn, ngẫu nhiên và phân biệt p và q có kích thước xấp xỉ nhau.

Bước 2: Tính $n = pq$.

Bước 3: + Khoá công khai là n.

+ Khoá bí mật là các cặp số (p,q).

3.5.4.2. Mã hóa

B cần gửi bản tin M cho A, phải thực hiện các bước sau:

Bước 1: Nhận khoá công khai của A: n

Bước 2: Tính $C = M^2 \bmod n$.

Bước 3: B gửi bản mã C cho A.

3.5.4.3. Giải mã:

Để khôi phục bản rõ M từ C, A phải thực hiện các bước sau:

Bước 1: Tìm 4 căn bậc hai của $C \bmod n$ là M_1, M_2, M_3, M_4.

Bước 2: Thông báo cho người gửi là một trong 4 giá trị M_1, M_2, M_3, M_4. Bằng một cách nào đó A sẽ quyết định M là giá trị nào.

Chú ý:

Tìm các căn bậc 2 của $C \bmod n$, $n = pq$ khi $p \equiv q \equiv 3 \bmod 4$. Trong trường hợp này, việc tìm 4 căn bậc 2 của $C \bmod n$ được thực hiện khá đơn giản như sau:

Sử dụng thuật toán Euclide mở rộng để tìm các số nguyên a và b thoả mãn: $ap + bq = 1$. Chú ý rằng a và b có thể được tính trong giai đoạn tạo khoá.

Tính $r = C^{(p+1)/4} \bmod p$.

Tính $s = C^{(q+1)/4} \bmod q$.

Tính $x = (aps + bqr) \bmod n$.

Tính $y = (aps - bqr) \bmod n$.

Bốn giá trị căn bậc 2 của $C \bmod n$ là: $x, -x \bmod n, y$ và $-y \bmod n$

Ví dụ 3.25:

** Tạo khoá.*

Bước 1: Bên A chọn hai số nguyên tố $p = 277, q = 331$.

Bước 2: A tính $n = pq = 91687$

Bước 3: + Khoá công khai của A là $n = 91687$.

+ Khoá bí mật của A là cặp số ($p = 277, q = 331$).

** Mã hoá*

Giả sử rằng 6 bit cuối cùng của bản tin gốc được lặp lại trước khi thực hiện mã hoá. Việc thêm vào độ thừa này nhằm giúp cho bên giải mã nhận biết được bản mã đúng.

Để mã hoá bản tin 10 bit $\overline{M} = 1001111001$, B sẽ lặp lại 6 bit cuối cùng của $\overline{M}$ để có được bản tin 16 bit sau: $M = 1001111001111001$, biểu diễn thập phân tương ứng là $M_{dec} = 40596$.

Bước 1: B nhận khóa công khai của A: $n = 91687$

Bước 2: B tính $C = M^2 \bmod n = 40596^2 \bmod 91687 = 62111$

Bước 3: B gửi C cho A.

** Giải mã*

Để giải mã bản mã C, A thực hiện:

Bước 1: Tính bốn giá trị căn bậc 2 của $C \bmod n$

$$M_1 = 69654, \quad M_2 = 22033, \quad M_3 = 40596, \quad M_4 = 51118$$

Biểu diễn nhị phân tương ứng của các số trên là:

$$M_1 = 10001000000010110, \quad M_2 = 101011000010001$$
$$M_3 = 1001111001111001, \quad M_4 = 1100011110101110$$

Vì chỉ có M_3 mới có độ thừa cần thiết nên A sẽ giải mã C bằng M_3 và khôi phục lại bản tin gốc là $\overline{M} = 1001111001$.

Đánh giá hiệu quả

Thuật toán mã hoá Rabin là một thuật toán cực nhanh vì nó chỉ cần thực hiện một phép bình phương modulo đơn giản. Trong khi đó, chẳng hạn với thuật toán RSA có $e = 3$ phải cần tới một phép nhân modulo và một phép bình phương modulo. Thuật toán giải mã Rabin có chậm hơn thuật toán mã hoá, tuy nhiên về mặt tốc độ nó cung tương đương với thuật toán giải mã RSA.

3.6. BÀI TOÁN XẾP BA LÔ VÀ HỆ MẬT MERKLE – HELLMAN

3.6.1. Bài toán xếp ba lô

3.6.1.1. Dãy siêu tăng

Định nghĩa 3.25: **Dãy siêu tăng**

Dãy các số nguyên dương $(a_1, a_2, ..., a_n)$ được gọi là dãy siêu tăng nếu $a_i \geq \sum_{j=1}^{i-1} a_j$ với $\forall i,\ 2 \leq i \leq n$

Ví dụ: dãy số $(1, 2, 4, 8, 16, 32, 64, 128)$

3.6.1.2. Bài toán xếp balô

Cho một tập hợp các gói có các trọng lượng khác nhau, liệu có thể xếp một số gói này vào ba lô để ba lô có một trọng lượng cho trước hay không. Về mặt hình thức ta có thể phát biểu bài toán trên như sau:

Cho tập các giá trị $M_1, M_2, ..., M_n$ và một tổng S. Hãy tính các giá trị b_i để:

$$S = b_1 M_1 + b_2 M_2 + ... + b_n M_n$$

với $b_i \in \{0, 1\}$

$b_i = 1$: Có nghĩa là gói M_i được xếp vào ba lô.

$b_i = 0$: Có nghĩa là gói M_i không được xếp vào ba lô.

Bài toán có 2^n phương án (2^n vectơ nhị phân b), bài toán với n lớn (vài trăm) thì không thể giải ở thời gian thực, hay đây là bài toán khó.

3.6.1.3. Giải bài toán xếp ba lô trong trường hợp dãy siêu tăng.

Trong trường hợp $M=\{M_1,M_2,...,M_n\}$ là một dãy siêu tăng thì việc tìm $b=(b_1,b_2,...,b_n)$ tương đương như bài toán tìm biểu diễn nhị phân của một số S. Biểu diễn này sẽ tìm được sau tối đa là n bước (độ phức tạp tương đương n).

Thuật toán giải:

VÀO: Dãy siêu tăng $M=\{M_1,M_2,...,M_n\}$ và một số nguyên S là tổng của một tập con trong M

RA : $(b_1,b_2,...,b_n)$ trong đó $b_i\in\{0,1\}$ sao cho: $\sum_{i=1}^{n}b_iM_i=S$ (Phương án sắp xếp)

Bước 1: $i\leftarrow n$

Bước 2: Chừng nào $i\geq 1$ hãy thực hiện:

Nếu $S\geq M_i$ thì

$b_i\leftarrow 1$

và $S\leftarrow S-M_i$

ngược lại: $b_i\leftarrow 0$

$i\leftarrow i-1$

Bước 3: Return (b)

Ví dụ 3.26:

Cho $M=(1,2,4,8,16,32,64)_{n=7}$; $S=57$, hãy tìm phương án sắp xếp b :

Bước 1: $i=7$

Bước 2: $i=7$: $57<64\rightarrow b_7=0, i=6$

$i=6$: $57>32\rightarrow b_6=1, S=57-32=25;\ i=5$

$i=5$: $25>16\rightarrow b_5=1, S=25-16=9;\ i=4$

$i=4$: $9>8\rightarrow b_4=1, S=9-8=1;\ i=3$

$i=3$: $1<4\rightarrow b_3=0;\ i=2$

$i=2$: $1<2\rightarrow b_2=0;\ i=1$

$i=1$: $1=1\rightarrow b_1=1, S=1-1=0;\ i=0$

Bước 3: ta có phương án sắp xếp: $b=(b_1,b_2,...,b_7)=(1,0,0,1,1,1,0)$

3.6.2. Hệ mật Merkle - Hellman

Hệ này và các hệ liên quan dựa trên tính khó giải của bài toán tổng các tập con (bài toán này là bài toán NP đầy đủ - là một lớp khá lớn các bài toán không có giải thuật được biết trong thời gian đa thức). Tuy nhiên tất cả các hệ mật xếp ba lô khác nhau đều đã bị chứng tỏ là không mật (ngoại trừ hệ mật Chor-Rivest).

3.6.2.1. Thuật toán tạo khoá

Mỗi bên liên lạc tạo cho mình một cặp khóa khoá công khai - khoá bí mật theo các bước sau đây:

Bước 1: Chọn một dãy siêu tăng $\{M_1, M_2, ..., M_n\}$ và một giá trị modulo M thỏa mãn:

$M \geq \sum_{i=1}^{n} M_i$ (M là phần tử thứ $n+1$ của dãy siêu tăng)

Bước 2: Chọn một số nguyên ngẫu nhiên W $(1 < W < M)$ sao cho $(W, M) = 1$. (Dễ nhất là chọn M là số nguyên tố)

Bước 3: Tính $a_i \equiv W M_i \bmod M$, với $i = 1, 2, ..., n$

Bước 4: + Khóa công khai: $A = (a_1, a_2, ..., a_n)$

+ Khóa bí mật: $(M, W, \{M_1, M_2, ..., M_n\})$

3.6.2.2. Mã hóa:

B cần gửi bản tin $m = (m_1, m_2, ..., m_n)$, $m_i \in \{0,1\}$ cho A. B thực hiện các bước sau:

Bước 1: B nhận khóa công khai của A: $(M, W, \{M_1, M_2, ..., M_n\})$

Bước 2: B tính $C = \sum_{i=1}^{n} m_i a_i$

Bước 3: B gửi bản mã C cho A.

3.6.2.3. Giải mã

A nhận bản mã C và tiến hành giải mã theo các bước sau:

Bước 1: A tính $d = W^{-1} C \bmod M = \sum_{i=1}^{n} m_i M_i$

(Chú ý: $d \equiv W^{-1} C \equiv W^{-1} \sum_{i=1}^{n} m_i a_i = W^{-1} \sum_{i=1}^{n} m_i W M_i = \sum_{i=1}^{n} m_i M_i$)

Do $(W, M) = 1$ nên sẽ $\exists W^{-1}$ việc tìm W^{-1} có thể theo thuật toán Euclid mở rộng.

Bước 2: Sử dụng thuật giải bài toán xếp ba lô trong trường hợp dãy siêu tăng để tính

$$d = \sum_{i=1}^{n} m_i M_i$$

Và tìm lại được $m = (m_1, m_2, ..., m_n)$

Ví dụ 3.27:

Bên A tạo khóa:

Bước 1: A chọn ngẫu nhiên các giá trị M_i trong các dải số sau để tạo dãy siêu tăng:

$$M_1 \in [1,16], M_2 \in [17,32], M_3 \in [33,64], M_4 \in [113,128]$$

A có dãy siêu tăng: $M = (5,23,57,119)$

do $\sum_{i=1}^{4} M_i = 204$ nên A chọn $M = 257$ (nguyên tố).

Bước 2: Chọn $W = 113$, $\exists W^{-1}$ vì $(113,257) = 1$, và tính được

$$W^{-1} = W^{\Phi(M)-1} \bmod M = 113^{255} \bmod 257 = 116$$

Bước 3: Tính $a_i \equiv W M_i \bmod M$, với $i = 1,2,3,4$

$$a_1 = 113.5 \bmod 257 = 51$$
$$a_2 = 113.23 \bmod 257 = 29$$
$$a_3 = 113.57 \bmod 257 = 16$$
$$a_4 = 113.119 \bmod 257 = 83$$

Bước 4: + Khóa công khai: $A = (a_1, a_2, a_3, a_4) = (51,29,16,83)$

+ Khóa bí mật: $(M, W, \{M_1, M_2, ..., M_n\}) = (257,113,\{5,23,57,119\})$

Mã hóa:

B cần gửi bản tin $m = (1,1,0,1)$ cho A.

Bước 1: B nhận khóa công khai của A: $(257,113,\{5,23,57,119\})$

Bước 2: B tính $C = \sum_{i=1}^{n} m_i a_i = 1 \times 51 + 1 \times 29 + 0 \times 16 + 1 \times 83 = 163$

Bước 3: B gửi bản mã $C = 163$ cho A.

Giải mã:

A nhận bản tin $C = 163$ và giải mã:

Bước 1: A tính $d = W^{-1} C \bmod M = 116.163 \bmod 257 = 147$

Bước 2: Sử dụng thuật giải bài toán xếp ba lô trong trường hợp dãy siêu tăng để tính

$$d = \sum_{i=1}^{n} m_i M_i = 147$$

$i = 4$ $\quad 147 > 119 \rightarrow m_4 = 1,\ \mathrm{S} = 147 - 119 = 28,\ i = 3$

$i = 3:$ $\quad 28 < 57 \rightarrow m_3 = 0;\ i = 2$

$i = 2:$ $\quad 28 > 23 \rightarrow m_2 = 1;\ \mathrm{S} = 28 - 23 = 5;\ i = 1$

$i = 1:$ $\quad 5 = 5 \rightarrow m_1 = 1,\ \mathrm{S} = 1 - 1 = 0;\ i = 0$

Và bản tin sau giải mã là: $m = (m_1, m_2, m_3, m_4) = (1,1,0,1)$

3.6.3. Hệ mật Chor-Rivest (CR)

Hệ mật CR là hệ mật khoá công khai xếp ba lô duy nhất hiện nay không sử dụng phép nhân modulo để ngụy trang bài toán tổng tập con.

3.6.3.1. Tạo khóa

Mỗi bên liên lạc tạo một khoá công khai và một khoá riêng tương ứng. A thực hiện các bước sau:

(1) Chọn một trường hữu hạn F_q có đặc số q, trong đó $q = p^h$, $p \geq h$ và đối với nó bài toán logarit rời rạc là khó giải.

(2) Chọn một đa thức bất khả quy định chuẩn ngẫu nhiên $f(x)$ bậc h trên Z_p. Các phần tử của F_q sẽ được biểu diễn bằng các đa thức trong $\mathrm{Z}_p[x]$ có bậc nhỏ hơn h với phép nhân được thực hiện theo $\bmod f(x)$.

(3) Chọn một phần tử nguyên thuỷ ngẫu nhiên $g(x)$ của F_q.

(4) Với mỗi phần tử của trường cơ sở $i \in \mathrm{Z}_p$, tìm logarit rời rạc $a_i = \log_{g(x)}(x + i)$ của các phần tử $x + i$ theo cơ số $g(x)$.

(5) Chọn một phép hoán vị ngẫu nhiên π trên các số nguyên $\{1, 2, \ldots, p-1\}$.

(6) Chọn một số nguyên ngẫu nhiên d, $0 \leq d \leq p^h - 2$

(7) Tính $C_i = \left(a_{\pi(i)} + d\right) \bmod \left(p^h - 1\right)$, $0 \leq i \leq p-1$.

(8) Khoá công khai của A là $\left(\left(C_0, C_1, \ldots, C_{p-1}\right), p, h\right)$

Khoá riêng của A là $\left(f(x), g(x), \pi, d\right)$.

3.6.3.2. Mã hoá

B cần mã hoá thông báo m để gửi cho A. B thực hiện các bước sau:

(1) Nhập khoá công khai của A $\left(\left(C_0, C_1, \ldots, C_{p-1}\right), p, h\right)$

(2) Biểu diễn thông báo như một xâu bit có độ dài $\left[\lg\binom{p}{h}\right]$ trong đó:

$$\binom{p}{h} = \frac{p!}{h!(p-h)!}.$$

(3) Xem m như là biểu diễn nhị phân của một số nguyên. Biến đổi số nguyên này thành một véctơ nhị phân $M = (M_0, M_1, \ldots, M_{p-1})$ có độ dài p và có đúng h con 1 như sau:

(a) Đặt $l \leftarrow h$

(b) For i from 1 to n do:

Nếu $m \geq \binom{P-i}{l}$ thì đặt $M_{i-1} \leftarrow 1, m \leftarrow m - \binom{p-i}{l}, l \leftarrow l - 1$.

Nếu không thì đặt

$$M_{i-1} \leftarrow 0 \quad \left(CY : \binom{n}{0} = 1 \qquad n \geq 0 \right.$$

$$\left. \binom{0}{l} = 0 \qquad l \geq 1 \right)$$

(4) Tính $C = \sum_{i=1}^{p-1} M_i c_i \bmod (p^h - 1)$.

(5) Gửi bản mã C cho A.

3.6.3.3. Giải mã

Để khôi phục bản mã rõ m từ C, A phải thực hiện các bước sau:

(1) Tính $r = (c - hd) \bmod (p^h - 1)$

(2) Tính $u(x) = g^r(x) \bmod f(x)$

(3) Tính $s(x) = u(x) + f(x)$ là một đa thức định chuẩn h trên Z_p.

(4) Phân tích $s(x)$ thành các nhân tử bậc nhất trên Z_p.

$$s(x) = \prod_{j-1}^{h} (x + t_j) \text{ trong đó } t_j \in Z_p$$

(5) Các thành phần có giá trị 1 của vectơ M có các chỉ số là $\pi^{-1}(t_j)$ với $1 \leq j \leq h$.

Các thành phần còn lại bằng 0

(6) Thông báo m được khôi phục lại từ M như sau

a) Đặt $m \leftarrow 0, l \leftarrow h$

b) For i from 1 to p do:

Nếu $M_{i-1}=1$ thì đặt $m \leftarrow m+\binom{p-i}{l}, l \leftarrow l-1$.

Chứng minh hoạt động giải mã:

Ta thấy

$$u(x)=g^2(x) \bmod f(x)$$

$$\equiv \left[g(x)\right]^{c-hd} \equiv \left[g(x)\right]^{\left(\sum_{i=0}^{p-1} M_i c_i\right)-hd}$$

$$\equiv \left[g(x)\right]^{\left(\sum_{i=0}^{p-1} M_i\left(a_{\pi(i)}+d\right)\right)-hd}$$

$$\equiv \left[g(x)\right]^{\sum_{i=0}^{p-1} M_i a_{\pi(i)}} \bmod f(x)$$

$$u(x) \equiv \prod_{i=0}^{p-1}\left[g(x)^{a_{\pi(i)}}\right]^{M_i} \equiv \prod_{i=0}^{p-1}\left(x+\pi(i)\right)^{M_i} \left(\bmod f(x)\right)$$

Vì $\prod_{i=0}^{p-1}\left(x+\pi(i)\right)^{M_i}$ và $s(x)$ là các đa thức định chuẩn bậc h và đồng dư với nhau theo modulo $f(x)$ nên $s(x)=u(x)+f(x)=\prod_{i=0}^{p-1}(x+\pi(i))^{M_i}$

Bởi vậy tất cả các căn bậc h của $s(x)$ đều nằm trong Z_p và áp dụng π^{-1} đối với các căn này ta sẽ có các toạ độ của *M* là 1

Ví dụ 3.28:

Tạo khoá: A thực hiện các bước sau:

(1) Chọn $p=7$ và $h=4$.

(2) Chọn đa thức bất khả quy $f(x)=x^4+3x^3+5x^2+6x+2$ có bậc 4 trên Z_7. Các phần tử của trường hữu hạn F_{7^4} được biểu diễn bằng các đa thức trong $Z_7[x]$.

(3) Chọn phần tử nguyên thuỷ ngẫu nhiên $g(x)=3x^3+3x^2+6$.

(4) Tính các logarit rời rạc sau:

$$a_0 = \log_{g(x)}(x) \quad = 1028$$
$$a_1 = \log_{g(x)}(x+1) = 1935$$
$$a_2 = \log_{g(x)}(x+2) = 2054$$
$$a_3 = \log_{g(x)}(x+3) = 1008$$
$$a_4 = \log_{g(x)}(x+4) = 379$$
$$a_5 = \log_{g(x)}(x+5) = 1780$$
$$a_6 = \log_{g(x)}(x+6) = 223$$

(5) Chọn phép hoán vị ngẫu nhiên trên $\{0,1,2,3,4,5,6\}$ như sau:

$\pi(0)=6$ $\pi(3)=2$ $\pi(5)=5$
$\pi(1)=4$ $\pi(4)=1$ $\pi(6)=3$
$\pi(2)=0$

(6) Chọn số nguyên ngẫu nhiên $d = 1702$

(7) Tính

$$C_0 = (a_6 + d) \bmod 2400 = 1925$$
$$C_1 = (a_4 + d) \bmod 2400 = 2081$$
$$C_2 = (a_0 + d) \bmod 2400 = 330$$
$$C_3 = (a_2 + d) \bmod 2400 = 1356$$
$$C_4 = (a_1 + d) \bmod 2400 = 1237$$
$$C_5 = (a_5 + d) \bmod 2400 = 1082$$
$$C_6 = (a_3 + d) \bmod 2400 = 310$$

(8) Khoá công khai của A là $((C_0, C_1, C_2, C_3, C_4, C_5, C_6), p = 7, h = 4)$

Khoá bí mật của A là $(f(x), g(x), \pi, d)$

Mã hoá:

Để mã hoá bản tin $m = 22$ gửi cho A, B làm như sau:

(1) Nhận khoá công khai của A.

(2) Biểu diễn m như một xâu bit độ dài 5: $m = 10110$ (Chú ý rằng $\left\lfloor \lg \binom{7}{4} \right\rfloor = 5$)

(3) Dùng phương pháp đã nêu ở trên bước c trong thuật toán trên để biến đổi m thành véctơ nhị phân M có độ dài M: $M = (1,0,1,1,0,0,1)$

(4) Tính $C = (C_0 + C_2 + C_3 + C_6) \bmod 2400 = 1521$

(5) Gửi $C = 1521$ cho A

Giải mã

A nhận bản mã C và giải mã:

(1) Tính $r = (c - hd) \bmod 2400 = 1913$

(2) Tính $u(x) = g(x)^{1913} \bmod f(x) = x^3 + 3x^2 + 2x + 5$

(3) Tính $g(x) = u(x) + f(x) = x^4 + 4x^3 + x^2 + x$

(4) Phân tích $s(x) = x(x+2)(x+3)(x+6)$

(Do đó $t_1 = 0, t_2 = 2, t_3 = 3, t_4 = 6$)

(5) Các thành phần của M bằng 1 có các chỉ số

$$\pi^{-1}(0) = 2 \quad \pi^{-1}(2) = 3 \quad \pi^{-1}(3) = 6 \quad \pi^{-1}(6) = 0$$

Bởi vậy $M = (1,0,1,1,0,0,1)$

(6) Sử dụng bước 6 trong thuật toán giải mã để biến đổi M thành số nguyên $m = 22$ và như vậy khôi phục được bản rõ ban đầu

Chú ý:

- Hệ mật này được xem là an toàn nếu không bị lộ khoá bí mật.
- Có thể mở rộng hệ mật này cho trường hợp Z_p với p là luỹ thừa của một số nguyên tố .
- Để làm cho bài toán logarit rời rạc là dễ giải, các tham số p và h phải chọn sao cho $q = p^h - 1$ chỉ có các nhân tử có giá trị nhỏ.
- Trong thực tế kích thước khuyến nghị của các tham số là $p \approx 200, h \approx 25$ (Ví dụ $p = 197$ và $h = 24$)
- Trở ngại lớn nhất của thuật toán là khoá công khai với kích thước chừng $p.h \log p$ bit là quá lớn. Ví dụ với $p = 197$ và $h = 24$ khoá công khai có chừng 36.000 bit.

3.7. BÀI TOÁN MÃ SỬA SAI VÀ HỆ MẬT McELIECE

3.7.1. Bài toán mã sửa sai

Định nghĩa 3.26:

Giải sử k, n là các số nguyên dương và $k \le n$. Mã $C(n,k)$ là một không gian k chiều của $(¢_2)^n$ (không gian véctơ của tất cả các véctơ nhị phân n chiều).

Ma trận sinh $G_{k \times n}$ của mã $C(n,k)$ là ma trận nhị phân có kích thước $k \times n$, các hàng của ma trận này tạo nên cơ sở của C.

Ma trận $H_{r \times n}$ với $r = n - k$ là ma trận kiểm tra, với: $G.H^T = [0]$

Giả sử $y,z \in (¢_2)^n$, trong đó $y=(y_1,y_2,...,y_n)$ và $z=(z_1,z_2,...,z_n)$. Ta xác định khoảng cách Hamming: $d(y,z)=\left|\{i:1\le i\le n, y_i \ne z_i\}\right|$ tức là số các toạ độ mà ở đó y và z khác nhau.

Khoảng cách mã được định nghĩa như sau:

$$d=\min\{d(y,z)\} \tag{3.26}$$

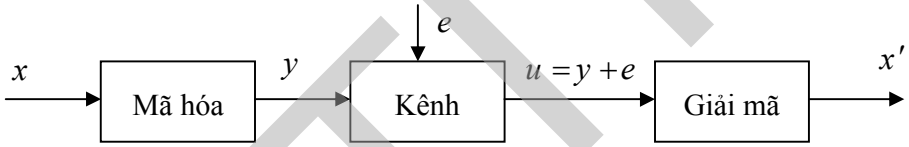
với y,z là hai từ mã khác nhau.

Mã sửa sai $C(n,k)$ có khoảng cách d được ký hiệu là mã $C(n,k,d)$.

Mã sửa sai được dùng để sửa các sai ngẫu nhiên xảy ra khi truyền số liệu (nhị phân) qua kênh có nhiễu. Điều đó được thực hiện như sau: Giả sử G là một ma trận sinh đối với mã $C(n,k,d)$, x là véctơ nhị phân k chiều cần truyền đi. Người gửi Alice sẽ mã hoá x thành một véctơ n chiều $y=xG$ rồi truyền y qua kênh.

Có hai phương pháp giải mã:

** Phương pháp "người láng giềng gần nhất"*



$x \in (¢_2)^k$ là bản tin đầu vào (k bit)

$y \in (¢_2)^n$ là từ mã đầu ra (n bit): $y=xG$

Giả sử Bob nhận được véctơ n chiều u không giống y, Bob sẽ giải mã u bằng chiến thuật giải mã "người láng giềng gần nhất". Theo chiến thuật này, Bob sẽ giải mã tất cả 2^k véctơ đầu vào x để được tập từ mã $|C|=2^k$ véctơ, sau đó so sánh tập này với véctơ thu được u để tìm thấy y' có khoảng cách tới u nhỏ nhất. Và từ đó sẽ xác định véctơ k chiều x' sao cho $y'=x'G$. Bob hy vọng $y=y'$ và do đó $x'=x$ (tức là Bob tin rằng các sai số trên đường truyền đã được sửa).

Dễ dàng thấy rằng, nếu sai số trên đường truyền nhiều nhất là $(d-1)/2$ thì trên thực tế chiến thuật này sẽ sửa được tất cả các sai.

Vì $|C|=2^k$ khi Bob so sánh u với mỗi từ mã, anh ta phải kiểm tra 2^k véctơ là một số lớn theo hàm mũ. Nói cách khác, thuật toán này không phải là thuật toán chạy trong thời gian đa thức. Hay độ phức tạp của thuật toán theo hàm mũ, và khi giá trị k lớn thì nó sẽ là bài toán khó giải.

** Phương pháp giải mã theo syndrom*

Một biện pháp khác (tạo cơ sở cho nhiều thuật toán giải mã thực tế) dựa trên khái niệm về syndrom. Các hàng của ma trận kiểm tra H sẽ tạo cơ sở cho các phần bù trực giao của C (ký hiệu là $C^{\perp}$) và được gọi là mã đối ngẫu với C. Nói cách khác, các hàng của H là những véctơ độc lập tuyến tính, còn $GH^T = [0]_{k\times r}$ là một ma trận $[0]$ cấp $k \times r$.

Cho véctơ $y \in (\not{\in}_2)^n$, ta xác định syndrom của y là $s(y) = yH^T$. Syndrom $s(y)$ là một véctơ hàng có $r = n - k$ thành phần.

Định lý 3.20:

Giả sử C là một bộ mã (n,k) có ma trận sinh G và ma trận kiểm tra tính chẵn lẻ H. Khi đó $y \in (\not{\in}_2)^n$ là một từ mã khi và chỉ khi $yH^T = [00...0]$.

Hơn nữa nếu $y \in C, e \in (\not{\in}_2)^n$ và $u = y + e$ thì $uH^T = eH^T$ hay $s(u) = s(e) = uH^T$.

Định lý trên phát biểu rằng syndrom chỉ phụ thuộc vào các sai số mà không phụ thuộc vào từ mã cụ thể nào được truyền đi.

Điều này gợi ý tới một cách giải mã gọi là *giải mã theo syndrom*. Trước tiên Bob tính $s(u) = uH^T$ nếu $s(u) = [0]$, thì anh ta có từ mã đúng $y = u$ và tìm ra x. Nếu không thì Bob sẽ lần lượt tạo tất cả các véctơ sai e có trọng số 1. Với mỗi véctơ này, Bob tính $s(e) = eH^T$, nếu có một véctơ e nào đó thoả mãn $s(e) = eH^T = s(u)$ thì Bob sẽ có từ mã đúng là $y = u - e$ và từ sẽ tìm lại được x. Nếu không có e nào thỏa mãn Bob tiếp tục làm như thế với các vectơ sai có trọng số $2,3,...,\lfloor (d-1)/2 \rfloor$ để tìm ra e thỏa mãn $s(e) = s(u)$.

Số các trường hợp sai $N_e \le t$, tính như sau (với t là số sai khả sửa $t = \lfloor (d-1)/2 \rfloor$):

$$N_e = \sum_{i=0}^{t} C_n^i \tag{3.27}$$

Như vậy, khi giải mã theo syndrom Bob phải so sánh $s(u)$ với $s(e)$ trong N_e trường hợp. Và theo (3.27) độ phức tạp của bài toán tương đương 2^r, tức là vẫn theo hàm mũ, khi r lớn thì bài toán là khó giải.

3.7.2. Hệ mật McEliece

3.7.2.1. Tạo khóa

Mỗi bên liên lạc tạo cho mình một cặp khóa công khai - khóa bí mật theo các bước sau:

Bước 1: Chọn một mã tuyến tính sửa sai (n,k,d) (khuyến nghị $k \approx n/2$) với ma trên sinh G, bộ mã có một thuật toán giải mã hiệu quả.

Bước 2:

+ Chọn ma trận hoán vị P_n cấp $n \times n$, sao cho tồn tại ma trận nghịch đảo P_n^{-1} với: $P_n P_n^{-1} = I$ (với I là ma trận đơn vị). Mỗi hàng và mỗi cột của P_n chỉ có duy nhất một con số 1.

+ Chọn một ma trận khả nghịch $S_{k\times k}$ $\left(\exists S^{-1}, SS^{-1} = I\right)$

Bước 3: Tính $G' = SGP$

Bước 4: + Khóa công khai: (G', t).

+ Khóa bí mật: (S, P, G)

Chú ý: Các tham số của mã Goppa có dạng $n = 2^v, d = 2t + 1$ và $k = n - vt$. Để áp dụng trong thực tế cho một hệ mật khoá công khai, McEliece đề nghị chọn $v = 10$ và $t = 50$. Điều này ứng với mã Goppa $(1024, 524, 101)$. Mỗi bản rõ m là một véctơ nhị phân cấp 524 và mỗi bản mã C là một véctơ nhị phân cấp 1024. Khoá công khai G' là một ma trận nhị phân cấp $524 \text{ x } 1024$.

3.7.2.2. Mã hóa

Giả sử B cần gửi bản tin m cho A, B thực hiện các bước sau:

Bước 1: B nhận khóa công khai của A: (G', t)

Bước 2: B tính $y = mG' = xG', (x = m)$

Bước 3: B chọn $e \in (¢_2)^n$ thỏa mãn $W(e) \le t$

Bước 4: B tính $c = y + e$ (Chèn thêm véctơ sai)

Bước 5: B gửi bản mã c cho A.

3.7.2.3. Giải mã

A nhận bản mã c từ B và tiến hành giải mã theo các bước sau:

Bước 1: A tính:

$$\begin{aligned} u = c.P^{-1} &= (y + e)P^{-1} = yP^{-1} + eP^{-1} \\ &= xG'P^{-1} + eP^{-1} \\ &= xSGPP^{-1} + eP^{-1} = xSG + eP^{-1} \end{aligned}$$

Hay $$u = x'G + eP^{-1} = y' + e'$$

Trong đó: $e' = eP^{-1}$ là một véctơ sai nào đó.

Bước 2: A giải mã sửa sai từ mã u để tìm x'.

Bước 3: A tính $x = x'S^{-1} = x = m$ để lấy lại bản tin rõ.

Ví dụ 3.29:

** Tạo khóa: Bên A tạo khóa.*

Bước 1: Chọn mã tuyến tính sửa sai $(n,k,d)=(7,4,3)$, với ma trận sinh:

$$G=\begin{pmatrix}1&0&0&0&1&1&0\\0&1&0&0&1&0&1\\0&0&1&0&0&1&1\\0&0&0&1&1&1&1\end{pmatrix}$$

Và $$t=\left\lfloor\frac{d-1}{2}\right\rfloor=1$$

Bước 2:

+ Chọn ma trận hoán vị P_n :

$$P_n=\begin{pmatrix}0&1&0&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&0&0&1\\1&0&0&0&0&0&0\\0&0&1&0&0&0&0\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\end{pmatrix}$$

+ Chọn một ma trận khả nghịch $S_{k\times k}$:

$$S=\begin{pmatrix}1&1&0&1\\1&0&0&1\\0&1&1&1\\1&1&0&0\end{pmatrix}$$

Bước 3: Tính $G'=SGP$

$$G'=\begin{pmatrix}1&1&1&1&0&0&0\\1&1&0&0&1&0&0\\1&0&0&1&1&0&1\\0&1&0&1&1&1&0\end{pmatrix}$$

Bước 4: + Khóa công khai: (G',t).

+ Khóa bí mật: (S,P,G)

** Mã hóa:*

Giả sử B cần gửi bản tin $m=x=(1\quad 1\quad 0\quad 1)$ cho A.

Bước 1: B nhận khóa công khai của A: (G',t)

Bước 2: B tính $y = mG'$

$$y = \begin{pmatrix}1 & 1 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0\\ 1 & 1 & 0 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 1 & 1 & 0 & 1\\ 0 & 1 & 0 & 1 & 1 & 1 & 0\end{pmatrix} = \begin{pmatrix}0 & 1 & 1 & 0 & 0 & 1 & 0\end{pmatrix}$$

Bước 3: B chọn véctơ sai $e = \begin{pmatrix}0 & 0 & 0 & 0 & 1 & 0 & 0\end{pmatrix}$ có $W(e) = 1 = t$

Bước 4: B tính $c = y + e$

$$c = \begin{pmatrix}0 & 1 & 1 & 0 & 0 & 1 & 0\end{pmatrix} + \begin{pmatrix}0 & 0 & 0 & 0 & 1 & 0 & 0\end{pmatrix} = \begin{pmatrix}0 & 1 & 1 & 0 & 1 & 1 & 0\end{pmatrix}$$

Bước 5: B gửi bản mã c cho A.

** Giải mã:*

Khi A nhận được bản mã c , A tiến hành giải mã:

Bước 1: A tính $u = cP^{-1}$

$$u = \begin{pmatrix}0 & 1 & 1 & 0 & 1 & 1 & 0\end{pmatrix}\begin{pmatrix}0 & 0 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0\end{pmatrix} = \begin{pmatrix}1 & 0 & 0 & 0 & 1 & 1 & 1\end{pmatrix}$$

Bước 2: B giải mã $u = y' + e'$ bằng phương pháp syndrom tìm được:

$$y' = \begin{pmatrix}1 & 0 & 0 & 0 & 1 & 1 & 0\end{pmatrix}$$

Cần để ý là $e' \neq e$ do phép nhân với P^{-1}

Sau đó B lập $x' = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}$ (bốn thành phần đầu tiên của y', vì mã sửa sai là mã hệ thống).

Bước 3:

Cuối cùng B tính ra bản rõ:

$$x = x'S = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}1 & 1 & 0 & 1\\ 1 & 1 & 0 & 0\\ 0 & 1 & 1 & 1\\ 1 & 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 1 & 0 & 1\end{pmatrix} = m$$

3.8. ĐƯỜNG CONG ELLIPTIC VÀ CÁC HỆ MẬT LIÊN QUAN

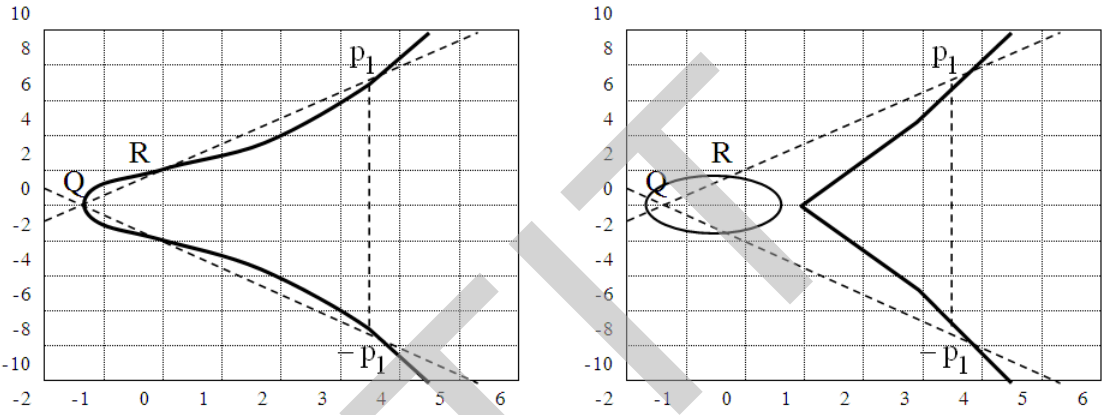
3.8.1. Các đường cong Elliptic.

Một đường cong Elliptic là một phương trình bậc 3 có dạng sau:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \tag{3.28}$$

Trong đó a,b,c,d,e là các số thực.

Trên các đường cong E ta xác định một phép cộng đặc biệt với một điểm O được gọi là điểm vô cực. Nếu trên đường thẳng cắt đường cong E ở ba điểm thì tổng của chúng bằng điểm vô cực O (điểm O này có vai trò như phần tử đơn vị trong phép cộng này). Hình 3.7 sau mô tả các đường cong E $y^2 = x^3 + 2x + 5$ và $y^2 = x^3 - 2x + 1$



Hình 3.7. Các đường cong $y^2 = x^3 + 2x + 5$ và $y^2 = x^3 - 2x + 1$

3.8.2. Các đường cong Elliptic trên trường Galois

Định nghĩa 3.27:

Đường cong Elliptic trên trường hữu hạn Z_p (với p là số nguyên tố) dạng Weiestrass được mô tả như sau:

$$y^2 \equiv x^3 + ax + b \bmod p \tag{3.29}$$

với $x \in Z_p$; a,b nguyên dương.

Điều kiện tồn tại là:

$$\boxed{\Delta = 4a^3 + 27b^2 \bmod p \neq 0} \tag{3.30}$$

Ví dụ 3.30:

Xét $p = 17$ và đường cong elliptic: $y^2 \equiv x^3 + x + 1 \bmod 17$.

$\Delta = 4 + 27 \bmod 17 = 14 \neq 0$ đảm bảo điều kiện tồn tại.

Xác định các thặng dư bậc 2: Q_{17}

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
i^2	1	4	9	16	8	2	15	13	13	15	2	8	16	9	4	1

Vậy $Q_{17} = \{1,4,9,16,8,2,15,13\}$ và các căn bậc hai:

$$\sqrt{1} = (1,16);\ \sqrt{4} = (2,15);\ \sqrt{9} = (3,14);\sqrt{16} = (4,13);$$
$$\sqrt{8} = (5,12);\sqrt{2} = (6,11);\ \sqrt{15} = (7,10);\ \sqrt{13} = (8,9)$$

Chú ý: Việc tính thặng dư bậc 2 và các căn bậc 2 theo các định nghĩa 3.18, 3.19; và các định lý 4.14 và 3.15.

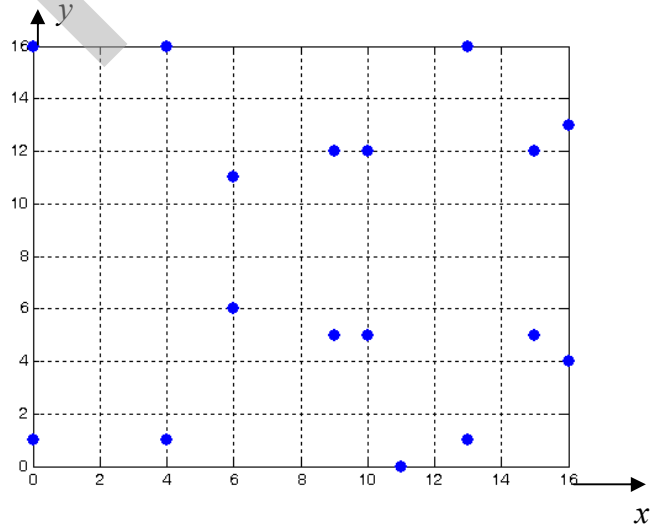
Từ đây tính được các cặp giá trị (x,y) trên đường cong elliptic như bảng dưới:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
y^2	**1**	3	11	14	**1**	12	**2**	11	11	**8**	**8**	0	7	**1**	5	**8**	**16**
$y^2 \in Q_{17}$?	Y	N	N	N	Y	N	Y	N	N	Y	Y	**N**	N	Y	N	Y	Y
y_1	1				1		6			5	5	**0**		1		5	4
y_2	16				16		11			12	12	**0**		16		12	13

Chú ý: mặc dù $y^2 = 0$ không thuộc Q_{17} nhưng số 0 có căn bậc hai chính là 0.

Từ đây ta có được nhóm cộng $E_{17}(1,1)$ gồm 18 phần tử trên đường cong elliptic $y^2 \equiv x^3 + x + 1 \bmod 17$ như sau:

$$E_{17}(1,1) = \{(0,1),(0,16),(4,1),(4,16),(6,6),(6,11),(9,5),(9,12),(10,5)$$
$$(10,12),(11,0),(13,1),(13,16),(15,5),(15,12),(16,4),(16,13),0\}$$



Hình 3.8. Tọa độ các điểm của nhóm cộng trên đường cong elliptic $y^2 \equiv x^3 + x + 1 \bmod 17$

Trong nhóm cộng này có thêm phần tử cuối cùng là phần tử zero (0).Nhóm này có cấp bằng 18 và có 6 phần tử nguyên thủy $(\Phi(18)=6)$ để tạo nhóm cyclic.

Tọa độ các điểm của nhóm được mô tả trong Hình 3.8

3.8.3. Các phép toán cộng và nhân trên các nhóm E

Giả sử $P=(x_1,y_1)$, $Q=(x_2,y_2)$ là các điểm trong nhóm $E_p(a,b)$, 0 là điểm vô cực. Các quy tắc đối với phép cộng trên nhóm con $E_p(a,b)$ như sau:

(1) $P+0=0+P=P$.

(2) Nếu $x_2=x_1$ và $y_2=-y_1$ tức là $P=(x_1,y_1)$ và $Q=(x_2,y_2)=(x_1,-y_1)=-P$ thì $P+Q=0$.

(3) Nếu $Q\neq -P$ thì tổng $P+Q=K(x_3,y_3)$ với tọa độ x_3,y_3 của K xác định như sau:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \bmod p \\ y_3 &= \lambda(x_1 - x_3) - y_1 \bmod p \end{aligned} \tag{3.31}$$

Trong đó:

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \bmod p, \text{ nếu } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1} \bmod p, \text{ nếu } P = Q \end{cases} \tag{3.32}$$

** Cách xây dựng nhóm cộng trên đường cong elliptic*

Các phần tử của nhóm được tính như sau:

- Phần tử zero: $0(\infty,\infty)$ nằm ngoài tập hợp.
- Các phần tử khác tính theo quy tắc phép cộng các điểm trên đường cong Elliptic như trình bày trên đây.

Ví dụ 3.31:

Xây dựng nhóm $E_p(a,b)=E_{17}(1,1)$ từ phần tử nguyên thủy $P(0,1)$ và tìm tất cả các phần tử nguyên thủy của nhóm.

Theo nguyên tắc tính các điểm ở biểu thức (3.31) và (3.32) ta tính được các điểm của nhóm cộng như sau.

+ Điểm $P(0,1)$

+ Điểm $2P=P+P$

$$\lambda = \frac{3.0+1}{2.1} = 2^{-1} \bmod 17 = 9 \bmod 17 \Rightarrow x_3 = 81 \bmod 17 = 13, y_3 = 1$$

Vậy $2P(13,1)$

Chú ý: số 2 và 9 là hai số nghịch đảo, có nhiều cách tính nghịch đảo, sau đây là một cách tính theo nhóm nhân Z_{17}^* có phần tử sinh là 3.

Bảng 3.11. Nhóm nhân Z_{17}^* với phần tử sinh $\alpha = 3$

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$3^i \bmod 17$	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1

+ Điểm $3P = P + 2P$

$$\lambda = \frac{0}{12} = 0 \Rightarrow \begin{cases} x_3 = -13 \bmod 17 = 4 \\ y_3 = -1 \bmod 17 = 16 \end{cases}$$

$$\Rightarrow 3P(4,16)$$

+ Điểm $4P = P + 3P$

$$\lambda = \frac{15}{4} = 15.4^{-1} \bmod 17 = 15.13 \bmod 17 = 8 \Rightarrow \begin{cases} x_3 = 9 \\ y_3 = 12 \end{cases}$$

$$\Rightarrow 4P(9,12)$$

+ Điểm $5P = P + 4P$

$$\lambda = \frac{15}{9} = 15.9^{-1} \bmod 17 = 15.6 \bmod 17 = 5 \Rightarrow \begin{cases} x_3 = 16 \\ y_3 = 4 \end{cases}$$

$$\Rightarrow 5P(16,4)$$

+ Điểm $6P = P + 5P$

$$\lambda = \frac{3}{16} = 3.16^{-1} = 3.16 \bmod 17 = 14 \Rightarrow \begin{cases} x_3 = 10 \\ y_3 = 12 \end{cases}$$

$$\Rightarrow 6P(10,12)$$

+ Điểm $7P = P + 6P$

$$\lambda = \frac{11}{10} = 11.10^{-1} = 11.12 \bmod 17 = 13 \Rightarrow \begin{cases} x_3 = 6 \\ y_3 = 6 \end{cases}$$

$$\Rightarrow 7P(6,6)$$

+ Điểm $8P = P + 7P$

$$\lambda = \frac{5}{6} = 5.6^{-1} = 5.3 \bmod 17 = 15 \Rightarrow \begin{cases} x_3 = 15 \\ y_3 = 12 \end{cases}$$

$\Rightarrow 7P\left(15,12\right)$

+ Điểm $9P = P + 8P$

$$\lambda = \frac{11}{15} = 11.15^{-1} = 11.8 \bmod 17 = 3 \Rightarrow \begin{cases} x_3 = 11 \\ y_3 = 0 \end{cases}$$

$\Rightarrow 9P\left(11,0\right)$

Từ điểm $10P$ trở đi có thể tính theo quy tắc sau: $P\left(x,y\right) = -P\left(x,-y\right)$

+ Điểm $10P = -8P \Rightarrow 10P\left(15,-12\right) = 10P\left(15,5\right)$. (Chú ý: $-12 \bmod 17 \equiv 5 \bmod 17$)

+ Điểm $11P = -7P \Rightarrow 11P\left(6,-6\right) = 11P\left(6,11\right)$.

+ Điểm $12P = -6P \Rightarrow 12P\left(10,-12\right) = 12P\left(10,5\right)$.

+ Điểm $13P = -5P \Rightarrow 13P\left(16,-4\right) = 13P\left(16,13\right)$.

+ Điểm $14P = -4P \Rightarrow 14P\left(9,-12\right) = 14P\left(9,5\right)$.

+ Điểm $15P = -3P \Rightarrow 15P\left(4,-16\right) = 15P\left(4,1\right)$.

+ Điểm $16P = -2P \Rightarrow 16P\left(13,-1\right) = 16P\left(13,16\right)$.

+ Điểm $17P = -P \Rightarrow 17P\left(0,-1\right) = 17P\left(0,16\right)$.

+ Điểm $18P = 0$.

Các điểm nguyên thủy của nhóm nhân được xác định như sau:

Nếu P là một điểm nguyên thủy thì kP sẽ là nguyên thủy với $\left(k,\left|E_p\left(a,b\right)\right|\right) = 1$

Trong ví dụ này ta có $\left|E_{17}\left(1,1\right)\right| = 18$ vậy các điểm nguyên thủy kP có $\left(k,18\right) = 1$, hay $k = \left\{1,5,7,11,13,17\right\}$.

Các điểm nguyên thủy là: $P, 5P, 7P, 11P, 13P, 17P$ (6 điểm)

3.8.4. Các hệ mật trên đường cong elliptic

3.8.4.1. Trao đổi thỏa thuận khóa Diffie - Hellman

Xét nhóm $E_p\left(a,b\right)$ trên đường cong elliptic và P là điểm nguyên thủy $P \in E_p\left(a,b\right)$. Quá trình trao đổi khóa diễn ra như sau:

A	B
+ A chọn ngẫu nhiên x với điều kiện: $\left(1 < x < \#E_p\left(a,b\right)\right)$ sau đó A tính xP và gửi cho B.	+ B chọn ngẫu nhiên y với điều kiện: $\left(1 < y < \#E_p\left(a,b\right)\right)$ sau đó A tính yP và gửi cho A.

+ A nhận yP và tính ra khóa dùng chung: $K = x(yP) = xyP$	+ B nhận xP và tính ra khóa dùng chung: $K = y(xP) = xyP$

Ví dụ 3.32: Xét $E_{17}(1,1)$ và điểm nguyên thủy $P(0,1)$ (Nhóm $E_{17}(1,1)$ như đã tính ở ví dụ 3.31)

A	B
+ A chọn ngẫu nhiên $x = 3$ và tính: $3P = (4,6)$ và gửi cho B.	+ B chọn ngẫu nhiên $y = 5$ và tính: $5P = (16,4)$ và gửi cho A.
+ A nhận $5P$ và tính ra khóa dùng chung: $K = 3(5P) = 15P(4,1)$	+ B nhận $3P$ và tính ra khóa dùng chung: $K = 5(3P) = 15P(4,1)$

3.8.4.2. Hệ mật Omura – Massey

Tạo khóa:

+ Hai bên liên lạc chọn trước nhóm $E_p(a,b)$ làm khóa công khai

+ Khóa bí mật tạo như sau:

- A chọn ngẫu nhiên hai số m,n làm khóa bí mật, sao cho $m+n = \#E_p(a,b)$

- B chọn ngẫu nhiên hai số u,v làm khóa bí mật, sao cho $u+v = \#E_p(a,b)$

Giả sử A cần gửi bản tin M cho B, quá trình truyền tin bảo mật diễn ra như sau:

$A_{(m,n)}$	$B_{(u,v)}$
+ A tính $mP+M$ và gửi cho B	+ B nhận $mP+M$ và tính $(mP+M)+uP$ và gửi lại A.
+ A nhận $(mP+M)+uP$ và tính: $[(mP+M)+uP]+nP = M+uP$ và gửi cho B.	+ B nhận $M+uP$ và giải mã ra bản tin: $(M+uP)+vP = M$

Ví dụ 3.33:

Xét $E_{17}(1,1)$ và điểm nguyên thủy $P(0,1)$ (Nhóm $E_{17}(1,1)$ như đã tính ở ví dụ 3.31). Bản tin $M(4,16) = 3P$ (Chú ý phải có phép biến đổi bản tin thành một trong các điểm thuộc $E_{17}(1,1)$)

+ Khóa công khai: $E_{17}(1,1)$

+ Khóa bí mật:

- A chọn ngẫu nhiên hai số $(m,n)=(3,15)$ với $m+n=18=\#E_{17}(1,1)$

- B chọn ngẫu nhiên hai số $(u,v)=(5,13)$ với $u+v=18=\#E_{17}(1,1)$

Quá trình truyền tin bảo mật:

$A_{(3,15)}$	$B_{(5,13)}$
+ A tính $mP+M=3P+3P=6P(10,12)$ và gửi cho B	+ B nhận $6P(10,12)$ và tính $6P+5P=11P(6,11)$ và gửi lại A.
+ A nhận $11P(6,11)$ và tính: $11P+15P=8P(15,12)$ và gửi cho B.	+ B nhận $8P(15,12)$ và giải mã ra bản tin: $8P+13P=3P(4,16)=M$

3.8.4.3. Hệ mật Elgamal

a) Tạo khóa

Mỗi bên liên lạc tạo cho mình một cặp khóa công khai – bí mật theo các bước sau:

Bước 1: Chọn nhóm $E_p(a,b)$ và điểm nguyên thủy P

Bước 2: Chọn ngẫu nhiên a với $(1<a<\#E_p(a,b))$ và tính $aP=Q$

Bước 3: + Khóa công khai: $(E_p(a,b),P,Q)$

+ Khóa bí mật: a

b) Mã hóa

Giả sử B cần gửi bản tin M cho A, B thực hiện các bước sau:

Bước 1: B nhận khóa công khai cả A : $(E_p(a,b),P,Q)$

Bước 2: B chọn k ngẫu nhiên $(1<k<\#E_p(a,b))$ và tính:

$$\begin{cases}\gamma=kP\\ \delta=M+kQ\end{cases}$$

Bước 3: B gửi bản mã $C=(\gamma,\delta)$ cho A.

c) Giải mã

A nhận bản mã $C=(\gamma,\delta)$ và giải mã theo các bước:

Bước 1: A tính $a\gamma=akP=kQ$

Bước 2: A tính $\delta - kQ = M + kQ - kQ = M$

Ví dụ 3.34: Xét $E_{17}(1,1)$ và điểm nguyên thủy $P(0,1)$ (Nhóm $E_{17}(1,1)$ như đã tính ở ví dụ 3.31). Bản tin $M(15,5) = 10P$

** Bên A tạo khóa:*

Bước 1: A chọn $a = 3$ và tính $aP = 3P(4,16) = Q$

Bước 2: + Khóa công khai của A: $(E_{17}(1,1), P(0,1), Q(4,16))$

+ Khóa bí mật: $a = 3$

** Mã hóa:*

B gửi bản tin $M = 10P(15,5)$ cho A

Bước 1: B nhận khóa công khai cả A : $(E_{17}(1,1), P(0,1), Q(4,16))$

Bước 2: B chọn $k = 5$ ngẫu nhiên và tính:

$$\begin{cases} \gamma = kP = 5P(16,4) \\ \delta = M + kQ = 10P + 5Q = 10P + 5 \times 3P = 7P(6,6) \end{cases}$$

Bước 3: B gửi bản mã $C = (\gamma, \delta) = (5P, 7P)$ cho A.

** Giải mã:*

A nhận bản mã $C = (5P, 7P)$ và giải mã theo các bước:

Bước 1: A tính $a\gamma = 3 \times 5P = 15P(4,1)$

Bước 2: A tính $\delta - kQ = 7P - 15P = 10P(15,5) = M$

3.8.5. Độ an toàn của hệ mật trên đường cong Elliptic

Sức mạnh ECC nằm ở sự khó khăn đối với thám mã khi phải xác định số ngẫu nhiên bí mật k từ kP và P. Phương pháp nhanh nhất để giải bài tóan này là phương pháp phân tích S - Pollard. Để phá ECC độ phức tạp tính toán khi dùng phương pháp S – Pollard là $3,8.10^{10} MIPS$ - năm với kích thước khóa 150 bit (đây là số năm cần thiết với một hệ thống tính toán có tốc độ hàng triệu lệnh/giây). Để so sánh với phương pháp nhanh nhất phá RSA (là phương pháp sàng trừng số để phân tích hợp số n thành tích của 2 số nguyên tố p và q) ta thấy rằng với n có kích thước 768 bit độ phức tạp tính toán là: $2.10^8 MIPS$ - năm , với n có kích thước 1024 bit, độ phức tạp tính toân là 3.10^{11} năm .

Nếu độ dài khóa của RSA tăng lên tới 2048 bit thì cần $3.10^{20} MIPS$ - năm, trong khi đó với ECC chỉ cần độ dài khóa là 234 bit đã phải yêu cầu tới $1,6.10^{28} MIPS$ - năm.

3.9. ƯU VÀ NHƯỢC ĐIỂM CỦA MẬT MÃ KHÓA CÔNG KHAI

3.9.1. Ưu điểm

+ Mật mã khóa công khai không sử dụng kênh an toàn riêng để truyền khóa.

+ Dễ sinh khóa, bảo vệ và lưu trữ khóa.

+ Dễ xây dựng các dịch vụ an toàn khác như: xác thực, chữ ký số…

So sánh với mật mã khóa bí mật như bảng sau (giả sử ta xét n người dùng trên mạng):

Bảng 3.12. So sánh số lượng khóa của mật mã khóa bí mật và mật mã khóa công khai

n người dùng	Số khóa cần tạo	Số khóa cần lưu trữ
Mật mã khóa bí mật	$\frac{n(n-1)}{2}$	$n-1$
Mật mã khóa công khai	$2n$	1

3.9.2. Nhược điểm

+ Nói chung mật mã khóa công khai phức tạp (phải xây dựng trên các trường lớn, mạch điện phức tạp, thời gian xử lý chậm hơn so với mật mã khóa bí mật)

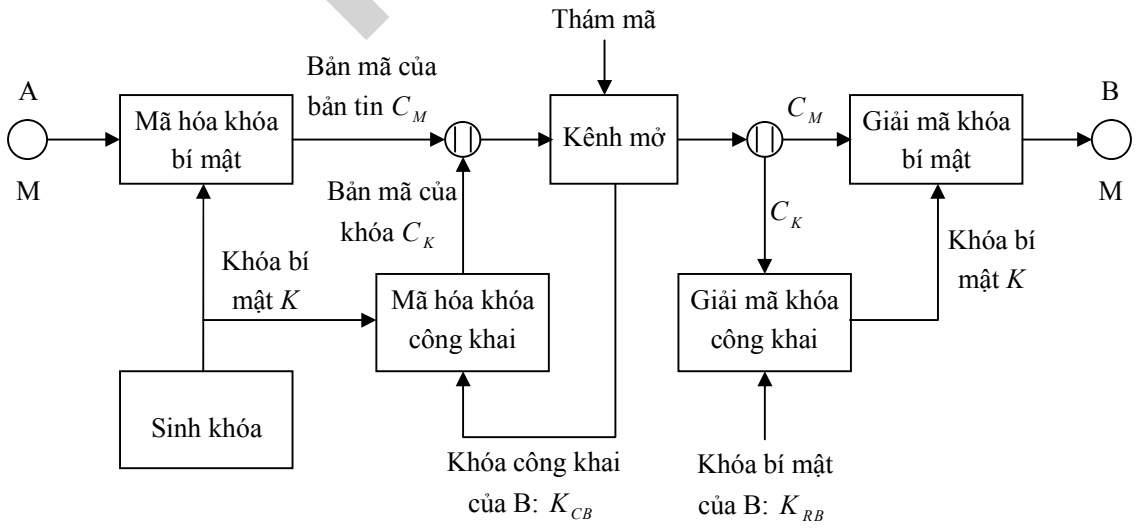
+ Hiệu quả không cao (nói chung các hệ mật khóa công khai có tốc độ mã thấp)

Từ các nhược điểm này ta thấy mật mã khóa công khai khó thực hiện cho các dịch vụ nhạy cảm với độ trễ hoặc các dịch vụ di động.

3.10. XÂY DỰNG CÁC CHƯƠNG TRÌNH ỨNG DỤNG KIẾN TRÚC PGP

(PGP – Pretty Good Privacy: tạm dịch là "Riêng tư tốt đẹp")

Từ các ưu và nhược điểm của mật mã khóa công khai và mật mã khóa bí mật, người ta kết hợp ưu điểm của hai loại mật mã này xây dựng nên một cấu trúc truyền tin PGP như Hình 3.9:



Hình 3.9. Sơ đồ kiến trúc PGP

Trong sơ đồ này, bản tin M được truyền đi bảo mật sử dụng mật mã khóa bí mật với khóa là K. Khóa bí mật K được mã hóa bằng hệ mật khóa công khai từ A đến B (sử dụng khóa công khai và bí mật của B).

Việc mã hóa bản tin bằng mật mã khóa bí mật sẽ có ưu điểm tốc độ mã hóa nhanh, hiệu quả cao, còn việc truyền khóa sử dụng mật mã khóa công khai là để thay thế kênh an toàn tiết kiệm chi phí.

BÀI TẬP CHƯƠNG 3

Bài 3.1: Sử dụng thuật toán Euclide mở rộng để tìm ước chung lớn nhất của hai số $a = 1573,\ b = 308$.

Bài 3.2: Hãy tính $3^{32} \bmod 23$ bằng cách dùng thuật toán nhân và bình phương có lặp.

Bài 3.3: Hãy tính các căn bậc hai của $12 \bmod 37$.

Bài 3.4: Tìm tất cả các phần tử nguyên thuỷ của nhóm nhân Z_{19}^*.

Bài 3.5: Tìm phần tử nghịch đảo của 3 trong Z_{31}^*.

Bài 3.6: Với $m, n, s \in N$ và p_i là các số nguyên tố. Hãy chứng minh các tính chất sau của hàm Phi-Euler

a) $\varphi(p^s) = p^s\left(1 - \frac{1}{p}\right)$.

b) $\varphi(m,n) = \varphi(m)\varphi(n)$ nếu $UCLN(m,n) = 1$.

c) $\varphi(n) = m\left(1 - \frac{1}{p_1}\right)...\left(1 - \frac{1}{p_r}\right)$ trong đó $m = p_1^{e_1}...p_r^{e_1}$ là phân tích của m thành tích của thừa số nguyên tố.

Bài 3.7: Hãy tính $\varphi(490)$ và $\varphi(768)$.

Bài 3.8: Giải hệ phương trình đồng dư sau:

$$5x \equiv 20 \bmod 6$$
$$6x \equiv 6 \bmod 5$$
$$4x \equiv 5 \bmod 77$$

Bài 3.9: Hãy dùng thuật toán Euclide mở rộng để tính các phần tử nghịch đảo sau:

a) $17^{-1} \bmod 101$

b) $357^{-1} \bmod 1234$

c) $3125^{-1} \bmod 9987$

Bài 3.10: Ta nghiên cứu một số tính chất của các phần tử nguyên thuỷ:

a) 97 là một số nguyên tố. Hãy chứng minh rằng $x \neq 0$ là một phần tử nguyên thuỷ theo modulo 97 khi và chỉ khi:

$$x^{32} \neq 1 \bmod 97 \text{ và } x^{48} \neq 1 \bmod 97$$

b) Hãy dùng phương pháp này để tìm phần tử nguyên thuỷ nhỏ nhất theo modulo 97.

c) Giả sử p là một số nguyên tố và $p-1$ có phân tích ra luỹ thừa của các nguyên tố sau:

$$p-1 = \prod_{i=1}^{n} p_i^{e_i}$$

Ở đây p_i là các số nguyên tố khác nhau. Hãy chứng tỏ rằng $x \neq 0$ là một phần tử nguyên thuỷ theo modulo p khi và chỉ khi $x^{(p-1)/p_i} \neq 1 \bmod p$ với $1 \leq i \leq n$.

Bài 3.11: Cho $¢_{13}$, biết $\alpha = 2$ là phần tử nguyên thuỷ của $¢^*_{13}$. Hãy:

a) Tìm tất cả các phần tử nguyên thuỷ của $¢^*_{13}$

b) Giải bài toán logarit rời rạc: Tìm $\log_\alpha y$ với α là phần tử nguyên thuỷ, $y \in ¢^*_{13}$

c) Tìm các thặng dư bậc 2 của $¢^*_{13}$

Bài 3.12: Giải bài 3.11 trong các trường hợp sau:

a) $¢_{19}$, biết $\alpha = 2$ là phần tử nguyên thuỷ của $¢^*_{19}$.

b) $¢_{23}$, biết $\alpha = 5$ là phần tử nguyên thuỷ của $¢^*_{23}$.

c) $¢_{29}$, biết $\alpha = 2$ là phần tử nguyên thuỷ của $¢^*_{29}$.

Bài 3.13: Tính khoá chung trong thủ tục thoả thuận khoá Diffie – Hellman với các trường hợp sau: (với p -nguyên tố, α là phần tử nguyên thủy).

a) $p = 11, \alpha = 7 \in Z^*_{11}$.

b) $p = 17, \alpha = 6 \in Z^*_{17}$.

c) $p = 19, \alpha = 10 \in Z^*_{19}$.

d) $p = 23, \alpha = 11 \in Z^*_{23}$.

e) $p = 29, \alpha = 8 \in Z^*_{29}$.

Bài 3.14: Hãy xây dựng hệ mật Omura-Massey để thực hiện việc truyền khoá bí mật k từ A đến B:

- Tính số các trường hợp có thể lựa chọn tham số của hệ
- Lấy 1 ví dụ với khoá được chọn bởi A là $k = 5$

Với các trường hợp sau: $p = 11,13,17,19,23,29$

Bài 3.15: Hãy xây dựng hệ mật ElGamal với mỗi trường hợp sau và thực hiện mã hoá bản tin $M = 7$. (p nguyên tố, α là phần tử nguyên thủy)

a) $p = 11, \alpha = 8$ b) $p = 13, \alpha = 11$

c) $p = 17, \alpha = 10$ d) $p = 23, \alpha = 14$

e) $p = 29, \alpha = 10$ f) $p = 31, \alpha = 12$

Bài 3.16: Ví dụ về hệ mật RSA. Cho $p = 7$ và q = 17.

a) Tính n .

b) Cho e (số mũ mã hoá) bằng 5. Hãy tính số mũ giải mã d .

c) Hãy mã hoá và giải mã cho các số 49 và 12.

Bài 3.17: Người ta biết rằng đối với hệ mật RSA, tập các bản rõ bằng tập các bản mã. Tuy nhiên bạn có cho rằng một số giá trị trong không gian thông báo (bản rõ) là không mong muốn?

Bài 3.18: Tìm n và số mũ giải mã d của mỗi hệ mật RSA có tham số dưới đây và mã hóa bản tin $M = 5$ với mỗi trường hợp tương ứng:

a) $p = 13, q = 17, e = 11$

b) $p = 19, q = 17, e = 149$

c) $p = 19, q = 23, e = 85$

d) $p = 23, q = 29, e = 123$

Bài 3.19: Trong hệ mật Rabin, giả sử $p = 199, q = 211$.

a) Xác định 4 căn bậc hai của $1 \bmod n$, trong đó $n = pq$.

b) Tính bản mã của 32767.

c) Xác định 4 bản giải mã có thể của bản mã trên.

Bài 3.20: Xét trường hợp đơn giản của hệ mật Merkle-Hellman sử dụng phép hoán hoán vị đồng nhất. Giả sử dãy siêu tăng được chọn là (2, 3. 6, 13, 27, 52) giá trị ngẫu nhiên w được chọn là 31, modulo M được chọn là 105.

a) Hãy xác định khoá bí mật.

b) Bản tin ở dạng nhị phân có dạng 011000_110101_101110.

Hãy tính bản mã và hãy giải mã để tìm lại bản tin ban đầu.

Bài 3.21: Đây là một ví dụ về hệ mật ElGamal áp dụng trong $GF\left(3^3\right)$. Đa thức $x^3 + x^2 + 1$ là một đa thức bất khả quy trên $Z_3[x]$ và bởi vậy $Z_3[x]/\left(x^3 + x^2 + 1\right)$ chính là $GF\left(3^3\right)$. Ta có thể gắn 26 chữ cái của bảng chữ cái tiếng Anh với 26 phần tử khác không của trường và

như vậy có thể mã hoá một văn bản thông thường theo cách truyền thống. Ta sẽ dùng thứ tự theo từ điển của các đa thức khác không để thiết lập sự tương ứng.

$A \leftrightarrow 1$	$B \leftrightarrow 2$	$C \leftrightarrow x$
$D \leftrightarrow x+1$	$E \leftrightarrow x+2$	$F \leftrightarrow 2x$
$G \leftrightarrow 2x+1$	$H \leftrightarrow 2x+2$	$I \leftrightarrow x^2$
$J \leftrightarrow x^2+1$	$K \leftrightarrow x^2+2$	$L \leftrightarrow x^2+x$
$M \leftrightarrow x^2+x+1$	$N \leftrightarrow x^2+x+2$	$O \leftrightarrow x^2+2x$
$P \leftrightarrow x^2+2x+1$	$Q \leftrightarrow x^2+2x+2$	$R \leftrightarrow 2x^2$
$S \leftrightarrow 2x^2+1$	$T \leftrightarrow 2x^2+2$	$U \leftrightarrow 2x^2+x$
$V \leftrightarrow 2x^2+x+1$	$W \leftrightarrow 2x^2+x+2$	$X \leftrightarrow 2x^2+2x$
$Y \leftrightarrow 2x^2+2x+1$	$Z \leftrightarrow 2x^2+2x+2$	

Giả sử Bob dùng $\alpha = x$ và a = 11 trong hệ mật ElGamal, khi đó $\alpha^a = x+2$. Hãy chỉ ra cách mà Bob sẽ giải mã cho bản mã sau:

(K, H) (P,X) (N,K) (H, R) (T, F) (V, Y) (E, H) (F, A) (T, W) (J, D) (V, J).

Bài 3.22: Mã BCH (15, 7, 5) có ma trận kiểm tra sau:

$$H = \begin{pmatrix} 1&0&0&0&1&0&0&1&1&0&1&0&1&1&1 \\ 0&1&0&0&1&1&0&1&0&1&1&1&1&0&0 \\ 0&0&1&0&0&1&1&0&1&0&1&1&1&1&0 \\ 0&0&0&1&0&0&1&1&0&1&0&1&1&1&1 \\ 1&0&0&0&1&1&0&0&0&1&1&0&0&0&1 \\ 0&0&0&1&1&0&0&0&1&1&0&0&0&1&1 \\ 0&0&1&1&0&0&0&1&1&0&0&1&1&1&0 \\ 0&1&1&1&1&0&1&1&1&1&0&1&1&1&1 \end{pmatrix}$$

Hãy giải mã cho các vectơ nhận được sau bằng phương pháp giải mã theo syndrom:

a) $r = (1,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0)$

b) $r = (1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 0)$

c) $r = (1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 0,\ 1,\ 0,\ 1,\ 1,\ 0,\ 0,\ 0,\ 0)$

CHƯƠNG 5. CÁC THỦ TỤC VÀ CÁC CHÚ Ý TRONG THỰC TẾ KHI SỬ DỤNG MÃ HOÁ

Trong các chương 3 và 4 ta đã xem xét các ví dụ về các hệ thống mật mã được coi là mật. Có hàng trăm phương pháp mã hoá khác nhau ngoài DES và RSA là hai hệ mật khoá công khai và khoá bí mật được thừa nhận rộng rãi nhất trong thực tế.

Tuy nhiên chỉ đơn giản là có và sử dụng một hệ mật mạnh là chưa đủ để đảm bảo mọi giao dịch sử dụng hệ mật đó được đảm bảo an toàn. Có những cách thức đúng hoặc không đúng khi sử dụng các phương pháp mã hoá. Hơn nữa các thuật toán này có thể được dùng để giải quyết các vấn đề mà bí mật hoặc xác thực chỉ là một phần của nó. Trong chương này ta sẽ nghiên cứu và đánh giá các kỹ thuật sử dụng mã hoá để thiết lập một kênh liên lạc mật giữa hai người dùng. Ta cũng khai thác các cách thích hợp để sử dụng mã hoá

5.1. CÁC THỦ TỤC: HÀNH VI CÓ THỨ TỰ

Các hệ thống mã hoá là một công cụ quan trọng trong an toàn máy tính, chúng cho phép bạn phát thông tin ở dạng được che dấu. Chúng được dùng để phát các tài liệu và số liệu trên một kênh có thể bị thu chặn. Bằng cách sử dụng các quy ước được thiết lập giữa hai bên với hệ mật có thể được dùng cho những mục đích khác với việc liên lạc an toàn. Các quy ước này được gọi là các thủ tục, chúng sẽ là chủ đề của phần sau:

5.1.1. Định nghĩa thủ tục

Một thủ tục là một dãy các bước có thứ tự mà hai bên (hoặc nhiều hơn) phải thực hiện để hoàn tất một công việc nào đó. Các bên sử dụng thủ tục phải nhất trí hoàn tất thủ tục trước khi dùng nó. Thứ tự của các bước cũng quan trọng như hoạt động của mỗi bước. Người ta sử dụng các thủ tục để điều chỉnh hành vi và quyền lợi chung.

Sử dụng điện thoại là một ví dụ đơn giản của một thủ tục. Người quay số sẽ nghe cả âm chuông và tiếng nhấc máy khi kết nối được thiết lập. Trong thực tế, thủ tục ở đây là người nhận sẽ nói trước (chẳng hạn "xin chào", hoặc "tôi nghe đây"). Người ta sẽ trả lời bằng một lời chào giới thiệu bản thân. Hai bên sau đó sẽ lần lượt trao đổi. Không có thủ tục này cả hai bên có thể nói cùng một lúc khi kết nối được thiết lập và không một ai có thể nghe được người khác.

Tương tự như ví dụ trên ta có thể thấy rằng một thủ tục tốt sẽ có các đặc tính sau:

- Được thiết lập trước : Thủ tục được thiết kế hoàn chỉnh trước khi nó được sử dụng.
- Có sự thống nhất chung : Một thành viên nhất trí tuân thủ các bước trong thủ tục theo thứ tự.
- Không nhập nhằng : Không một ai có thể thực hiện sai một bước do không hiểu nó.

- Hoàn chỉnh : Đối với mọi tình huống có thể xảy ra đều phải có một hành độngđược mô tả trước cần thực hiện.

Các thủ tục cũng được dùng trong liên lạc giữa máy tính với máy tính. Một máy tính cần phải biết khi nào "nói" và "nghe" với máy đang liên lạc với nó và liệu nó đã nhận đủ thông tin chưa… Hiển nhiên là cả hai máy tính phải tuân theo cùng một thủ tục.

5.1.2. Các loại thủ tục

Các nhiệm vụ nhất định, chẳng hạn như thoả thuận hợp đồng, bầu cử, phân phối thông tin và thậm chí là chơi bài đều là các hoạt động của con người. Tuy nhiên nhiều nhiệm vụ kiểu này phụ thuộc vào người làm chứng để đảm bảo vị công bằng. Liệu bạn có tin vào một người nói rằng anh ta sẽ xóc các quân bài mà không nhìn vào chúng và đưa cho bạn? Liệu bạn có tin vào một người nếu bạn không quen biết và nếu số tiền đặt cược cao?

Xã hội hiện đại đòi hỏi việc sử dụng máy tính và liên lạc như những công cụ thương mại. Nhiều người sử dụng máy tính không có sự quen biết cá nhân đối với người quản lý và những người sử dụng khác trong hệ thống. Trong nhiều trường hợp việc liên lạc máy tính được thực hiện trên những khoảng cách lớn. Do tính vô danh và do khoảng cách người dùng sẽ không tin vào các nhà quản lý và những ngườidùng khác trong hệ thống. Để sử dụng máy tính một cách hiệu quả ta phải phát triển các thủ tục mà nhờ chúng hai người đa nghi có thể giao tiếp với nhau và tin vào sự công bằng.

Hơn nữa để điều chỉnh hành vi, các thủ tục còn phục vụ cho một mục đích rất quan trọng khác là các thủ tục phải tách quá trình hoàn tất một nhiệm vụ khỏi cơ chế thực thi nó. Một thủ tục sẽ chỉ xác định các quy tắc của hành vi. Bằng cách này ta có thể kiểm tra một thủ tục để tin rằng nó đạt kết quả mong muốn. Ta sẽ kiểm tra tính đúng đắn của quá trình ở mức cao.

Sau khi đã tin vào tính đúng đắn của thiết kế ta có thể áp dụng thủ tục bằng cách dùng một cơ chế nào đó (tức là dùng một ngôn ngữ riêng nào đó hoặc một hệ thống mã hoá). Áp dụng phải tách khỏi thiết kế. Bởi vậy ta chỉ cần kiểm tra rằng cơ chế sẽ phải ảnh thiết kế đúng đắn, ta không cần kiểm tra lại rằng ứng dụng sẽ giải quyết vấn đề mà thủ tục được thiết kế cho nó. Hơn nữa, sau này ta có thể thay đổi ứng dụng mà không ảnh hưởng tới thiết kế. Việc tách rời thiết kế khỏi các ứng dụng là một ưu điểm quan trọng trong việc dùng các thủ tục.

5.1.3. Các thủ tục có trọng tài

Trọng tài là một bên thứ ba vô tư được tin cậy để hoàn tất một giao dịch giữa hai bên không tin cậy nhau. Nếu bạn bán một chiếc xe cho một người lạ và anh ta đưa cho bạn một tấm séc thì bạn không có cách nào để biết rằng tấm séc này có giá trị không. Bạn muốn gửi tấm séc này và giữ xe lại trong ít ngày cho tới khi bạn tin chắc rằng séc không có vấn đề gì. Một người mua đa nghi sẽ không chịu như vậy vì bạn lại có cả xe và séc và biết đâu bạn có thể chuồn khỏi thành phố với chúng?

Giải pháp ở đây là sử dụng một bên thứ ba được tín nhiệm chẳng hạn một chủ nhà băng hoặc một luật sư làm trọng tài. Bạn trao chứng nhận sở hữu xe và chìa khoá xe cho trọng tài và người mua xe đưa séc cho trọng tài. Bạn có một sự đồng ý tay ba vào thời điểm séc được

xác nhận. Trọng tài sẽ gửi séc vào tài khoản của bạn. Nếu trong một thời hạn xác định séc được xác nhận thì trọng tài sẽ chuyển xe của bạn cho người mua. Nếu séc không được xác nhận thì trọng tài sẽ trả xe về cho bạn. Trong một thủ tục máy tính, trọng tài là một bên thứ ba tin cậy đủ đảm bảo sự công bằng. Trọng tài có thể là một người, một chương trình hoặc một máy tính. Ví dụ trong một mạng, trọng tài có thể là một chương trình chạy trên một máy trong mạng. Chương trình sẽ nhận và gửi các thông báo giữa các người dùng. Người dùng tin rằng khi trọng tài gửi tới một thông báo nói rằng nó tới từ A thì thông báo thực sự được gửi từ người dùng A. Khái niệm trọng tài là một khái niệm cơ bản đối với loại thủ tục an toàn được gọi là thủ tục có trọng tài.

Các thủ tục máy tính có trọng tài có một số nhược điểm:

- Hai bên không có khả năng tìm một bên thứ ba vô tư mà cả hai đều tin tưởng. Những người dùng đa nghi có lý khi họ không tin vào một trọng tài không được biết trên mạng.
- Việc duy trì hoạt động của trọng tài sẽ làm tăng chi phí cho những người sử dụng hoặc mạng chi phí này có thể rất lớn.
- Trọng tài sẽ gây nên thời gian giữ chậm khi liên lạc vì bên thứ ba phải thu, xem xét và rồi mới gửi đi đối với mỗi giao dịch.
- Nếu dịch vụ trọng tài được dùng quá nặng nề thì nó có thể trở thành một "nút cổ chai " trong mạng vì nhiều người dùng đều muốn có trọng tài riêng
- Tính bí mật trở nên dễ bị tổn thương vì trọng tài phải truy nhập tới nhiều thông tin nhạy cảm.

Vì những lý do trên mà người ta thường tránh dùng thủ tục này nếu có thể.

5.1.4. Các thủ tục có phán xét

Tương tự như trọng tài là ý tưởng sử dụng quan toà. Quan toà là một bên thứ ba có thể phán xét liệu một giao dịch có được thực hiện một cách công bằng hay không. Ví dụ công chứng viên là một bên thứ ba vô tư được tin cậy sẽ chứng thực rằng một tài liệu đã được ký một cách tự nguyện và xác nhận rằng anh ta có đầy đủ lý do để xác định rằng người ký là có thẩm quyền. Chữ ký của công chứng viên thường cần thiết đối với những tư liệu hợp lệ mà tính xác thực của nó sau này có thể bị nghi ngờ. Công chứng viên không thêm một tý gì vào giao dịch ngoài việc là một người làm chứng - một người sau này có thể kiểm tra khi có sự nghi ngờ.

Một số thủ tục máy tính sử dụng một kiểu tương tự công chứng viên để xây dựng bằng chứng công bằng. Với một thủ tục có khả năng phán xét, số liệu đủ là cần thiết để bên thứ ba vô tư phán xét được tính công bằng dựa trên bằng chứng. Bên thứ ba không chỉ có thể xác định liệu hai bên tranh chấp có sử sự đúng không (tức là nằm trong các quy tắc của thủ tục) mà còn có thể xác định được ai là người gian lận.

Các thủ tục có phán xét sẽ xoay quanh các dịch vụ của bên thứ ba chỉ trong trường hợp có tranh chấp. Bởi vậy các thủ tục này có chi phí thấp hơn các thủ tục có trọng tài. Tuy nhiên chúng chỉ xác định được sai sót sau khi sai sót xảy ra.

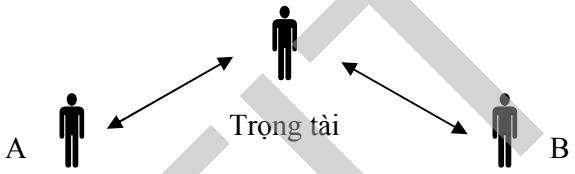
5.1.5. Các thủ tục tự ràng buộc

Một thủ tục tự ràng buộc là một thủ tục đảm bảo sự công bằng. Nếu một bên nào đó cố gắng gian lận thì điều này phải được ben kia thấy rõ. Không cần bất cứ một người ngoài nào để đảm bảo sự công bằng. Hiển nhiên là các thủ tục tự ràng buộc được ưu thích hơn cả. Tuy nhiên, không có một thủ tục tự ràng buộc đối với mọi tình huống.

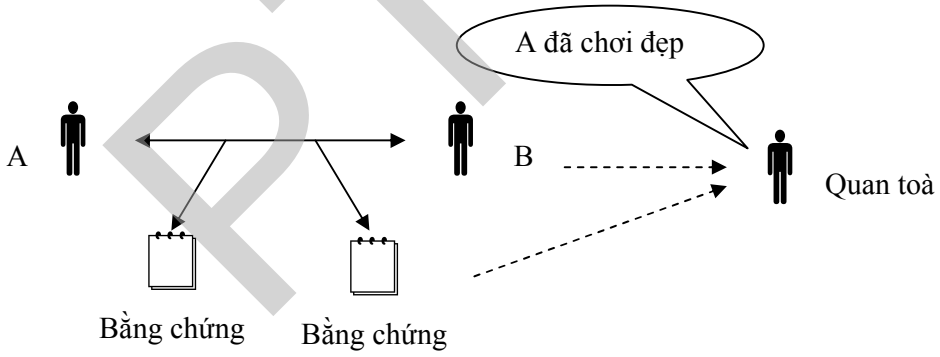
Như vậy có ba mức các thủ tục:

- *Các thủ tục có trọng tài* trong đó một bên thứ ba tin cậy sẽ tham gia vào mỗi giao dịch để đảm bảo rằng cả hai bên đều sử sự đúng đắn.
- *Các thủ tục có sự phán xét* trong đó một bên thứ ba có thể phán xét liệu cả hai bên có sử sự đúng đắn hay không và nếu không thì bên nào là gian lận.
- *Các thủ tục tự ràng buộc* trong đó những mưu toan gian lận của một bên nào đó sẽ bị bên khác hiện.

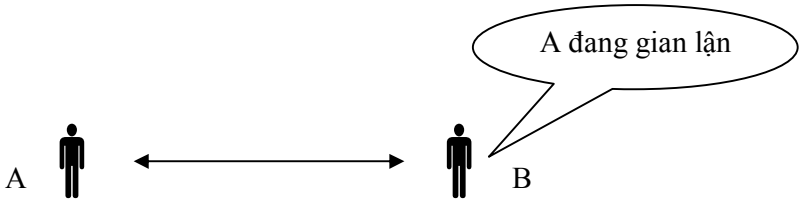
Các loại thủ tục này được mô tả ở hình 5.1.



a) Các thủ tục có trọng tài



b) Các thủ tục có phán xét



c) Các thủ tục ràng buộc

Hình 5.1. Các loại thủ tục

Sau đây ta sẽ hướng mối quan tâm vào việc sử dụng các thủ tục để giải quyết các bài toán trong an toàn mạng máy tính.

5.2. CÁC THỦ TỤC ĐỂ GIẢI QUYẾT CÁC VẤN ĐỀ

Bây giờ ta sẽ nghiên cứu việc sử dụng các thủ tục trên những bài toán thực tế. Rõ ràng là khi hai người tiếp xúc trực tiếp họ sẽ làm khác với khi có một máy tính giữa họ. Mặc dù có thể có những khác biệt về quan điểm nhưng chúng tôi vẫn muốn đưa ra những thủ tục cho một môi trường tự động mà nhờ nó con người có thể tiến hành các công việc hàng ngày như ký các hợp đồng, chi trả các hoá đơn, bỏ phiếu bầu cũng an toàn như các giao tiếp giữa người với người. Ta cũng nghiên cứu một số vấn đề mà có các thủ tục an toàn cho chúng.

5.2.1. Phân phối khoá

Thay đổi khoá mã hoá là một vấn đề quan trọng nhưng không dễ. Có thể thấy rằng thám mã sẽ càng có cơ may thành công nếu có càng nhiều bản mã. Bản mã từ các khoá khác nhau có thể giúp cho việc xác định cấu trúc của một thuật toán bí mật hoặc các khoá bí mật nhưng bản mã từ cũng "một khoá" lại giúp cho việc tìm giá trị của khoá. Như vậy, trên thực tế cần thay đổi khoá một cách định kỳ để đảm bảo lượng bản mã tạo từ một khoá bất kỳ không giúp ích đáng kể cho thám mã.

Tuy nhiên ta hãy xem xét một việc sử dụng mã hoá cho các chữ ký số trên các hợp đồng. Nếu bạn đưa ra hoặc thu nhận một hợp đồng đã được ký bằng chữ ký số thì bạn muốn giữ hợp đồng làm bằng chứng cho tới khi mọi tranh chấp có thể có đã được giải quyết . Điều này có thể là phải nhiều năm sau khi hợp đồng đã được thực hiện . Người gửi và người nhận cần phải giữ mọi khoá cần thiết để kiểm chứng giá trị các chữ ký của hợp đồng. Nếu người gửi thay đổi các khoá mã hoá hàng ngày hoặc thậm chí là hàng tháng thì vẫn phải lưu giữ hàng trăm hoặc hàng nghìn khoá. Và những khoá này phải được lưu giữ một cách an toàn. Điều này cho thấy không nên thay đổi khoá thường xuyên hơn mức cần thiết. Mặt khác, nếu một khoá bị mất hoặc bị lộ thì càng dùng ít khoá nguy cơ này càng ít. Bởi vậy việc chọn một tần suất thay đổi khoá thích hợp không phải là một công việc dễ dàng.

Một số thủ tục đã được phát triển cho việc phân phối khoá. Các thủ tục này phụ thuộc vào mức độ chia sẻ thông tin giữa người gửi và người nhận.

5.2.1.1. Trao đổi khoá đối xứng không có Server (máy chủ)

Giả sử rằng hai người sử dụng đều có một bản sao của khoá mã hoá đối xứng (bí mật) *K* mà chỉ có họ biết. Khi lượng thông báo không lớn lắm hoặc nếu không có nguy cơ xâm nhập đáng kể thì họ có thể dùng *K* để trao đổi các thông báo.

Tuy nhiên nếu hai người muốn an toàn hơn họ có thể đồng ý thay đổi khoá thường kỳ thậm chí là dùng khoá khác nhau đối với mỗi thông báo. Để làm được điều này một trong hai người có thể tạo một khoá mới (được gọi là Knew) rồi mã hoá nó bằng *K* và gửi *E* (Knew, K) tới người còn lại. Trong một số hệ mật *K* được gọi là khoá chủ hay mã hoá khoá và Knew được gọi là khoá phiên hay khóa lưu chuyển.

Nhược điểm của phương pháp này là hai người dùng phải cùng chia sẻ một khoá duy nhất của họ. Các cặp người dùng khoá cũng cần các khoá duy nhất và nói chung n người dùng phải cần tới $n(n+1)/2$ khoá.

5.2.1.2. Trao đổi khoá đối xứng có Server

Một phương pháp khác là sử dụng dịch vụ phân phối khoá trọng tâm cho 2 người dùng. Ở thủ tục này số khoá sẽ giảm nhưng tính mềm dẻo của giải pháp cũng giảm đi một mức nào đó.

Giả sử Pablo và Rơnê muốn có một khoá bí mật dùng để trao đổi các thông báo nhưng họ không có khoá chung. Tuy nhiên ta cũng coi rằng có một kho chứa khoá trung tâm sao cho Pablo và kho này có một khoá chung K_P, Rơnê và kho có một khoá chung khác K_R. Như đã nêu trong [8], trước tiên Pablo sẽ gửi tới kho (P,R,I_P) báo danh tính của mình (P), danh tính của người nhận (R) và một định danh I_P để đánh dấu kết quả, kho sẽ gửi trở lại thông tin của Pablo (giả sử rằng Pablo có thể gửi nhiều yêu cầu tới kho và cần một định danh để phân biệt) và để tránh các tấn công theo kiểu phát lại (các tấn công dạng này có thể buộc Pablo và Rơnê dùng lại khoá trước). Thông tin này không cần phải mã hoá vì:

- Nếu thông báo được đọc bởi một người ngoài (Chẳng hạn Octavia) thì việc biết rằng Pablo và Rơnê muốn trao đổi thông báo riêng cũng chẳng gây tổn hại gì.
- Nếu thông báo bị thu chặn và bị sửa đổi thì trường hợp xấu nhất có thể xảy ra là yêu cầu sẽ bị thay đổi từ yêu cầu Pablo/ Rơnê sang yêu cầu Pablo/ Octavia, vì vậy sự thay đổi này sẽ được phát hiện ngay ở bước sau.
- Nếu thông báo bị Octavia thu chặn và không phát đi thì Pablo sẽ nhận thấy không có trả lời và sẽ có hành động đối phó.
- Nếu thông báo bị Octavia thu chặn và giả mạo tạo một câu trả lời thì Pablo sẽ nhận thấy rằng câu trả lời không được mã hoá đúng ở bước tiếp.

Trung tâm phân phối khoá sẽ tạo một khoá mã hoá mới cho Pablo và Rơnê để sử dụng (ta gọi là K_{PR}). Trung tâm phân phối sẽ gửi cho Pablo.

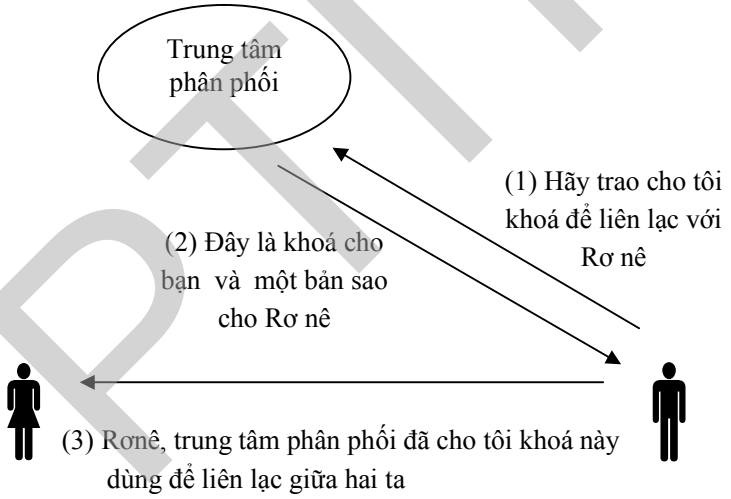
$$E\left(\left(I_P,R,K_{PR},E\left(\left(K_{PR},P\right),K_R\right)\right),K_P\right) \tag{5.1}$$

- Trong phần giao dịch này Pablo sẽ nhận 4 tin sau được mã hoá bằng K_P:
- Định danh thông báo I_P.
- Tính danh của Rơnê R.
- Khoá liên lạc K_{PR}.
- Một xâu bao gồm tính danh của anh ta và cũng khoá đó được mã bằng khoá của trung tâm phân phối chung với Rơnê, $E\left(\left(K_{PR},P\right),K_R\right)$.

Pablo không thể giải mã thông tin cuối cùng này nhưng anh ta có thể gửi nó cho Rơnê. Một lần nữa, các tấn công của Octavia cũng không thành công vì:

- Nếu Octavia thay R bằng O trong thông báo gốc của Pablo để thay đổi tính danh của mình vào vị trí của Rơnê thì trong trả lời từ trung tâm phân phối Pablo sẽ nhận thấy rằng tính danh của Rơnê đã bị thay đổi thành tính danh của Octavia.
- Nếu Octavia thay P bằng O trong thông báo gốc của Pablo để thay tính danh của mình vào bị trí tính danh của Pablo. Khi đó Octavia (chứ không phải là Pablo) sẽ thu được trả lời, còn về phân mình Rơnê sẽ xác định rằng Octavia (chứ không phải là Pablo) là người muốn liên lạc với mình (qua thông báo được mã bằng khoá chỉ có trung tâm phân phối và Rơnê được biết).
- Nếu Octavia thu chặn thông báo từ trung tâm phân phối thì toàn bộ thông báô được mã bằng khoá K_P nên Octavia không thể biết được và cũng không thể thay đổi nó một cách logic.

Cuối cùng Pablo sẽ gửi cho Rơnê $E\left(\left(K_{PR},P\right),K_R\right)$, khoá được xác nhận và được ký bởi trung tâm phân phối là khoá dùng để liên lạc với Pablo. Thủ tục này được chỉ ra trên hình 5.2.



Hình 5.2. Phân phối khoá qua trung tâm phân phối khoá

Thủ tục được đưa ra ở đây đòi hỏi phải luôn có trung tâm phân phối khoá. Việc thêm vào trung tâm này có thể gây ra hiện tượng tắc nghẽn (do nhiều người yêu cầu khoá), nó cũng là một mục tiêu rất hấp dẫn để qua mặt tại sao Octavia lại phải mất thời gian để thu chặn và thay đổi các thông báo riêng nếu như cô ta có thể qua mặt, giả mạo hoặc làm mất hiệu lực của trung tâm phân phối.

Tuy nhiên sơ đồ này tốt khi mở rộng vì khi thêm một người dùng mới chỉ cần thêm một khoá chung vơí trung tâm phân phối khoá. Hơn nữa, người dùng có thể thay đổi khoá thường xuyên theo mong muốn bằng cách đưa ra yêu cầu khoá mới cho trung tâm phân phối. Thủ tục này cũng thích hợp khi Pablo và Rơnê không cần trao đổi liên lạc trước nếu họ đã được ghi nhận ở trung tâm phân phối.

5.2.1.3. Trao đổi khoá không đối xứng không có Server

Sử dụng mã hoá không đối xứng (khoá công khai) sẽ làm giảm nhu cầu đối với các khoá riêng và cũng không làm giảm khả năng dễ bị tấn công của kho chứa trung tâm. Giả sử Pablo và Rơnê muốn trao đổi một thông báo, mỗi người đều có một cặp khoá riêng và khoá công khai và mỗi người đều có thể truy nhập được vào khoá công khai của người khác. Ta ký hiệu các khoá rêng và khoá công khai của Pablo là E_P và D_P, các khoá tương ứng này của Rơnê là E_P và D_P.

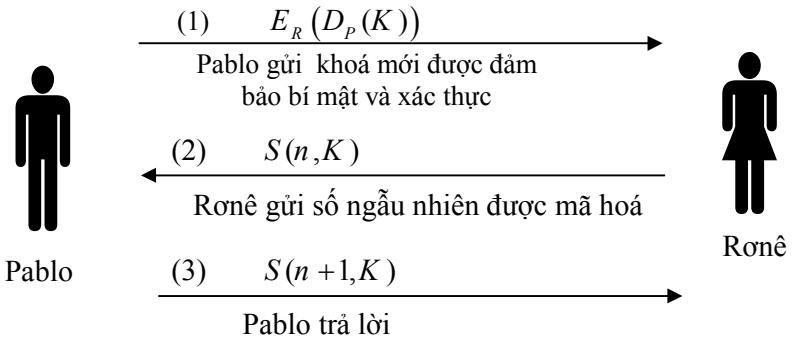
Pablo có thể gửi trực tiếp $E_R(M)$ tới Rơnê. Tuy nhiên các thuật toán mã hoá công khai điển hình thực hiện chậm hơn hẳn so với các thuật toán mã hoá đối xứng (thường kém tới một vài bậc). Bởi vậy, trong khi bản thân việc mã hoá là một gánh nặng đối với tốc độ liên lạc thì mã hoá khoá công khai lại là một gánh nặng tới mức không thể chấp nhận được. Vì lý do đó, Pablo và Rơnê có thể không muốn sử dụng thuật toán khoá công khai để bảo vệ tổng thể quá trình liên lạc.

Pablo có thể chọn một khoá mã hoá K_{PR} của một thuật toán đối xứng S để sử dụng với Rơnê. nếu Pablo gửi $E_R(K_{PR})$ tới Rơnê thì chỉ có Rơnê mới có thể đọc được nó vì chỉ có Rơnê mới có khoá riêng D_R để giải ra khoá. Hơn nữa việc sử dụng mã hoá khoá công khai cho công việc này chỉ trong một thời gian ngắn không ảnh hưởng nhiều tới việc sử dụng các tài nguyên tính toán. Tuy nhiên Rơnê không thể đảm bảo được rằng khoá này thực sự tới từ Pablo ngay cả khi anh ta gửi $E_R(P, K_{PR})$. Những thông báo này có thể bị giả mạo bởi Octavia.

Pablo có thể gửi $E_R(D_P(K_{PR}))$ tới Rơnê. Chỉ có Rơnê mới chỉ có thể giải mã được lớp ngoài, bởi vậy Rơnê được đảm bảo về tính bí mật của thông báo. Tương tự, chỉ có Pablo mới có thể áp dụng mã hoá lớp trong $(D_P(\ ))$, do đó Rơnê được đảm bảo về tính xác thực của nó. Bởi vậy thông báo này sẽ chuyển một khoá xác thực và tin cậy.

Tuy nhiên bằng việc trao đổi khoá đối xứng hai bên cũng trao đổi một thông báo kiểm tra để chứng tỏ rằng mỗi bên đã nhận đúng một khoá mới (tức là không phải một khoá được dùng lại). Sau khi thu khoá của Pablo, Rơnê phải giữ một số ngẫu nhiên đã mã n, Pablo sẽ giải mã số này và để chứng tỏ rằng mình đã làm việc đó anh ta sẽ gửi trả lại bản mã của số $(n+1)$. Thủ tục này được mô tả trên hình 5.3.

Thủ tục này trả lời cho câu hỏi Pablo và Rơnê phải làm thế nào để nhận được các khoá công khai của nhau một cách bí mật. Ta sẽ xét câu trả lời này trong trường hợp có trung tâm phân phối khoá.

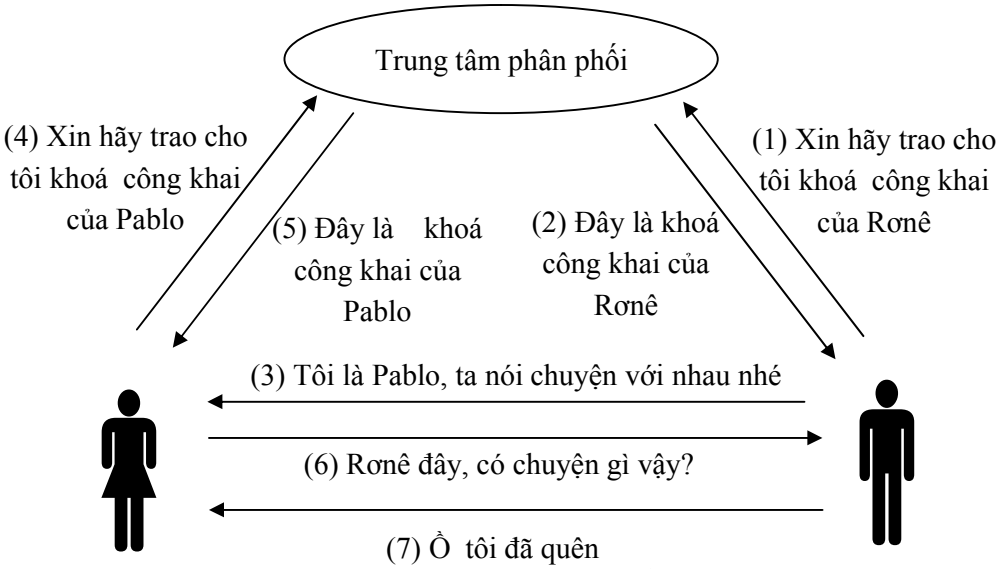


Hình 5.3. Thủ tục trao đổi khoá đối xứng

5.2.1.4. Trao đổi khoá không đối xứng có Server

Thủ tục này là sự mở rộng trực tiếp của thủ tục sử dụng trung tâm phân phối khoá đối xứng. Pablo sẽ gửi một thông báo tới trung tâm phân phối nói rằng anh ta muốn liên lạc với Rơnê (P/R). Như đã nói ở trên, thông báo này có thể sử dụng ở dạng rõ. Trung tâm phân phối sẽ trả lời bằng $D_D(E_R,R)$ (khoá công khai của Rơnê và tính danh của cô ta). Thông báo này được mã bằng khoá riêng của trung tâm phân phối. Không có bí mật nào chứa trong thông báo này ngoài tính toàn vẹn. Bởi vậy, bất cứ ai có khoá công khai của trung tâm phân phối đều có thể đọc nó nhưng chỉ có trung tâm phân phối mới có thể tạo ra được thông báo này. Pablo bây giờ đã có khoá công khai của Rơnê và có thể bắt đầu liên lạc trực tiếp với Rơnê nhưng thủ tục chưa kết thúc. Nếu Pablo gửi một thông báo bằng khoá công khai của Rơnê thì anh ta tin rằng chỉ có Rơnê mới có thể đọc được nó. Tuy nhiên Rơnê không có khoá dùng để trả lời ngay cả khi Pablo gửi khoá công khai của mình kèm theo thông báo, Rơnê chẳng có lý do nào để tin rằng đó là khoá của Pablo chứ không phải là của Octavia.

Pablo sẽ gửi $E_R(P,I_P)$ tới Rơnê chứa tính danh của anh ta và một thông tin phụ dùng cho thông báo trả lời của Rơnê. Bây giờ Rơnê sẽ liên lạc với trung tâm phân phối theo cách mã Pablo đã làm để nhận được khoá của anh ta. Rơnê gửi (R,P) và nhận trở lại $D_D(E_R,R)$. Cô ta gửi cho Pablo $E_P(I_P,I_R)$ để cho Pablo biết rằng cô ta đã thu được thông báo của ta, đã nhận được khoá công khai của anh ta và đang sẵn sàng liên lạc. I_P báo cho Pablo biết rằng thông báo này tới từ Rơnê (bởi vì anh ta đã gửi nó cho Rơnê sau khi đã mã nó bằng mã công khai của Rơnê). Pablo sẽ hoàn tất thủ tục bằng việc gửi cho Rơne thông báo M và gửi I_R để chứng tỏ rằng thông báo này thật sự là từ Pablo, tức là nó có sau khi cô ta đã có khoá công khai của Pablo nhận từ trung tâm. Tóm lại, Pablo sẽ gửi cho Rơnê $E_R(M,I_R)$ (Thông thường M là một khoá mã hoá đối xứng cho cả hai người sử dụng. Nếu không, họ có thể tiếp tục gửi các tính danh I để tiếp tục đảm bảo tính xác thực). Thủ tục này có 7 bước như được mô tả trên hình 5.4.



Hình 5.4. Thủ tục trao đổi khoá không đối

Trung tâm phân phối làm thế nào để có được các khoá? Trung tâm phân phối có thể công bố khoá công khai của mình một cách rộng rãi và bất kỳ ai muốn ghi nhận vào trung tâm này chỉ cần đưa khoá và tính danh đã được mã bằng khoá của trung tâm. Do hiệu năng, độ tin cậy và kích cở mà có thể có nhiều trung tâm phân phối. Ở các bước 2 và 5, nếu trung tâm phân phối không có các khoá được yêu cầu thì nó có thể dàn xếp với các trung tâm khác để thu nhận và phân phối lại các khoá. Người dùng không cần biết quá trình thoả thuận này. Hai hoặc nhiều trung tâm có thể dùng như bộ phận dự phòng của trung tâm khác để nếu một trung tâm bị quá tải hoặc hỏng thì trung tâm khác có thể cung cấp các giá trị khoá như vậy. Cũng như vậy, một khi Pablo và Rơnê có khoá công khai của nhau thì họ có thể tự lưu trữ khoá này, chỉ khi một bên muốn thay đổi khoá công khai thì mới phải yêu cầu tới trung tâm phân phối.

Trung tâm phân phối khoá là một khâu quyết định trong thủ tục: nó phải hoạt động bất cứ khi nào có nhu cầu về khoá. Tuy nhiên, cần chú ý là trung tâm phân phối sẽ gửi cho Rơnê $D_D(E_P, P)$ bao gồm khoá và tính danh của Pablo để đảm bảo rằng khoá này thực sự tới từ trung tâm phân phối (bởi vì đây là kết quả được mã hoá bằng D_D là khoá riêng của trung tâm phân phối). Pablo có thể yêu cầu khoá riêng của mình ở bất cứ thời điểm thích hợp nào và sẽ nhận được kết qủa tương tự từ trung tâm phân phối. Sau đó anh ta có thể chuyển các bản sao của kiểu xác thực này đối với khoá của mình tới bất cứ ai mà anh ta muốn thiết lập liên lạc. Theo cách này, tính khả dụng liên tục của trung tâm phân phối không còn quan trọng nữa.

Câu hỏi còn lại là điều gì làm cho ta tin rằng các khoá là xác thực, tức là các khoá này là đúng của người có tính danh tương ứng. Liệu Octavia có thể đưa ra khoá công khai của mình mà tuyên bố rằng đó là khoá cử Rơnê không? Điều cần nói ở đây là ta phải làm thế nào để ràng buộc một cách chắc chắn tính danh và khoá công khai theo một cách thực sự tin cậy? Câu hỏi này được gọi là chứng thực.

5.2.1.5. Chứng thực

Cũng như mọi người chúng ta sẽ thiết lập sự tin cậy ở mọi lúc. Với con người ta thường xác định họ qua các đặc điểm như giọng nói, khuôn mặt hoặc chữ viết. Tuy nhiên, đôi khi ta

trả lời điện thoại và nghe thấy những câu đại loại: "Tôi là đai diện của chính quyền địa phương…" hoặc "Tôi thay mặt tổ chức từ thiện….". Tuỳ theo nội dung của cuộc gọi mà ta quyết định liệu có nên tin vào người gọi hay phải tìm một cách kiểm tra độc lập chẳng hạn như nhờ cảnh sát kiểm tra số máy đã gọi đến trong danh bạ điện thoại. Ta cũng có thể tìm kiếm thông tin phụ từ người gọi: ("Cô ta mặc áo màu gì"). Ta có thể hành động để ngăn chặn kẻ lợi dụng ("Tôi sẽ gửi séc trực tiếp tới tổ chức từ thiện của bạn"). Tuy nhiên đối với những công việc giao tiếp kiểu này ta đều có một ngưỡng tin cậy nào đó, một mức độ để ta có thể sẵn lòng tin tưởng vào một người chưa được biết.

Đối với liên lạc điện tử ta cần phải xây dựng những phương pháp để hai bên thiết lập được sự tin cậy mà không phải gặp gỡ. Trong thương mại rất cần những ứng dụng này. Công ty chế tạo máy Acorn gửi tới cho công ty thép Big một đơn đặt hành 10.000 tấm thép cần được đóng hàng xong trong một tuần và sẽ trả tiền trong vòng 10 ngày. Đơn đặt hàng được in theo mẫu của Acorn và được ký bởi một người có tên là Helen Smudge - đại diện thương mại. Big có thể bắt đầu chuẩn bị thép. Big có thể kiểm tra mức độ tài khoản của Acorn, để quyết định xem có nên gửi hàng mà không được trả tiền trước hay không. Nếu nghi ngờ, Big có thể điện thoại cho Acorn và yêu cầu nói chuyện với bà Smudge ở bộ phận thương mại.

Tuy nhiên thường là Big sẽ đóng hàng mà không cần biết ai sẽ là bà Smudge, liệu bà ta có phải là đại diện thương mại không và liệu bà ta có thẩm quyền để đặt những hợp đồng có tầm cỡ như vậy hay không. Đôi khi những giao dịch kiểu này được thực hiện qua Fax nên Big thậm chí cũng không có cả chữ ký gốc trên tài liệu. Các giao dịch kiểu này thường xuyên xảy ra. Ở đây sự tin cậy được dựa trên tính xác thực (khuôn dạng bản in, chữ ký), thông tin phụ (báo cáo tài chính) và mức độ khẩn cấp (Acorn yêu cầu có thép nhanh).

Tất cả chúng ta đã từng cho bạn bè vay một thứ gì đó. Giả sử bạn gặp ở một bữa tiệc một người nói rằng cô ta là bạn của một người bạn và cô ta yêu cầu bạn vay tiền. Bạn trao đổi với cô ta một số chi tiết và bạn có thể hài lòng biết rằng cô ta có biết đôi chút về người bạn này. Dựa trên mối quan hệ đó bạn có thể quyết định cho cô ta vay tiền. Sự tin tưởng ở đây là dựa vào vẻ ngoài đáng tin cậy của cô ta và sự quen biết của cô ta với người bạn của bạn. Câu chuyện tương tự có thể tiếp tục với một số người bạn của một người bạn ("Tôi sống cạnh Anand và anh ta thường nói về Clara là người láng giềng của anh, anh có biết cô ấy không?"). Bạn có thể cho một người bạn của một người bạn vay tiền nếu bạn cảm thấy đủ tin cậy với chuỗi tình bạn này. Mối quan hệ "bạn của" không dễ dàng chuyển bắc cầu như vậy.

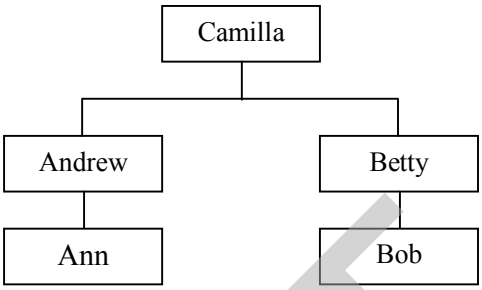
Tuy nhiên, trong thực tế có thể thấy được một số điều đảm bảo trong các ví dụ trên. Cảnh sát sẽ đảm bảo tính xác thực của người gọi. Acorn sẽ đảm bảo rằng bà Smudge là đại diện thương mại của họ (bằng cách chuyển cuộc gọi tới bà ta ở bộ phân thương mại) và theo một nghĩa nào đó, công ty điện thoại sẽ đảm bảo tính xác thực của bộ phận cảnh sát hoặc của Acorn qua danh sách cuả họ trong danh bạ điện thoại. Những "đảm bảo" này có thể được dùng là cơ sở để tin cậy trong các quan hệ thương mại mà hai bên không biết nhau.

Một công ty lớn có thể có một vài bộ phận, mỗi bộ phận có thể có một số phòng, mỗi phòng có thể có một vài đề án và mỗi một đề án có thể có một số nhóm tác nghiệp (với những khác biệt về tên, số mức và cấp độ hoàn chỉnh của cấu trúc). Người điều hành cao nhất có thể không biết tên và mắt của mỗi nhân viên trong công ty nhưng trưởng nhóm tác nghiệp phải

biết tất cả các thành viên trong nhóm mình và lãnh đạo dự án phải biết tất cả các trưởng nhóm tác nghiệp…

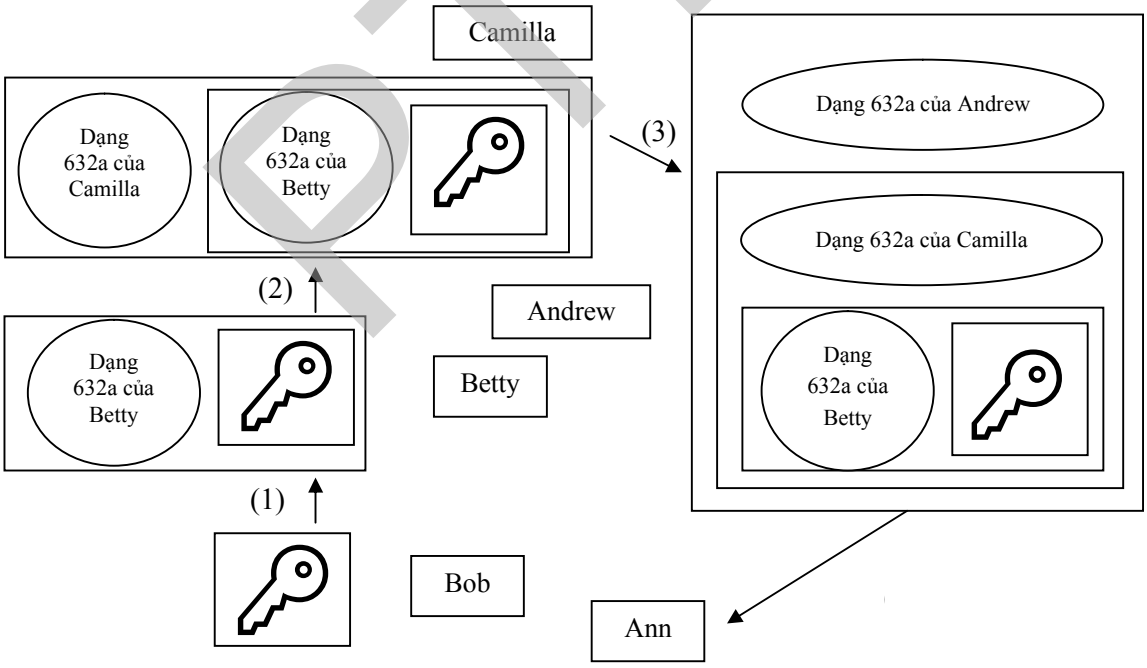
Xem xét cơ cấu tổ chức của một công ty có thể cho ta thấy sự tin cậy được xây dựng ra sao. Giả sử Ann và Bob gặp nhau. Bob nói rằng anh ta cũng làm trong công ty với Ann.

Trong công ty của họ Andrew và Betty là hai trưởng nhóm tác nghiệp trong cùng một dự án mà Camilla là giám đốc và Annlàm việc cho Andrew, Bob làm việc cho Betty. Các yếu tố này có thể giúp cho Ann và Bob tin cậy nhau. (Các quan hệ về mặt tổ chức được chỉ ra ở hình 5.5).



Hình 5.5. Tổ chức trong một công ty giả định

Ann sẽ hỏi Andrew: Bob là ai, Andrew hoặc hỏi Betty nếu anh ta biết Betty, nếu không anh ta sẽ hỏi Camilla, (Camilla hỏi Betty), Betty trả lời Bob làm việc cho mình. Câu trả lời này sẽ đi theo con đường ngược với các câu hỏi. Tuy nhiên nếu Bob nằm trong tác nghiệp khác thì có thể cần phải leo cao hơn theo cây tổ chức này cho tới khi tìm thấy điểm chung.



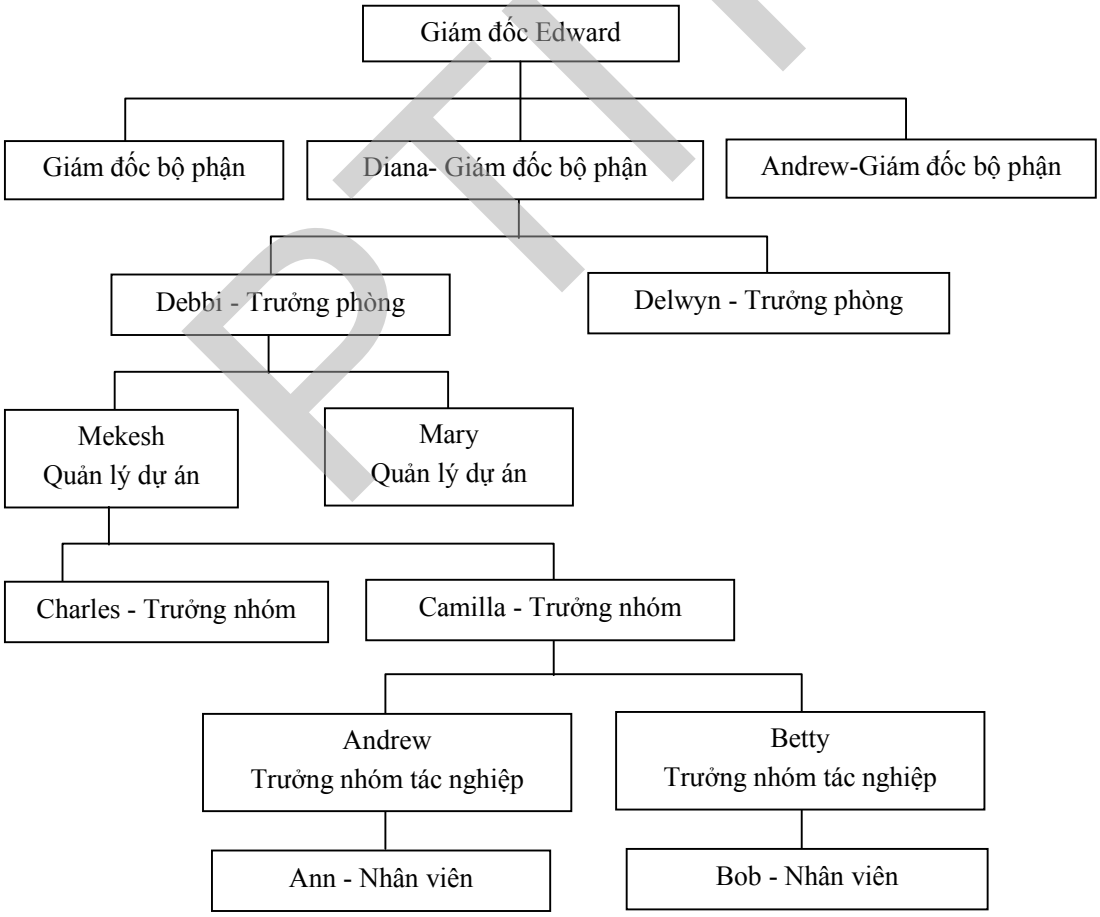
Hình 5.6. Bob chuyển khoá cho Ann

Ta có thể sử dụng một quá trình tương tự để trao đổi khoá mật mã. Nếu Bob và Ann muốn liên lạc, Bob có thể trao khoá công khai của mình cho Betty là người sẽ chuyển nó cho Camilla hoặc trao trực tiếp cho Andrew - là người trao nó cho Ann. Tuy nhiên cách thức trong thực tế không diễn ra đúng như vậy, khoá phải được gắn thêm một ghi chú nói rằng nó là từ

Bob ở dạng thông tin trong danh bạ điện thoại hoặc ở dạng 947 của Hồ sơ cá nhân (tương tự như mẫu 2a/TCTW Hồ sơ công chức Việt Nam). Và nếu dạng 947 được sử dụng thì Betty phải gắn thêm thông tin dạng 632a. Camilla cũng sẽ gắn thêm một thông tin dạng 632a khác và cuối cùng là Andrew cũng làm như vậy (Xem hình 5.6).

Chuỗi các thông tin dạng 632a này muốn nói rằng "Tôi là Betty và tôi đã nhận khoá này và có gắn kèm hồ sơ cá nhân của một người mà tôi biết là Bob". "Tôi là Camilla và tôi đã nhận khoá này có gắn kèm hồ sơ cá nhân và trích lục dạng 632a từ một người mà tôi biết là Betty" và cứ như vậy. Khi Ann nhận được khoá cô ta có thể xem lại chuỗi bằng chứng này và kết luận với một đảm bảo hợp lý là khoá này thực sự đã tới từ Bob. Thủ tục này là một cách để thu được một khoá công khai có chứng thực (kết hợp giữa khoá và một tính danh tin cậy). Mô hình này làm việc tốt trong một công ty vì luôn có một người nào đó là chung đối với hai nhân viên bất kỳ, thậm chí nếu hai nhân viên nằm ở các bộ phận khác nhau tới mức mà người chung là Giám đốc công ty.

Tuy nhiên quá trình này sẽ gặp trở ngại do mỗi khi Ann và Bob muốn liên lạc thì cả Ann, Andrew, Camilla, Betty và Bob đều phải tham gia. Nếu Betty đi công tác hoặc Andrew bị ốm thì thủ tục sẽ chuyệch choạc. Thủ tục này cùng không làm việc tốt nếu giám đốc không thể làm được bất cứ một việc quan trọng nào do phải mất thời gian vào việc tạo các biểu 632a.



Hình 5.7. Cấu trúc công ty mở rộng

Để khắc phục vấn đề thứ nhất Bob có thể yêu cầu được cung cấp đầy đủ các mẫu biểu 632a từ giám đốc xuống anh ta. Sau đó Bó có thể trao bản sao của chúng tới bất kỳ ai trong

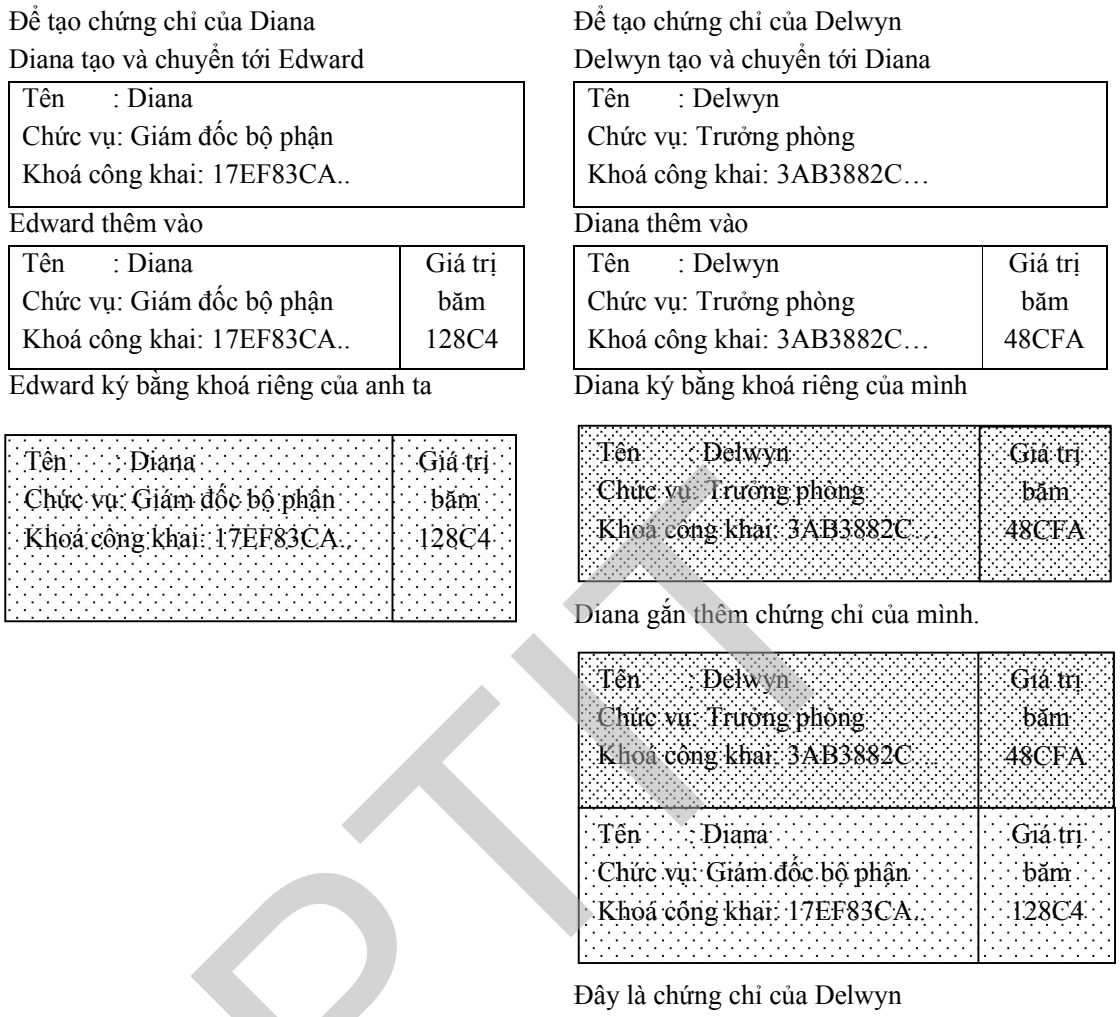
công ty cần tới khoá của mình. Thay cho phải làm từ đáy tới điểm chung, Bob sẽ bắt đầu từ đỉnh và nhận chuỗi đầy đủ các mẫu biểu của mình. Anh ta sẽ nhận các chữ ký cần thiết vào bất cứ lúc nào mà các cấp trên của anh ta rỗi việc, như vậy họ không nhất thiết phải có mặt khi anh ta muốn trao khoá công khai đã được chứng thục của mình. Thay cho việc bắt đầu từ đáy (từ các nhân viêc tác nghiệp) và làm việc dần tới đỉnh của cây (Giám đốc) ta sẽ bắt đầu từ đỉnh. Như vậy, Bob sẽ có một khoá đã được chứng thực trước để có thể sử dụng được không hạn chế trong tương lai. Ta sẽ mở rộng cấu trúc của công ty giả định từ giám đốc tới các mức (Xem hình 5.7).

Giám đốc công ty sẽ viết một bức thư cho mỗi giám đốc bộ phận nói rằng."Tôi là Edward - chủ tịch công ty. Tôi chứng nhận cho Diana là giám đốc bộ phận là người mà tôi biết rõ và tôi trao cho Diana quyền chứng nhận cho các thuộc cấp của mình". Tương tự mỗi giám đốc sẽ sao lại thư này và kèm theo thư của mình… Bob sẽ nhận được một tập thư từ chủ tịch xuống tới trưởng nhóm tác nghiệp của mình, mỗi thư được kết nối theo tên tới thư tiếp. Nếu mỗi nhân viên trong công ty đều có một tập thư như vậy thì hai nhân viên bất kỳ muốn trao đổi các khoá đã chứng thực thì họ chỉ cần so sánh các tập thư này của nhau: cả hai tập thư sẽ có ít nhất một người chung là Edward. Chẳng hạn Bob và Ann sau khi so sánh thấy rằng các thư này là như nhau sau Camilla còn phần còn lại ở bên dưới thì Bob biết rằng phần của Ann đã được Camilla chứng thực vì nó giống như phần của anh ta và Ann cũng biết như vậy. Mỗi người đều biết phần còn lại là đúng vì nó được gắn với nhau bằng một dãy liên tục các tên và chữ ký.

Thủ tục này về mặt điện tử có thể biểu thị dễ hơn trên giấy. Với giấy tờ thì phải đảm bảo chống giả mạo nhằm tránh để một thư nào đó trong tập thư bị thay thế và để đảm bảo rằng khoá công khai ở đáy được gắn kết với dãy này. Về mặt điện tử thì toàn bộ các công việc này được làm thông qua các chữ ký số và các hàm băm (hash). Có thể coi Kohnfelder là người đầu tiên đưa ra ý tưởng này và sau đó Merkle đã mở rộng nó trong bài báo của mình và gần đây hơn là trong chuẩn quốc tế X.509.

Khoá công khai và tính danh của người dùng được gắn với nhau thành một chứng chỉ và rồi nó lại được ký bởi một người nào đó để chứng thực tính đúng đắn của mối liên kết này. Trước tiên Edward chọn một cặp khoá công khai và đặt phần công khai ở nơi mà mọi người trong công ty có thể nhận được nó và giữ lại phần bí mật của mình. Sau đó một giám đốc bộ phận (chẳng hạn Diana) sẽ tạo cặp khoá công khai của mình rồi đặt khoá công khai trong một thông báo cùng với tính danh và chuyển nó một cách an toàn tới Edward. Edward sẽ ký nó bằng cách tạo ra một giá trị băm của thông báo (tóm lược thông báo) và rồi mã hoá thông báo và hàm băm này bằng khoá riêng của minh. Bằng cách ký thông báo như vậy Edward sẽ khẳng định rằng khoá công khai (của Diana) và tính danh (cũng của Diana) trong thông báo là của cùng một người. Thông báo này được gọi là chứng chỉ của Diana. Các trưởng phòng của Diana đều tạo các thông báo bằng khoá công khai của họ. Diana sẽ ký và tóm lược (băm) chúng rồi chuyển lại đồng thời gắn thêm vào một bản sao chứng chỉ mà mình đã nhận đực từ Edward. Bằng cách này mọi người có thể kiểm tra chứng chỉ của người phụ trách bằng cách xuất phát từ khoá công khai đã biết của Edward để giải mã chứng chỉ của Diana nhằm thu được khoá công khai (và tính danh) của Diana và dùng khoá này để giải mã cho chứng chỉ của người phụ trách. Hình 5.8 chỉ ra cách tạo các chứng chỉ cho Diana và Delwyn. Quá trình này cứ tiếp tục như vậy cho tới Ann và Bob. Hình H.5.9 ta thấy rằng chứng chỉ của Bob là chứng

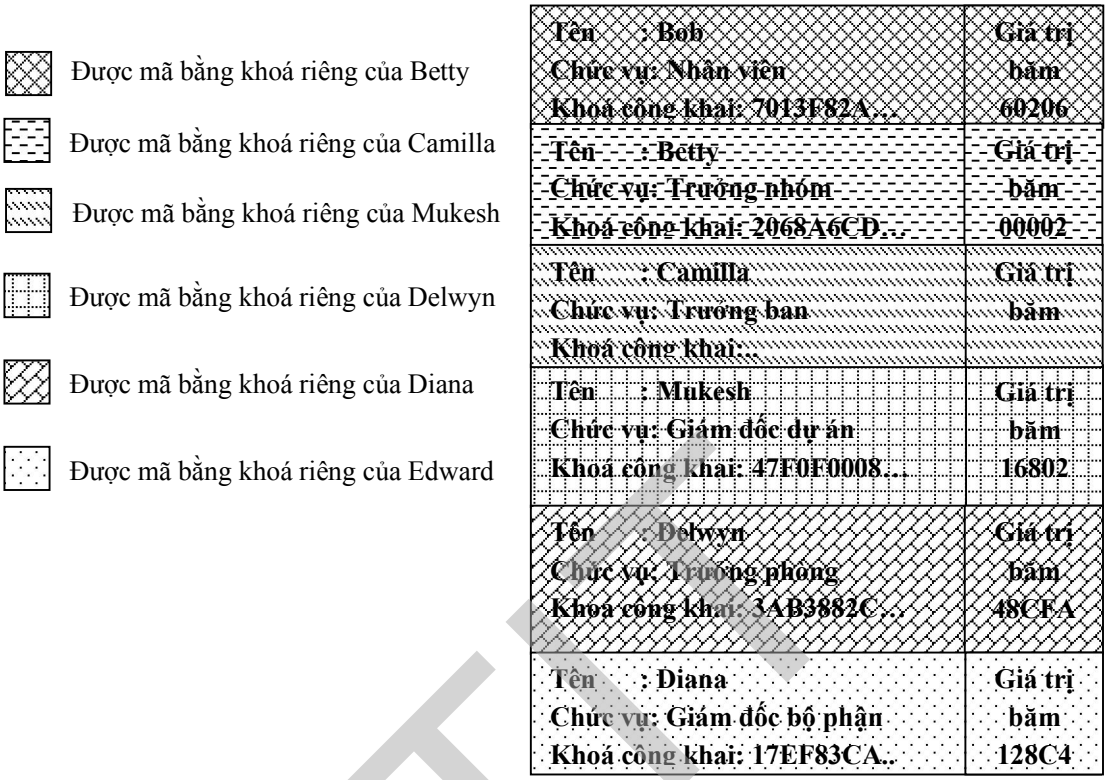
chỉ riêng của anh ta kèm theo tất cả các chứng chỉ các xếp từ người phụ trách trực tiếp cho tới ông giám đốc công ty.



Hình 5.8. Các chứng chỉ có ký

Trong ví dụ này, các chứng chỉ được đưa ra theo tuyến quản lý. Tuy nhiên không nhất thiết phải có hoặc tuân theo cấu trúc quản lý để sử dụng việc ký chứng thực. Bất cứ một ai được coi là có thẩm quyền đều có thể ký một chứng chỉ. Ví dụ, nếu bạn muốn xác định xem bậc một người nào đó coi đúng là có bằng đại học hay không thì bạn không cần gặp giám đốc hoặc viện trưởng, thay vào đó bạn chỉ cần đi tới cơ quan lưu trữ. Để kiểm tra công việc của một người nào đó bạn phải gửi tới cơ quan nhân nhân sự hoặc người phụ trách nhân sự. Và để kiểm tra xem liệu một người nào đó có sống ở một địa chỉ xác định thì bạn có thể hỏi cơ quan lưu trữ hành chính công cộng. Một công chứng viên sẽ chứng nhận tính hợp pháp của một chữ ký trên một tài liệu một số công ty có cơ quan an ninh, một số công ty có một cơ quan nhân sự tách biệt đối với mỗi bộ phận hoặc mỗi một nhà máy. Các cán bộ phụ trách của bộ phận này có thể ký xác nhận cho những người trong phạm vi quản lý của họ. Các phân cấp tự nhiên tồn tại trong xã hội và các phân cấp tương tự cũng có thể được sử dụng để tạo hiệu lực cho các chứng chỉ. Vấn đề duy nhất trong cấu trúc phân cấp là cần phải trông cậy vào mức đỉnh. Toàn bộ chuỗi xác thực là an toàn vì mỗi chứng chỉ sẽ chứa khóa phải mã cho chứng chỉ tiếp sau ngoại trừ mức đỉnh. Trong một công ty việc đặt niềm tin vào giám đốc là một điều

hợp lý. Tuy nhiên nếu các chứng chỉ trở nên được sử dụng rộng rãi trong thương mại điện tử thì mọi người phải có khả năng trao đổi các chứng chỉ một cách an toàn qua các công ty, các tổ chức và các quốc gia.



Hình 5.9. Chuỗi các chứng chỉ

Internet là một tập hợp rộng lớn của các mạng dùng trong liên lạc bên ngoài các công ty, các tổ chức và quốc tế (cũng như bên trong các công ty, các tổ chức và một đất nước). Internet được chạy (nói một cách chính xác hơn là được phối hợp) bởi một nhóm nhỏ được gọi là Cộng đồng Internet. Cộng đồng Internet đã đề nghị rằng nó là người ký cho các chứng chỉ với khoá công khai cho liên lach Internet. Cộng đồng này sẽ ký các chứng chỉ cho Nhà chức trách chứng thực của cảnh sát và cơ quan này lại chứng thực cho các chứng chỉ cho các cơ quan nhà nước, các doanh nghiệp, các nhà trường, các tổ chức khác và cho các cá nhân. Quan điểm này có lẽ là một quan điểm đúng đắn nhất.

Bây giờ ta sẽ lướt qua một số quan điểm về phân phối khoá từ phân phối qua trao đổi trực tiếp tới phân phối qua trung tâm phối có chứng thực. Mỗi loại đều có ưu nhược điểm riêng. Những điểm cần nhớ trong các thủ tục này (cũng như những điểm khác mà ta sẽ nghiên cứu) là:

- Có những hạn chế hoạt động nào? Chẳng hạn thủ tục có cần một phương tiện khả dụng liên tục ví như trung tâm phân phối khoá hay không?
- Có những yêu cầu tin cậy nào? Ai và những thực thể nào phải được tin cậy để hoạt động đúng?

- Có những gì để bảo vệ tránh khỏi những trục trặc? Liệu một kẻ bên ngoài có thể giả mạo một thực thể nào đó trong thủ tục và gây tổn hại tới tính an toàn? Liệu có một kẻ đồng loã nào đó trong thủ tục thực hiện lừa đảo mà không bị phát hiện?
- Hiệu quả của thủ tục ra sao? Một thủ tục yêu cầu một số bước để thiết lập một khoá mã hoá sẽ được sử dụng nhiều lần là một vấn đề, một vấn đề khác là phải qua một số bước mất rất nhiều thời gian để có một khoá sử dụng một lần.
- Sử dụng một thủ tục có dễ dàng không? Cần chú ý rằng độ phức tạp trong các ứng dụng máy tính có thể khác với độ phức tạp trong các ứng dụng thủ công. Ta sẽ đưa ra một số mô tả cho một vài thủ tục, một số thủ tục sẽ được đưa ra trong các ví dụ cuối chương. Bạn cần phải ghi nhớ những điểm này khi nghiên cứu những thủ tục còn lại trong chương này.

5.2.2. Các chữ ký số

Ta sẽ nghiên cứu một ứng dụng điển hình trong máy tính thể hiện một nhu cầu thông thường của con người. Lệnh chuyển tiền từ một người này tới một người khác. Về căn bản đây là một dạng séc đã được máy tính hoá. Theo truyền thống ta đã hiểu giao dịch này sẽ được thực hiện như thế nào ở dạng giấy tờ.

- Séc là một đối tượng xác định có tư cách giao dịch thương mại.
- Chữ ký trên séc sẽ xác nhận tính xác thực bởi vì chắc chắn chỉ có người ký hợp pháp mới có thể tạo được chữ ký này.
- Trong trường hợp có sự giả mạo bất hợp pháp thì một bên thứ ba có thể được gọi và để phán xét tính xác thực.
- Séc bị huỷ để nó không thể sử dụng lại.
- Séc giấy không thể thay đổi được hay hầu hết các kiểu thay đổi đều có thể dễ dàng phát hiện được.

Việc giao dịch thương mại bằng séc phụ thuộc vào các đối tượng xác định ở dạng được mô tả trước.

Đối với các giao dịch trên máy tính không tồn tại các đối tượng xác định (séc). Bởi vậy việc chấp nhận chi trả bằng máy tính đòi hỏi một mô hình khác. Ta sẽ xem xét các yêu cầu trong một tình thế tương tự từ phía nhà băng và phía người sử dụng.

Sandy gửi cho ngân hàng của mình một thông báo cho phép ngân hàng chuyển 100 USD cho Tim. Ngân hàng của Sandy phải có khả năng kiểm tra và chứng tỏ được rằng thông báo thực sự đến từ Sandy nếu sau đó cô ta không nhận là mình đã gửi nó. Ngân hàng cũng muốn biết rằng toàn bộ thông báo này là của Sandy và nó không thể sửa đổi. Về phần mình, Sandy cũng muốn chắc chắn rằng ngân hàng của cô ta không thể giả mạo những thông báo đó là thông báo mới, không phải là một thông báo trước đó đã được sử dụng lại và nó phải không bị sửa đổi trong khi truyền. Việc sử dụng các tín hiệu điện tử thay cho giấy sẽ làm phức tạp thêm giao dịch này.

Chữ ký số là một thủ tục tạo ra một hiệu quả tương tự như chữ ký thực. Nó là một dấu hiệu mà chỉ có người gửi mới có thể tạo ra nhưng những người khác có thể dễ nhận thấy được rằng nó là của người gửi. Giống như chữ ký thực, chữ ký số được dùng để xác nhận nội dung của một thông báo.

Các chữ ký phải thoả mãn các điều kiện cơ bản sau:

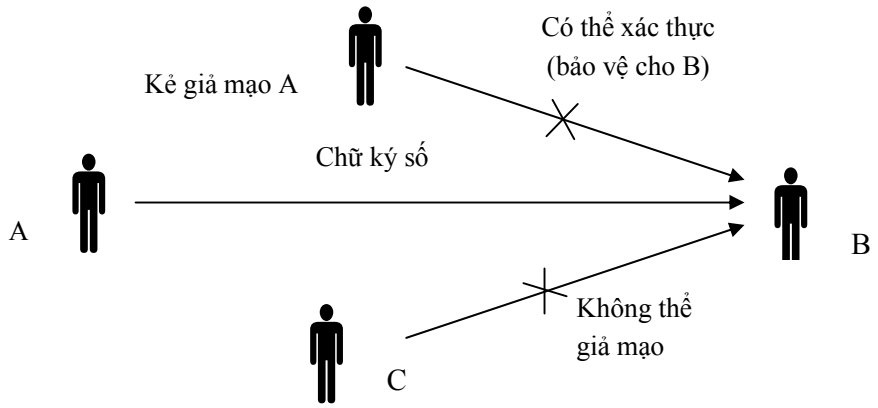
- Không thể giả mạo. Nếu P ký thông báo M bằng chữ ký S(P, M) thì không một ai có thể tạo được cặp $[M, S(M, P)]$
- Xác thực. Nếu R nhận được cặp $[M, S(M, P)]$ được coi là của P thì R có thể kiểm tra được rằng chữ ký có thực sự là của P hay không? Chỉ có P mới có thể tạo được chữ ký này và chữ ký được "gắn chặt" với M. Hai yêu cầu đầu tiên này là những trở ngại chính trong giao dịch qua máy tính. Hai tính chất bổ trợ sau là những tính chất mong muốn đối với giao dịch được hoàn tất nhờ chữ ký số:
- Không thể thay đổi. Sau khi được phát M không thể thay đổi bởi S, R hoặc bởi một kẻ thu trộm nào.
- Không thể sử dụng lại. Một thông báo trước đó được đưa ra sẽ ngay lập tức bị R phát hiện.

Trước tiên ta sẽ trình bày một cơ chế thoả mãn hai yêu cầu đầu tiên, sau đó ta sẽ thêm vào đó giải pháp để thoả mãn được các yêu cầu khác.

5.2.2.1. Các chữ ký số khoá đối xứng

Với hệ thống mã hoá riêng việc giữ bí mật khoá sẽ đảm bảo tính xác thực của thông báo cũng như độ mật của nó. Nếu Sandy và ngân hàng có một khoá mã hoá chung thì Sandy có thể mã hoá yêu cầu chuyển tiền của mình. Ngân hàng có thể tin vào tính xác thực của thông báo vì không một ai ngoài Sandy có khoá này.

Tuy nhiên hệ thống mã với khoá đối xứng truyền thống không tránh khỏi được sự gải mạo: Ngân hàng có thể tạo ra thông báo như vậy vì ngân hàng cũng có khoá này. Bởi vậy ngân hàng có thể tin chắc vào tính xác thực của thông báo, nhưng ngân hàng không có sự bảo vệ chống lại sự chối bỏ (không thừa nhận việc gửi thông báo). Vì cả Sandy và ngân hàng đều có cùng một khoá nên cả Sandy và ngân hàng đều có thể tạo một thông báo bất kỳ. Khi Sandy nhận một thông báo từ ngân hàng, cô ta tin vào tính xác thực của nó nhưng cô ta có thể tin vào bất cứ một ai khác. Xác thực nhưng không thể là không chối bỏ là một đặc điểm của mã hoá đối xứng. Với các hệ mật đối xứng như chuẩn mã dữ liệu (DES) cần phải có một trọng tài để tránh khỏi việc giả mạo.

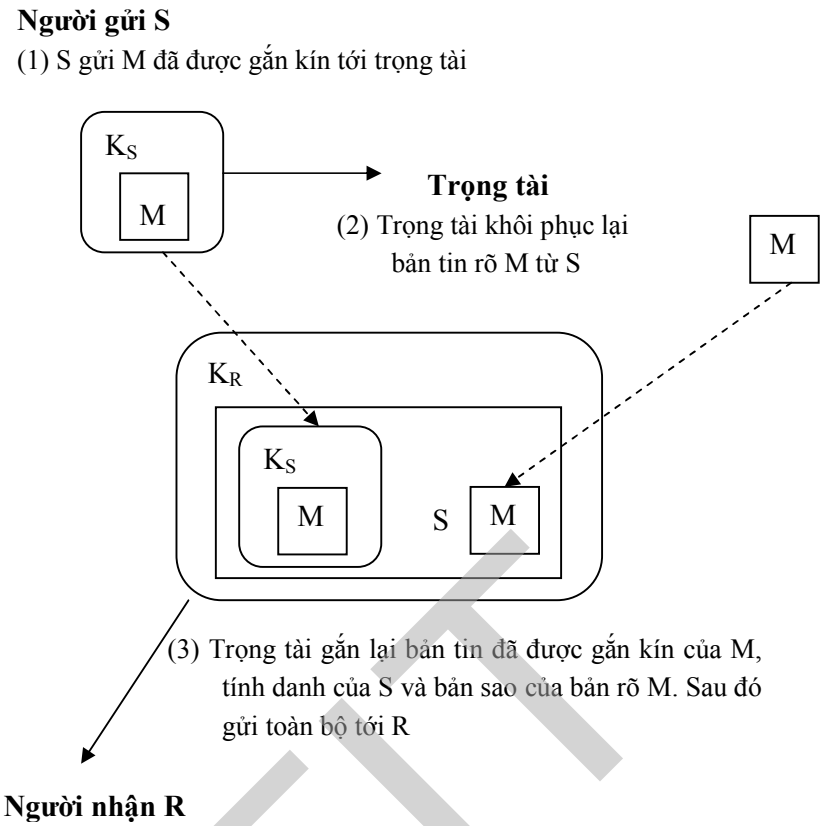


Hình 5.10. Những yêu cầu đối với chữ ký số

Sau đây là một phác thảo của thủ tục chữ ký số. Giả sử S là người gửi (Sandy), A là trọng tài và R là người nhận (Ngân hàng). S có khoá K_S chung với A và R có khoá K_R chung với A. Giả sử S và R đã thoả thuận trước về khuôn dạng cho một chữ ký số. S muốn gửi một thông báo M cho R với các yêu cầu phụ trợ là thông báo này không thể giả mạo về tính xác thực của nó có thể kiểm tra được.

Như trên hình 5.11, trước tiên S sẽ gửi $E(M,K_S)$ tới trọng tài A. A sẽ gửi ra M bằng cách dùng K_S. Sau khi kiểm tra thấy rằng thông báo này thực sự là từ S (vì trọng tài có thể giải nó bằng khoá K_S của Sandy), A sẽ gửi $E\left(\left(M,S,E(M,K_S)\right),K_S\right)$ tới R. R sẽ thu toàn bộ thông báo M đã được mã bằng khoá K_R để R có thể giải mã và xử sự theo nội dung của thông báo. R cũng thu thông báo S của A, thông báo này nói rằng A chứng thực là thông báo tới từ S. R cũng thu được $E(M,K_S)$, đây là thông báo mà R không thể giải mã vì nó được mã bằng khoá K_S. Tuy nhiên R sẽ giữ lại một bản sao của M và $E(M,K_S)$ phòng trường hợp có sự tranh chấp sau này. Ở đây điều kiện xác thực được thoả mãn vì R tin tưởng vào trọng tài (là người nói rằng thông báo tới từ S). Tính chất không thể giả mạo cũng được thoả mãn vì nếu sau đó S tuyên bố rằng đó là bản giả mạo thì S sẽ đưa ra M và $E(M,K_S)$. Trọng tài có thể mã hoá lại M bằng K_S và xác nhận rằng chỉ có S (hoặc trọng tài - người mà ta coi là trung thực) mới có thể tạo được $E(M,K_S)$ vì nó được mã bằng K_S. Bởi vậy trọng tài có thể chứng thực rằng S (hoặc một ai đó có khoá K_S) đã gửi M.

Thủ tục này sẽ tạo ra một hệ thống thực sự mạnh hơn yêu cầu, thông báo đã được phát ở dạng đã mã hoá ngay cả khi không cần phải bí mật trong mọi tình huống. Như ta đã thấy, một số thuật toán mã hoá tiêu tốn khá nhiều thời gian và sử dụng nó sẽ làm giảm tốc độ truyền thông báo. Bởi vậy ta muốn có một thủ tục khác không cần phải mã hoá toàn bộ thông báo.



Hình 5.11. Chữ ký số khoá đối xứng có trọng tài

5.2.2.2. Các chữ ký số không có mã hoá

Nếu S và R không quan tâm tới độ mật thì họ có thể thoả thuận dùng hàm niêm phong mật mã như một chữ ký. Niêm phong là một con tem, con dấu hoặc một nhãn gắn cố định vào tư liệu để chứng tỏ tính xác thực của nó. Hàm niêm phong là một hàm toán học chịu tác động bởi mọi bit đầu vào của nó. Ví dụ, các bytes của một thông báo có thể được dùng như các số và tổng của tất cả các bytes của một thông báo có thể tính toán được. Tổng này là duy nhất đối với một thông báo vì bất kỳ một sự thay đổi nào của thông báo cũng đều gây nên sự thay đổi của tổng này. Hàm niêm phong có thể là một hàm băm hoặc một hàm mã hoá một chiều, hoặc nó cũng có thể là một hàm khác phụ thuộc vào toàn bộ đầu vào và dễ tính toán.

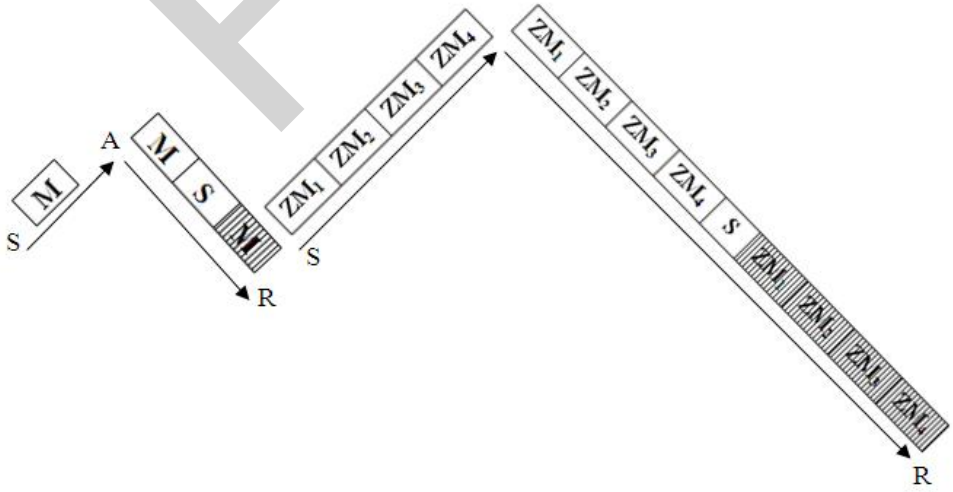
Giả sử S và R đều đăng ký một hàm niêm phong của mình với trọng tài: giả sử f_S và f_R là hai hàm này. Sau đó, S gửi M và $f_S(M)$ tới A. A cũng tính $f_S(M)$ theo bản sao của M đã nhận được từ S. Nếu hai giá trị này của $f_S(M)$ trùng nhau thì thông báo từ S được coi là xác thực. Sau đó, A gửi $M, S, f_S(M)$ và $f_R(M, S)$ cho R. R không thể hiểu được $f_S(M)$, nhưng R sẽ giữ lại nó làm bằng chứng báo rằng S đã gửi thông báo M. S sẽ kiểm tra tính đúng đắn của $f_R(M, S)$ để biết tính xác thực của thông báo.

5.2.2.3. Ngăn chặn sử dụng lại hoặc biến đổi

Cả hai giải pháp trên đều thoả mãn các đòi hỏi về tính xác thực và chống giả mạo. Bởi vậy, chúng đều là các thủ tục chữ ký số chấp nhận được. Tuy nhiên, ta còn muốn tránh được việc sử dụng lại hoặc biến đổi một thông báo cũ. Mặc dù R không thể tạo các thông báo mới bằng K_S hoặc f_S nhưng bên thu có thể sử dụng lại các thông báo cũ. Ví dụ, bên thu có thể lưu lệnh cũ trả cho Tim 100 USD hàng tháng. Hơn nữa, biết kỹ thuật mã hoá nhưng không biết khoá bên thu vẫn có thể cắt các mẩu từ các thông báo cũ và dán chúng với nhau để tạo nên một thông báo mới. Việc kiểm tra giấy tờ sẽ giải quyết vấn đề này như sau: ngân hàng sẽ đóng dấu huỷ séc và trả nó lại cho người gửi để người gửi biết không thể sử dụng lại séc này. Ta cần có một cách để tạo một chữ ký máy tính tự huỷ tương tự để không thể sử dụng lại được.

Để minh chứng việc sử dụng một thông báo, ta phải tạo một phần của chữ ký vào một nhãn thời gian. Ví dụ, nếu thông báo của Sandy có ghi ngày tháng và thời gian gửi thì ngân hàng không thể tạo lại thông báo tương tự vào tuần sau mà không bị Sandy và những người khác phát hiện ra sự giả mạo. Nhãn thời gian không phải là thời gian và ngày tháng theo nghĩa đen mà là một mã bất kỳ không lặp lại, chẳng hạn như tăng số loạt. Điều này sẽ gây khó khăn cho việc tạo lại.

Để tránh việc cắt thông báo thành nhiều mẩu và sử dụng lại một mẩu nào đó, Sandy có thể làm cho mỗi mẩu đều phụ thuộc nhãn thời gian. Ví dụ, với DES (thuật toán này sẽ mã các khối văn bản 64 bit thành một đầu ra 64 bit). Sandy có thể mã thời gian và ngày tháng ở 8 bit đầu của mỗi khối và chỉ để 56 bit làm nội dung của thông báo. Vì tất cả 64 bit của mỗi khối ra đều phụ thuộc vào toàn bộ 64 bit vào nên ngân hàng không thể gắn 56 bit thông báo với một nhãn thời gian 8 bit khác. Quá trình này được chỉ ra trên hình 5.12. Như sẽ thấy ở cuối chương, chế độ liên kết khối mã của DES cũng là một phương pháp khác nhằm tránh việc sử dụng lại một khối của một thông báo vào một thông báo khác.



Hình 5.12. Chữ ký số khoá đối xứng

Những giải pháp trên là khá phức tạp. Chúng yêu cầu phải luôn có một trọng tài trong mỗi giao dịch và để đảm bảo bí mật thì thông báo phải được mã hai lần. Rất may là thủ tục khoá công khai sẽ cho ta tạo một phương pháp đơn giản hơn.

5.2.2.4. Thủ tục khoá công khai

Các hệ thống khoá công khai là các hệ thống thích hợp nhất cho chữ ký số. Để đơn giản về ký hiệu, ta giả sử rằng $E(M, K_U)$ là mã hoá khoá công khai cho người dùng U và phép biến đổi khoá riêng cho U được ký hiệu là $D(M, K_U)$. Ta cũng có thể coi E là phép biến đổi riêng (vì chỉ có U mới có thể giải mã nó) và coi D là một phép biến đổi xác thực (vì chỉ có U mới có thể tạo ra nó). Tuy nhiên, cần nhớ rằng với một số thuật toán không đối xứng (như RSA), D và A là giao hoán. Bởi vậy:

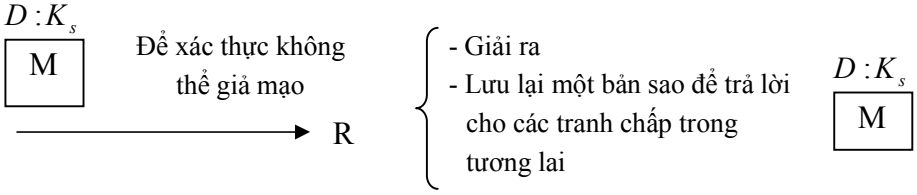
$$D(E(M, -), -) = M = E(D(M, -), -)$$

Nếu S muốn gửi M cho R thì S sẽ dùng phép biến đổi xác thực để tạo $D(M, K_S)$. Sau đó S gửi $D(M, K_S)$ cho R. R sẽ giải mã thông báo bằng phép biến đổi khoá công khai của S (S tính $E(D(M, K_S), K_S) = M$, xem hình 5.13). Vì chỉ có S mới có thể tạo ra một thông báo có nghĩa dưới dạng $E(-, K_S)$ nên thông báo này phải thực sự là từ S. Phép kiểm tra này thoả mãn yêu cầu về tính xác thực.

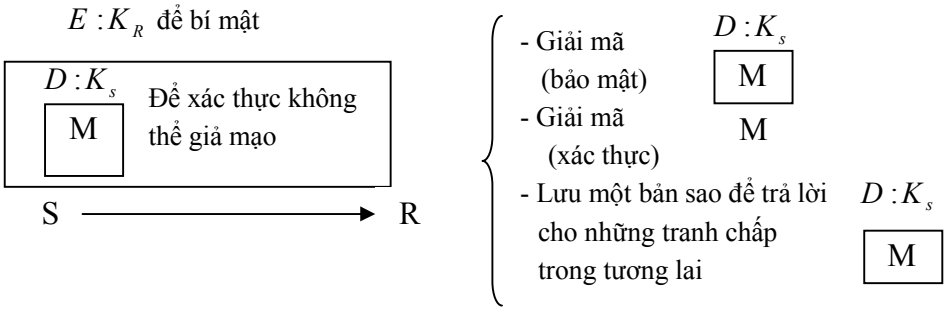
R sẽ lưu giữ $D(M, K_S)$. Nếu sau đó S khẳng định rằng thông báo này là giả mạo (không phải từ S) thì R chỉ cần đưa ra M và $E(M, K_S)$. Bất kỳ ai cũng có thể kiểm tra được rằng vì $D(M, K_S)$ được chuyển thành M nhờ dùng phép biến đổi khoá công khai của S nhưng chỉ có S mới có thể tạo ra $D(M, K_S)$ nên $D(M, K_S)$ phải là từ S. Phép kiểm tra này thoả mãn yêu cầu không thể giả mạo.

Giải pháp khoá công khai hiển nhiên là ít phức tạp hơn nhiều so với giải pháp khoá riêng. Một nhược điểm của giải pháp này là: thông báo là xác thực nhưng không đảm bảo tính bí mật. Tức là, bất cứ ai biết khoá công khai của S đều có thể dịch được thông báo này. Ta có thể khắc phục nhược điểm này bằng cách mã hoá 2 lần.

Vì S cần phải gửi M cho R nên tính xác thực có thể được đảm bảo bằng cách dùng khoá riêng của S, còn tính bí mật có thể được đảm bảo bằng khoá công khai của R. Ví dụ, S có thể gửi $E(D(M, K_S), K_R)$ tới R (như được chỉ ra trên hình 5.13). Vì chỉ có S mới có thể tạo ra $D(-, K_S)$ nên thông báo này phải từ S. Nhưng vì chỉ có R mới có thể giải mã cho $E(-, K_R)$ nên nội dung của thông báo vẫn được giữ bí mật cho tới khi R giải nó. Như vậy, với hai phép mã hoá, ta sẽ có một thủ tục đảm bảo cả tính riêng tư và tính xác thực.



Hình 5.13. Chữ ký số không đối xứng



Hình 5.14. Sử dụng hai phép mã hoá trong chữ ký số không đối xứng

Như đã nêu ở trên, các nhãn thời gian có thể được dùng để đảm bảo chống lại việc sử dụng lại. Một hàm niêm phong cũng có thể khắc phục được việc thay thế các mẩu trong bản mã.

5.2.3. Giao kèo về khoá

Giả sử bạn làm mất chìa khoá nhà.Với đôi chút bất tiện, bạn có thể gọi một thợ khoá để mở khoá cho bạn, và thậm chí có thể làm cho bạn một chìa khoá khác. Bạn cần phải thuyết phục người thợ khoá để bạn vào nhà. Tuy nhiên, nếu bạn có một giấy tờ xác nhận nào đó, hoặc những người láng giềng đảm bảo cho thẩm quyền của bạn thì bạn không cần phải thuyết phục người thợ khoá.

Nếu bạn mất khoá của két bạc thì vấn đề sẽ phức tạp hơn đôi chút và người thợ khoá phải mất nhiều thời gian hơn. Tuy nhiên, trong một thời gian tương đối ngắn, bạn vẫn có thể mở được. Sở dĩ như vậy vì các loại khoá nhà và khoá két bạc được thiết kế chỉ để chống lại các kiểu tấn công trong một khoảng thời gian ngắn: một hoặc tối đa hai ngày. An toàn về vật lý đóng một phần quan trọng trong việc đảm bảo an toàn cho các ngôi nhà và các công sở. Ta mong muốn có thể ngăn chặn một cuộc tấn công kéo dài trong nhiều giờ.

Bản mã không có tính an toàn về mặt vật lý. Trên thực tế, nó được dùng để liên lạc trên các kênh không an toàn, nơi mà các thám mã có thể thu được các bản mã và phân tích nó trong nhiều giờ, nhiều ngày, nhiều tuần và thậm chí là hàng năm. Bởi vậy, các thuật toán mã hoá phải đủ mạnh để chống chọi được với các kiểu tấn công. Do đó, không có một "người thợ khoá" nào có thể tìm được các khoá bị mất hoặc bị quên. Nếu bạn làm việc trong một công ty và bạn đã mã hoá các file của mình thì vẫn có người có nhu cầu chính đáng để truy nhập vào khi bạn rời khỏi cơ quan trong một chuyến công tác hay trong các ngày nghỉ, hoặc khi bạn ốm hay khi bạn không còn ở công ty nữa. Bạn cũng có thể quên mất khoá của bạn. Đây là một trong các ưu điểm của giao kèo khoá và tại sao nó lại thích hợp trong những tính huống như vậy: giao kèo khoá sẽ cung cấp một phương tiện để đảm bảo an toàn số liệu ở một mức độ thích hợp nhưng vẫn cho phép những người dùng hợp pháp khác truy nhập vào.

5.2.3.1. Giao kèo khoá Clipper

Quan điểm Clipper là một ví dụ về một thủ tục giao kèo khoá. Bản chất của thuật toán này là chia một khoá thành n mẩu rồi giao cho n đối tượng sao cho k người nào trong số này đều có thể giải mã nhưng không có $(k-1)$ người nào có thể giải mã được. Thủ tục này được gọi là thủ tục k trong n. Đối với thủ tục Clipper, $k = n = 2$. (Trường hợp tổng quát hơn khi

$n > 2$ đã được Shamir nghiên cứu, ông đã đưa ra một giải pháp dựa trên việc tìm các nghiệm của một đa thức bậc n. Tuy nhiên, Blakley đã độc lập giải quyết vấn đề trên theo một phương pháp khác; chương 9 trong [9] của Simon đã có một tổng quan khá hay về các sơ đồ khác.)

Những yêu cầu đối với một thủ tục giao kèo khoá bao gồm:

- Liên lạc phải xác định được nguồn mã hoá của nó.
- Tính danh chỉ cho biết đơn vị mã hoá mà không để lộ khoá mã hoá riêng.
- Có thể phục hồi được khoá nhờ thủ tục k trong n.

Thủ tục mã hoá Clipper thoã mãn được các yêu cầu trên. Với thủ tục Clipper, đơn vị mã hoá xuất phát sẽ tạo ngẫu nhiên một khoá phiên 80 bit. Nó sẽ mã hoá khoá 80 bit này bằng một khoá duy nhất của mỗi đơn vị vật lý cụ thể có gắn số loạt đối với mỗi đơn vị vật lý và mã hoá cả hai bằng một khoá được chia xẻ bởi tất cả các thiết bị tương tự; nhóm được mã hoá này được gọi là LEAF (nhóm truy nhập hợp lệ).

Nó cũng sử dụng thủ tục trao đổi khoá riêng của mình để chuyển khoá này tới nơi thu. Tất cả các đơn vị Clipper đã được lập trình trước từ nhà máy với một khoá mã hoá đối xứng được chia xẻ (gọi là khoá gia đình). Clipper sử dụng một thủ tục trao đổi khoá 3 giai đoạn để trao đổi khoá. Bài toán này có những đòi hỏi sau:

- Các khoá mã hoá chỉ được phát ở dạng đã mã hoá.
- Việc trao đổi giữa các đơn vị Clipper phải chống được việc sử dụng lại.
- Hai đơn vị Clipper trong trao đổi phải xác thực nhau.

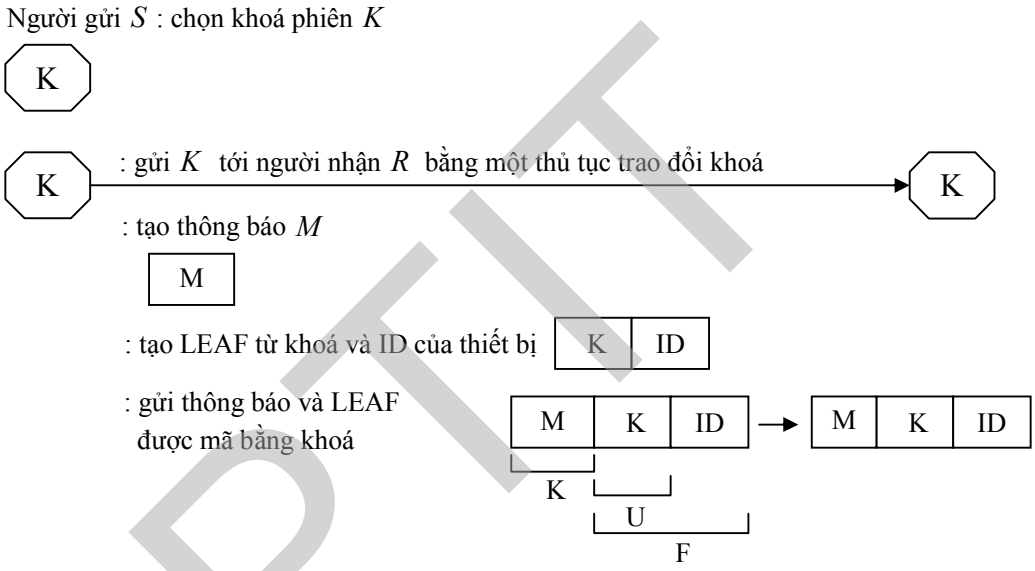
Thuật toán cần 3 bước để trao đổi khoá cộng thêm 4 bước để xác thực nhau. Thuật toán này do Bellovin và Merritt đưa ra [10]. Bên gửi S và bên nhận R chia sẻ một mật khẩu chung P, ngoài ra chúng có một thuật toán mã hoá không đối xứng E và một thuật toán mã hoá đối xứng chung Y.

1. S tạo một khoá công khai ngẫu nhiên E_S. S mã hoá E_S bằng Y_P và gửi $Y_P(E_S)$ cho R.
2. R biết P và dùng nó để giải mã và nhận được E_S. R sẽ tạo một khoá phiên ngẫu nhiên X. R chuyển X cho S bằng cách mã hoá kép bằng Y_P và E_S: R gửi $E_S(Y_P(X))$.
3. S dùng E_S và P để thu được X. Lúc này, việc trao đổi khoá đã hoàn tất. Bây giờ, S và R phải đảm bảo rằng họ đã có khoá mới X để sử dụng.
4. S tạo một xâu ngẫu nhiên M và gửi $E_X(M)$ cho R.
5. R giải mã thông báo này để thu được M. R chọn một xâu ngẫu nhiên N khác và chuyển trả lại cho S xâu ghép đã được mã hoá bằng khoá X: $E_X((M, N))$.

6. S giải mã thông báo này để thu được (M, N). Nếu phần đầu M chính là xâu mà S đã gửi thì R đã được xác thực đối với S. S gửi trả $E_X(N)$ cho R.

7. Nếu R thu được N đã gửi ở bước 4 thì S đã được xác thực đối với R. Sau khi Clipper đã hoàn thành việc trao đổi khoá, nó bắt đầu lưu chuyển thông báo với bên thu (mỗi thông báo có kèm theo LEAF). Như được mô tả trên H 5.15, một thông báo có kèm theo LEAF sẽ cho phép bên nhận hợp thức thu được khoá được dùng để mã hoá.

2. Bằng thủ tục này, các đối tác hợp thức thu chặn thông báo có thể dùng khoá gia đình F để giải mã đoạn cuối cùng của $E\left(\left(E(K, U)+ID\right), F\right)$ và thu được ID là số của thiết bị đã tạo ra khoá mã hoá thông báo. Họ có thể đệ trình lên một "toà án" để nhận được U mà từ nó, họ có thể thu được K từ $E(K, U)$.



Hình 5.15. Trao đổi thông báo Clipper

Tuy nhiên, thủ tục này có một số điểm không mong muốn. Trước hết là, một khi các đối tác hợp thức đã nhận được U từ "toà án" thì giá trị này sẽ cho phép họ truy nhập tới mọi thông báo đã được gửi bởi thiết bị này (trong quá khứ hoặc trong tương lai), thậm chí cả khi thiết bị này đã được mua hoặc bán một cách hợp pháp cho một người khác. Cũng vậy, một khi các đối tác hợp thức đã có U thì họ có thể tạo các thông báo mang danh người gửi, bởi vì E là một thuật toán không đối xứng. Tuy nhiên, những người sử dụng có thể tránh được tình huống này nếu dùng mã hoá bội: các đối tác hợp thức chỉ có thể giải mã $E(M, K)$ để tìm ra cái mà người gửi đã mã (là M) trước khi chuyển nó tới Clipper để mã. Tức là, M phải là $E'(M, K')$ đối với một phép mã hoá E' khác hoặc một khoá K' khác. Cần chú ý rằng, việc đặt phép mã hoá Skipjack chỉ ở trong phần cứng nhằm chống trả có hiệu quả việc thám mã bằng ứng dụng mật mã.

5.2.3.2. Giao kèo khoá bằng phần mềm

Các ứng dụng mật mã bằng phần mềm có cả ưu và nhược điểm so với ứng dụng phần cứng. Chúng dễ sửa đổi hoặc dễ làm phù hợp theo các yêu cầu của các ứng dụng riêng biệt. Chúng có thể được gắn vào hoặc được tích hợp vào các ứng dụng có thể được cài đặt hoặc chạy trên một máy tính bất kỳ. Các sai sót nếu có có thể dễ dàng được sửa và chi phí để sửa không đáng kể. Tuy nhiên, khả năng thay được cũng là một bất lợi lớn nhất: chúng có thể bị thay đổi bởi một sự cố hoặc bởi một chương trình ác ý nào đó nằm trên phương tiện lưu trữ chúng, trên bộ nhớ hoặc khi thực hiện chương trình. Ta cũng đã từng thấy rằng chỉ một thay đổi rất nhỏ, thậm chí chỉ là 1 bit cũng có thể gây nên hậu quả nghiêm trọng đối với sức mạnh của thuật toán mật mã. Tuy vậy, tính mềm dẻo của phần mềm vẫn làm cho nó có thể sử dụng được trong một số tình huống.

Việc mã hoá có giao kèo khoá dựa trên phần mềm cũng hoạt động tương tự như giao kèo khoá Clipper. Một cơ quan giao kèo sẽ lập trình trước một ứng dụng mật mã (phần mềm) với một khoá gia đình và một ID của thiết bị. Ứng dụng này sẽ được bán cho một nhà chế tạo phần mềm để gắn nó vào một sản phẩm phần mềm. Mỗi bản sao của phần mềm (tương tự như đối với một thiết bị phần cứng riêng) phải có ID riêng cuả nó. Sản phẩm sẽ nhận vào bản rõ và một khoá mã hoá, tạo ra bản mã và một LEAF. Phần mềm thu sẽ kiểm tra sự tồn tại của LEAF và giải mã. Người ta thấy rằng sử dụng mã hoá khoá công khai trong thủ tục giao kèo khoá là thích hợp hơn cả. Denning và Branstad [11] đã trình bày một số kết quả nghiên cứu lý thú về các thủ tục giao kèo khoá bằng phần mềm.

5.2.4. Chơi bài qua thư tín

Trong phần này, ta sẽ xem xét một tình huống tương tự khi cần đảm bảo cả bí mật lẫn xác thực. Ta xét một trò chơi bài mà những người tham gia không nhìn thấy nhau (chẳng hạn, chơi bài qua thư tín). Ngay cả khi việc chơi bài qua thư tín không phải là một công việc quan trọng thì thủ tục chơi vẫn có những ứng dụng quan trọng. Phần khó khăn nhất của việc chơi bài qua thư tín là việc đảm bảo rằng các quân bài phải được chia và phân phối một cách công bằng. Trong phần sau, ta sẽ sử dụng thủ tục này để phân phối các khoá mã hoá, trong trường hợp này, các quân bài chính là các khoá. Việc đặt quân bài sẽ dễ hơn cho việc giải thích thủ tục.

Giả sử Ann và Bill quyết định chơi bài qua thư tín. Ann là người chia và xóc các quân bài. (Để đơn giản, giả sử chỉ có 10 quân bài và mỗi bên chơi bài nhận 5 quân. Thủ tục này có thể dễ dàng mở rộng cho bộ bài truyền thống 52 quân).

5.2.4.1. Thủ tục phân phối

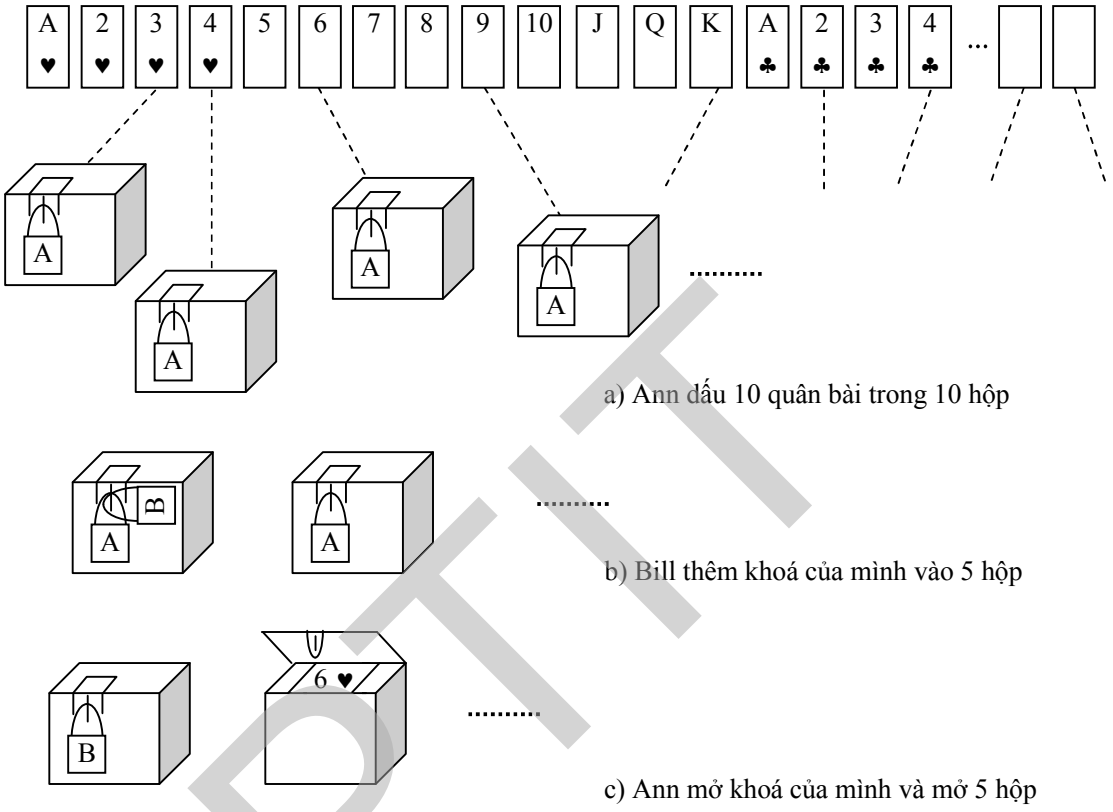
Để xáo bài, Ann sẽ đặt chúng theo một thứ tự tuỳ ý và đặt mỗi quân bài vào một hộp không đánh dấu sẵn rồi khoá từng hộp lại. Sau đó, Ann sẽ gửi 10 hộp cho Bill. Vì các hộp đều như nhau nên Bill sẽ chọn 5 hộp và đặt thêm một khoá thứ hai vào các hộp này. 5 hộp này là lựa chọn của Bill và Bill không động chạm đến 5 hộp còn lại (chỉ có khoá của Ann trên đó). Sau đó, Bill sẽ gửi trả lại Ann cả 10 hộp.

Theo các hộp đã nhận, Ann có thể xác định được Bill có xử sự đúng đắn không: anh ta đã chỉ chọn 5 hộp và không đụng chạm tới các quân bài (vì các khoá của Ann vẫn còn

nguyên). Ann sẽ mở cả 10 khoá của mình. Khi đó, Ann có 5 quân bài mà Bill không động tới (đây là những quân bài mà Ann đã chọn).

Ann sẽ gửi trả 5 hộp còn lại vẫn còn khoá của Bill. Bill sẽ mở khoá cho 5 hộp này. Ann không biết các quân bài này vì Ann không được mở các khoá của Bill. Bill (tin vào sự công minh của việc xáo bài) sẽ tiếp tục chơi.

Quá trình trên được mô tả trên hình 5.16.



Hình 5.16. Thủ tục chơi bài

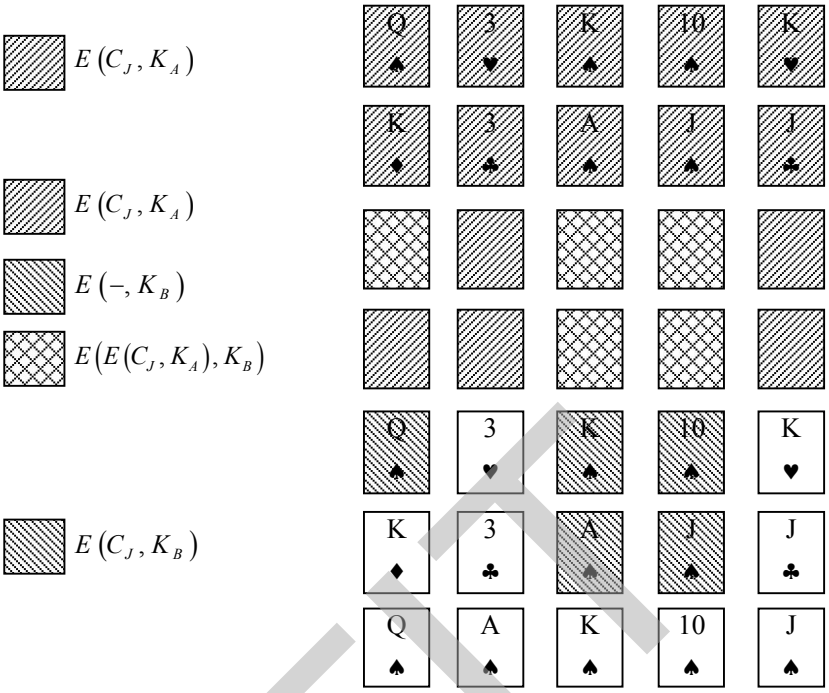
5.2.4.2. Áp dụng khoá đối xứng

Thủ tục này làm việc dễ dàng với các hệ thống mã hoá truyền thống. Để khái quát hoá, ta sử dụng thuật ngữ "thông báo" thay cho "quân bài" trong thủ tục. Thủ tục này được mô tả trên hình 5.17.

Ann sẽ "xáo" các thông báo và mã hoá chúng bằng khoá K_A (khoá của Ann). Sau đó, Ann sẽ gửi 10 thông báo đã mã này cho Bill. Khi nhận được chúng, Bill sẽ chọn ngẫu nhiên 5 cái. Bill không thể xác định được các thông báo này vì chúng đã được mã bằng khoá của Ann. Bill sẽ gửi trả lại cho Ann những thông báo không được chọn.

Bill sẽ nhận 5 thông báo còn lại (ta ký hiệu chúng là $B_1, \ldots B_5$). Bill sẽ gửi $E\left(B_1, K_B\right), \ldots, E\left(B_5, K_B\right)$ cho Ann. Chú ý rằng Ann đã gửi các thông báo đã mã cho Bill, các B_i, chính là $E\left(c_j, K_A\right)$ (thông báo c_j được mã bằng khoá của Ann). Bởi vậy, trên thực

tế Bill đã gửi $E\left(E\left(c_j, K_A\right), K_B\right)$ cho Ann. Phép mã hoá với khoá của Bill sẽ tạo nên một khoá kép trên các thông báo này để đảm bảo rằng Bill sẽ nhận trở lại những thông báo này và Ann không thể biết nội dung của chúng.



Hình 5.17. Thủ tục chơi bài với phép mã hoá

Bây giờ Ann đã nhận được 5 thông báo được mã một lần $E\left(c_j, K_A\right)$ và 5 thông báo được mã bằng hai khoá $E\left(E\left(c_j, K_A\right), K_B\right)$. Ann sẽ mở khoá 5 thông báo cần giữ bằng cách giải mã 5 thông báo $E\left(c_j, K_A\right)$. Cô ta cũng giải mã cho 5 thông báo mà Bill đã chọn bằng cách tạo ra $D\left(E\left(E\left(c_j, K_A\right), K_B\right), K_A\right)$. Nếu phép mã hoá và giải mã là giao hoán thì:

$$D\left(E\left(E\left(c_j, K_A\right), K_B\right), K_A\right) = E\left(D\left(E\left(c_j, K_A\right), K_A\right), K_B\right) = E\left(c_j, K_B\right)$$

Như vậy, các thông báo này bây giờ chỉ còn được "gắn" bởi khoá của Bill. Sau đó, Ann sẽ gửi lại những thông báo này cho Bill để giải chúng và tiếp tục trò chơi. Trên hình 5.17, các thông báo được xem là những quân bài. (Trên thực tế, thủ tục này có một yếu điểm nếu Ann chỉ cung cấp cho Bill 10 thông báo để lựa chọn. Hãy xem các bài tập ở cuối chương để thấy rõ).

5.2.4.3. Áp dụng khoá công khai

Ta sử dụng $D(\ )$ để ký hiệu cho phép biến đổi riêng (bí mật) và $E(\ )$ để ký hiệu cho phép biến đổi chung (công khai).Trước tiên, Ann sẽ "khoá" các thông báo bằng phép biến đổi

công khai; sau đó Bill sẽ khoá tiếp các thông báo được chọn bằng phép biến đổi công khai của mình. Ann sẽ mở khoá công khai của mình để thu được thông báo được chọn.

5.2.4.4. Phân phối khoá: Một ứng dụng của thủ tục

Thủ tục không bị hạn chế bởi tình huống khá phi thực tế là phân phối các quân bài bằng máy tính. Thay vào đó, ta xét một hệ thống mã hoá yêu cầu các khoá đặc biệt để không phải bất cứ số nào cũng là một khoá chấp nhận được. (Một số ứng dụng của DES đòi hỏi mỗi bit thứ 8 của khoá phải là bit kiểm tra chẵn của 7 bit đứng trước. Điều này dễ dàng được thực hiện với một khoá 56 bit bất kỳ. Tuy nhiên, điều này khác với điều một số nhị phân 56 bit bất kỳ có thể là khoá. Như đã mô tả trong chương 4, các thuật toán RSA, Merkle - Hellman và ElGamal đều yêu cầu một cặp khoá đặc biệt chỉ có thể do người sử dụng thuật toán xác định).

Nhìn chung, những người sử dụng không thể hoặc không muốn tự mình tạo khoá. Bởi vậy, việc cung cấp khoá cho người dùng cũng là một vấn đề. Một giải pháp là phải có một trung tâm để tạo và phân phối các khoá. Tuy nhiên, lại có những lo lắng xác đáng về việc giữ bí mật cho các kho giữ và gán các khoá.

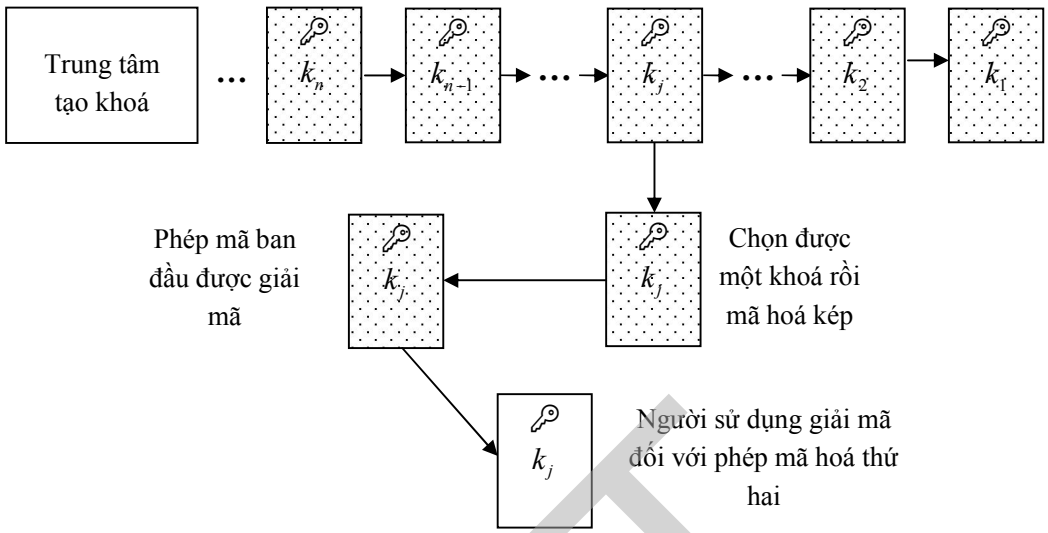
Cần phải có một thủ tục cho phép người dùng nhận được một khoá mới trong khi không một ai (kể cả kho lưu trữ) biết được khoá mà người dùng đã nhận là khoá nào. Ở đây có hai vấn đề:

- Không một ai có thể xác định được khoá riêng của người dùng cho ra khoá công khai của mình. Ta không thể xuất bản một "thư mục khoá" và cho phép mọi người chọn khoá ngẫu nhiên. Nếu có một danh bạ như vậy thì người ta có thể tìm một khoá riêng phù hợp với một khoá công khai ngay khi một người sử dụng chọn một cặp khóa và đưa ra khoá công khai. Bởi vậy, tính riêng tư của một khoá riêng phải được đảm bảo khi người sử dụng lựa chọn một khoá mới.
- Trung tâm phân phối khoá phải không biết khoá mà người dùng đã chọn là khoá nào? Trung tâm không chỉ đơn giản là phân phối một cặp khoá cho một người dùng, người dùng phải có một sự lựa chọn nào đó để trung tâm không thể biết chắc khoá nào là khoá mà người dùng đã chọn. Điều quan trọng là ta muốn trung tâm phân phối khoá đặt các khoá trong các "phong bì" được gắn kín trong một "thùng chứa" và để cho mỗi người dùng nhặt một phong bì.

Một trung tâm phân phối khoá tốt phải là một nguồn trung tâm tạo ra và mã hoá các khoá (giống như Ann trong ví dụ trước). Ann sẽ đưa ra một dòng liên tục các khoá mã hoá đã được mã. Vì các khoá đều được mã nên không ai có thể xác định được các khoá này.

Một người sử dụng riêng muốn có một khoá mới (giống như Bill) sẽ cho phép một ít khoá đã mã hoá của Ann chuyển tới và rồi chọn ngẫu nhiên một khoá $E\left(k_i, K_A\right)$. Sau đó (sau khi đã có đủ số khoá để Ann không thể biết được khoá mà Bill đã chọn), Bill sẽ mã hoá khoá đã chọn để tạo ra $E\left(E\left(k_i, K_A\right), K_B\right)$. Bill sẽ gửi khoá đã được mã hoá hai lần này cho Ann. Ann sẽ giải mã khoá này để bây giờ nó chỉ còn được mã bởi khoá của Bill và gửi nó trở lại cho Bill. Cuối cùng, Bill sẽ giải mã để sử dụng khoá này. Quá trình này tương tự như thủ

tục chơi bài đã được mô tả ở H5.18. Cần chú ý rằng đối với hệ thống mã khoá công khai, mỗi khoá từ Ann thực sự là một cặp khoá $E\left(\left(k_{P\cup B}, k_{PRIV}\right), K_A\right)$.



Hình 5.18. Thủ tục phân phối khoá mật

5.2.5. Bỏ phiếu bằng máy tính

Một vấn đề tương tự xoay quanh việc phát một thông báo xác thực nhưng không xác định được dấu vết. Chẳng hạn, cho phép một người (tham gia vào một thí nghiệm) trả lời một câu hỏi kín theo cách vô danh tính. Trong một trường hợp khác, các cử tri có thể phải được lập danh sách bằng máy tính. Một ví dụ thứ ba có liên quan đến các giao dịch riêng tư tự động (chẳng hạn như các tài khoản trong ngân hàng Thuỵ Sỹ), trong đó việc chuyển tiền là xác thực nhưng không thể truy tìm trở lại được người tạo giao dịch.

Trong cả ba trường hợp này ta cần có một thủ tục để đảm bảo bí mật và hợp thức. Mỗi thông báo trong hệ thống phải xuất phát từ một người sử dụng hợp pháp nhưng không một người sử dụng nào có thể gắn kết được với một thông báo riêng một khi nó được phát. Các yêu cầu bí mật của thủ tục này bao gồm:

- Chỉ có những người dùng hợp pháp mới có thể phát các thông báo.
- Ở một thời điểm, mỗi người dùng chỉ có thể phát một thông báo.
- Không một ai có thể xác định được ai là người đã gửi một thông báo riêng.

Demilo và Merritt đã thiết kế một thủ tục như vậy. Để chỉ rõ hoạt động của một thủ tục, ta sẽ xét một ví dụ bỏ phiếu. Giả sử có ba cử tri là Jan, Keith và Lee bỏ phiếu "Có" hoặc "Không" cho một vấn đề tranh cãi nào đó. Giả sử mỗi cử tri có hai hàm mã hoá khoá công khai. Hàm thứ nhất E là một hàm mã hoá công khai thông thường nhưng hàm thứ hai, R, sẽ gắn thông báo trong một xâu ngẫu nhiên và rồi mã hoá kết quả. D là hàm giải mã cho E, còn Q là hàm giải mã cho R. Vì có nhiều lớp hàm nên ta sẽ ký hiệu E_U là phép biến đổi của cá nhân U.

Hai người có thể có phiếu bầu như nhau. Thủ tục này phải cho phép mỗi người dùng biết được phiếu bầu của mình nhưng không thể biết được phiếu bầu của các cử tri khác cho dù có cùng ý kiến. Phép mã hoá R với xâu ngẫu nhiên phải cung cấp độ bảo mật mong muốn và khả năng phân biệt. Về hình thức, các phiếu bầu được gửi kín ví hai phiếu như nhau đều được gắn với các xâu ngẫu nhiên khác nhau. Tuy nhiên, mỗi cử tri đều có thể xác định đúng phiếu bầu của mình. Vì mỗi phép giải mã R của mỗi người là của riêng từng người nên chỉ có người gửi mới có thể xác nhận rằng phiếu của mình có nằm trong số phiếu được tính hay không?

5.2.5.1. Thủ tục bỏ phiếu

Mỗi cử tri chọn một phiếu v và tính:

$$R_J\left(R_K\left(R_L\left(E_J\left(E_K\left(E_L\left(v\right)\right)\right)\right)\right)\right)$$

bằng cách dùng các phép mã hoá khoá công khai. Tất cả các phiếu đã mã hoá sẽ được gửi cho cử tri thứ nhất là Jan. Jan sẽ kiểm tra xem lá phiếu của mình có nằm trong số các phiếu đã nhận hay không? Jan sẽ giải mã cho mức mã hoá đầu tiên cho tất cả các lá phiếu:

$$Q_J\left(R_J\left(R_K\left(R_L\left(E_J\left(E_K\left(E_L\left(v\right)\right)\right)\right)\right)\right)\right)=R_K\left(R_L\left(E_J\left(E_K\left(E_L\left(v\right)\right)\right)\right)\right)$$

vì

$$Q_J\left(R_J\left(x\right)\right)=x$$

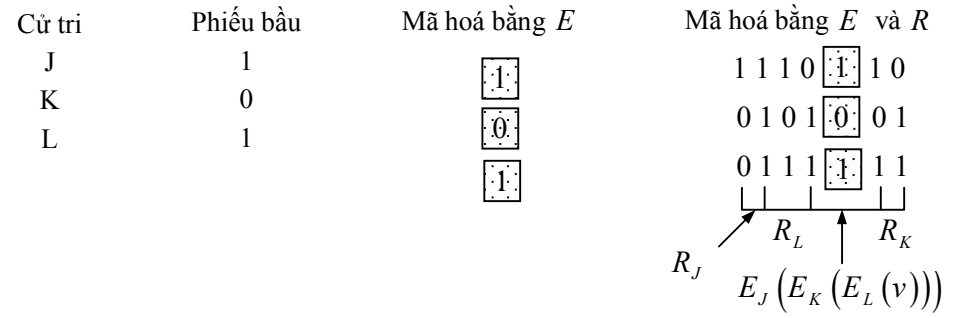
Bây giờ Jan sẽ gửi các lá phiếu tới Keith. Keith sẽ kiểm tra lá phiếu của mình và giải mã mức một để tạo ra:

$$Q_K\left(R_K\left(R_L\left(E_J\left(E_K\left(E_L\left(v\right)\right)\right)\right)\right)\right)=R_L\left(E_J\left(E_K\left(E_L\left(v\right)\right)\right)\right)$$

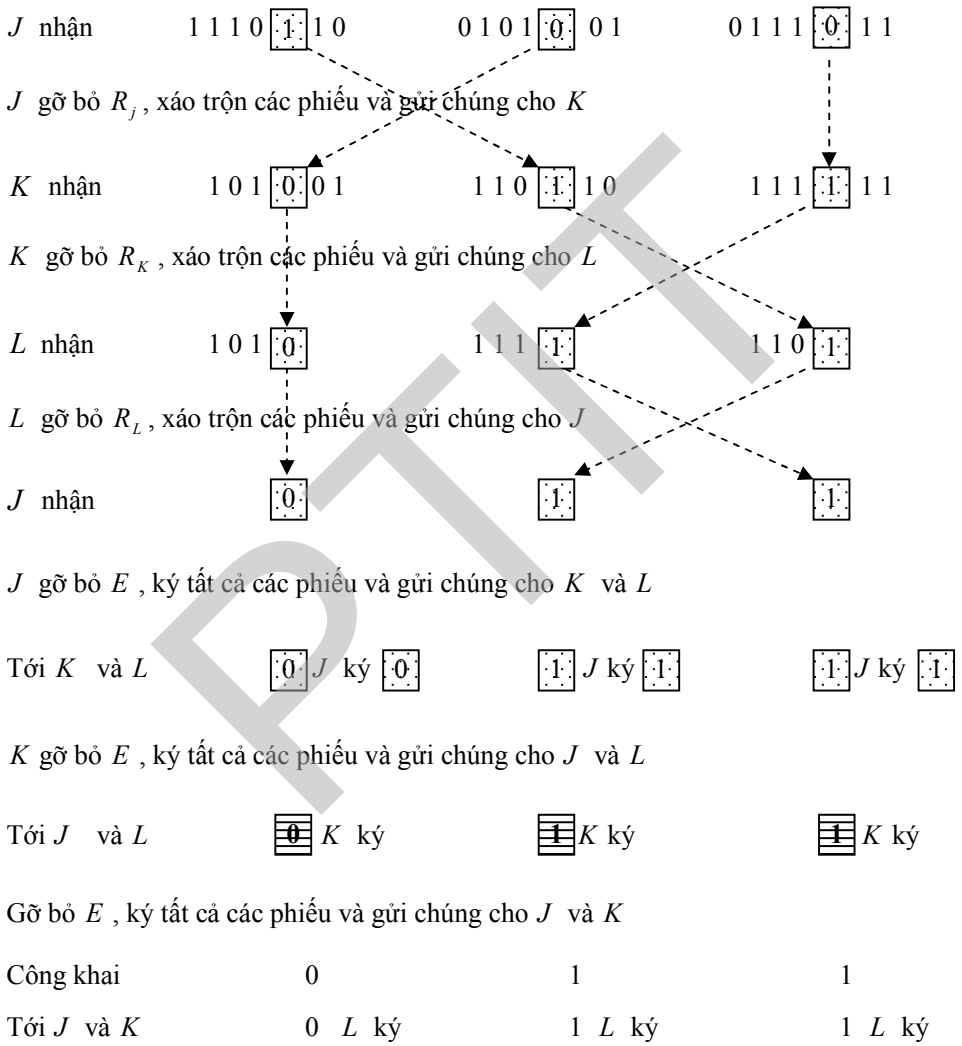
Keith sẽ gửi kết quả này cho Lee. Lee sẽ gửi $E_J\left(E_K\left(E_L\left(v\right)\right)\right)$ của tất cả các v cho Jan. Lee cũng ký các biểu quyết bằng một chữ ký số và gửi chữ ký này cho Jan và Keith.

Jan sẽ giải mã thêm một mức nữa và kiểm tra để thấy rằng biểu quyết của mình là vẫn còn nằm trong tập. Jan sẽ gửi các lá phiếu cho Keith và gửi các chữ ký số của các lá phiếu cho Keith và Lee. Keith sẽ thu được $E_K\left(E_L\left(v\right)\right)$ từ Jan. Từ đó, Keith sẽ giải mã để tạo ra $E_L\left(v\right)$ cho Lee. Lee sẽ giải mã E_L và công bố các kết quả.

Thủ tục này sẽ làm việc vì quá trình phân tích là mỗi chuỗi 6 liên kết: $J\rightarrow K\rightarrow L\rightarrow J\rightarrow K\rightarrow L$. Các kết quả ở mỗi kết nối của chuỗi có thể được làm công khai mà không phá huỷ tính vô danh tính của một lá phiếu bất kỳ. Hơn nữa, trong ba liên kết cuối của chuỗi, bất cứ ai cũng có thể đi "ngược" trở lại nhưng chỉ có một người có thể đi "xuôi". Tức là, giả sử Jan biến đổi các lá phiếu và chuyển chúng cho Keith. Chỉ có Jan mới có thể thực hiện phép biến đổi này. Nhưng Keith hoặc bất cứ một ai khác có thể thực hiện phép biến đổi ngược $\left(J\leftarrow K\right)$ để xem liệu cái mà Jan đã chuyển cho Keith có phù hợp với cái mà Jan đã bắt đầu xử lý. Điều này nhằm kiểm tra công việc của nhau và ngăn chặn sự gian lận của bất cứ ai.



J, K và L tính các phiếu đã mã hoá của họ; K và L gửi các lá phiếu của mình cho J.



Hình 5.19. Thủ tục bỏ phiếu

5.2.5.2. Phân tích thủ tục

Ta sẽ phân tích mỗi bước trong thủ tục này. Trước tiên, ta thấy không có thông tin nào để gắn một lá phiếu với một cá nhân nào và như vậy, các lá phiếu vẫn giữ được bí mật. Thứ hai là mỗi cá nhân chỉ có thể bỏ phiếu được một lần vì chỉ có 3 lá phiếu và mọi người sẽ chứng thực ở vòng đầu tiên là lá phiếu của mình nằm trong 3 lá phiếu này. Cuối cùng, không

một ai ngoài 3 người có thể bỏ phiếu được vì một trong 3 cử tri sẽ thấy được rằng mình đã bị mất phiếu.

Giả sử ở vòng thứ hai có một người nào đó quyết định can thiệp bất chính vào một lá phiếu. Một khi kết quả đã được công bố thì 3 cử tri có thể giải mã các phiếu kết quả bằng R để kiểm tra rằng các phiếu này phù hợp với tập đã gửi ban đầu. Bởi vì mỗi một cử tri đều ký cho tập các lá phiếu cần được chuyển tiếp nên có thể xác định được ai đã bóp méo các kết quả.

Hình 5.19 mô tả một ví dụ về thủ tục này. Để đơn giản các phép mã hoá kiểu E được mô tả bằng các vạch trực tiếp còn các phép mã hoá kiểu R được mô tả bằng cách Jan thêm vào 1 bit ngẫu nhiên về phía trái, Keith thêm vào 2 bit ngẫu nhiên về phía phải và Lee thêm vào 3 bit ngẫu nhiên về phía trái.

5.2.6. Chuyển giao không nhớ

Thủ tục sau đây là một công cụ sẽ được sử dụng trong nhiều thủ tục phức tạp hơn sau này. Ở đây, bài toán xoay quanh việc gửi một trong hai thông báo với một hạn chế là không một ai (kể cả người gửi và người nhận) biết được thông báo nào đã được gửi. Một ví dụ của bài toán này là việc gieo đồng tiền từ xa. Một người (người gửi) sẽ tung đồng tiền và viết kết quả (xấp hoặc ngửa). Người khác (người thu) sẽ viết ra dự báo kết quả. Sau đó hai người sẽ gặp nhau và trao đổi các biên bản. Khi đó, nếu các kết quả trên các biên bản giống nhau thì người thu sẽ thắng, ngược lại là người gửi sẽ thắng.

Giả sử rằng hai người không thể gặp nhau để trao đổi các biên bản. Hãy thử hình dung tình hình sau: Pete và Nancy rủ nhau đi chơi vào buổi tối. Trên điện thoại. Pete đề nghị tung một đồng tiền, nếu xuất hiện mặt ngửa thì Pete sẽ trả tiền ăn và tiền vé xem phim, nếu xuất hiện mặt sấp thì Nancy sẽ trả. Nancy trả lời rằng ý kiến đó rất hay đối với cô ta. Pete biết rằng nếu Nancy tung đồng tiền thì cô ta sẽ báo rằng "mặt ngửa xuất hiện" bất chấp kết quả ra sao; Pete biết điều đó vì anh ta cũng sẽ xử sự như vậy. Tuy nhiên, họ đã quyết định qua điện thoại để người còn lại phải ra ngân hàng nhận tiền cho buổi tối.

Vấn đề này được gọi là vấn đề chuyển giao không nhớ. Pete muốn gửi một trong hai thông báo tới Nancy với xác suất xuất nhận của mỗi thông báo là 0.5. Thông báo đầu tiên của Pete có thể là "Tôi sẽ trả. Pete". Nếu Nancy chỉ cho anh ta thông báo này thì anh ta sẽ có trách nhiệm chi trả, thông báo khác mà Nancy có thể nhận là một sự bóp méo vô nghĩa, trong trường hợp này cô ta sẽ phải chi trả. Nếu xác suất nhận được của mỗi một trong các thông báo của Pete là 0.5 thì nó tương đương với việc tung đồng tiền một cách thông minh.

5.2.6.1. Thủ tục chuyển

Thủ tục ban đầu do Blum đưa ra [12]. Giải pháp của Even [13] đưa ra rất dễ hiểu, mặc dù nó cần có một số khoá mã hoá. Sau đây là các bước của thủ tục:

1. Pete chọn hai cặp khoá mã hoá công khai (tổng cộng là 4 khoá). Ta ký hiệu các khoá này là E_i, D_i, E_j và D_j . E_i là một phép biến đổi công khai với khoá i và D_i là phép biến đổi riêng tương ứng của nó. (Tức là $D_i\left(E_i\left(M\right)\right)=M$ với thông báo M bất kỳ).

2. Nancy chọn một khoá K_N cho một thuật toán mã hoá đối xứng S.

3. Pete gửi cả hai khoá công khai (E_i, E_j) cho Nancy và giữ lại các khoá riêng.

4. Nancy chọn ngẫu nhiên một khoá và gửi $E_h(K_N)$ cho Pete, tức là cô ta mã hoá khoá đối xứng K_N của mình bằng một khoá công khai E_h.

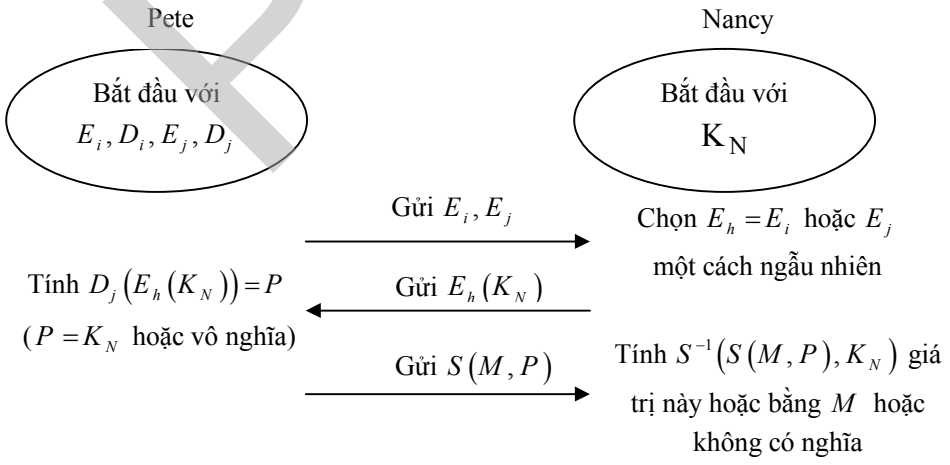
5. Pete chọn ngẫu nhiên hoặc i hoặc j (giả sử Pete chọn j), Pete tính $P = D_j(E_h(K_N))$. Đối với Pete thì đây là một khâu nhị phân, bởi vậy anh ta không thể biết liệu j có phải là giá trị mã Nancy đã chọn hay không?. Nếu $h = j$ thì P là khoá K_N của Nancy, ngược lại thì P là một khâu nhị phân vô nghĩa.

6. Pete tính S ("Tôi sẽ trả. Pete", P) rồi gửi cho Nancy, cùng với giá trị j.

7. Nancy giải mã thông báo của Pete bằng K_N. Cô ta đã chọn h và Pete đã chọn j. Nếu $h = j$ thì cô ta thắng, khi giải mã đúng thông báo của Pete, cô ta sẽ gửi M và h cho Pete để chứng thực rằng cô ta đã chọn đúng. Nếu $h \neq j$ thì cô ta thua, kết quả giải mã chỉ là một xâu bit vô nghĩa.

8. Sau khi người thắng cuộc đã được xác định, Pete sẽ trao các khoá riêng là D_i và D_j cho Nancy. Từ các khoá này, cô ta có thể kiểm tra xem h đã chọn là i hay j và xem liệu j mà Pete đã dùng có phải là một trong hai giá trị này hay không? Nếu $h = j$ và cô ta không nhận được thông báo thì cô ta có thể kết luận rằng Pete đã ăn gian.

Các bước của thủ tục này được mô tả trên hình 5.20.



Hình 5.20. Chuyển không nhớ

5.2.6.2. Phân tích thủ tục

Thủ tục này sẽ làm việc như thế nào? Trước tiên, Nancy chọn một trong các khoá của Pete, cô ta sẽ "ngụy trang" nó và trả lại cho Pete. Pete không biết khoá mà Nancy đã chọn bởi vì anh ta không thể biết K_N. Bởi vậy, khi anh ta chọn j thì cơ may để anh ta chọn đúng khoá

mà Nancy đã chọn chỉ là 1 trong 2. Anh ta sẽ mã hoá thông báo mà Nancy phải giải đúng để thuyết phục anh ta là anh ta đã chọn đúng khoá của cô ta. Nếu họ cùng chọn một khoá thì khoá K_N của Nancy sẽ giải mã được thông báo, ngược lại thì K_N sẽ tạo ra một khâu vô nghĩa. Sau đó, khi gặp nhau họ có thể trao đổi các khoá và kiểm tra được mỗi bên có tuân theo đúng các quy tắc của thủ tục hay không?

Dĩ nhiên là việc tung đồng tiền qua điện thoại không phải là một công việc rất quan trọng. Tuy nhiên, thủ tục này là cơ sở cho 2 thủ tục mà ta sẽ xem xét dưới đây.

5.2.7. Ký thoả thuận

Giả sử rằng Charles và Diane cùng thống nhất với nhau về một vấn đề nào đó và muốn ký một thoả thuận để xác nhận sự đồng ý của họ. Cả hai phải cam kết thực hiện một công việc nào đó theo thoả thuận nhưng người nào cũng chỉ muốn cam kết khi người khác cũng làm như vậy. Ví dụ, Charles cam kết bán xe của anh ta cho Diane nếu Diane đồng ý trao cho anh ta một cổ phần trong việc kinh doanh bán bánh pizza của cô ấy.

Charles ở California và Diane ở NewYork. Charles sẽ không ký thoả thuận trước và gửi nó cho Diane, bởi vì điều đó sẽ đặt số phận của anh ta vào bàn tay của Diane: anh ta đã đồng ý bán xe, bởi vậy anh ta phải đưa nó ra khỏi gian hàng. Tuy nhiên, Diane có thể xé bản thoả thuận để mặc Charles với chiếc xe; mặt khác Diane có thể ký thoả thuận bắt buộc Charles phải sản xuất xe. Charles sẽ bị ràng buộc bởi thoả thuận ngay khi anh ta ký. Tuy nhiên, anh ta không biết liệu Diane có ký hay không, bởi vậy, cô ta không bị ràng buộc bởi bản thoả thuận. Tình hình cũng tương tự nếu Diane ký trước: cô ta sẽ bị ràng buộc còn Charles thì không.

Trong thực tế, điều này được khắc phục khi cả hai bên cùng ký. Ở đây chúng ta muốn thiết lập một thủ tục để ký các thoả thuận bằng máy tính. Giải pháp này nhằm tránh phải gặp gỡ nhau và ký trên giấy.

Có một giải pháp phải nhờ tới một bên thứ 3 tin cậy: Charles và Diane, mỗi người phải ký vào một bản sao của bản thoả thuận để gửi cho một người thứ 3 tin cậy, người này sẽ giữ hai bản sao có một chữ ký này. Đây là bằng chứng chứng tỏ rằng họ đều muốn ký thoả thuận. Khi bên thứ 3 công bố rằng anh ta đã nhận được các bản sao được ký từ mỗi bên thì Charles sẽ ký một bản sao khác để gửi cho Diane và Diane cũng ký vào một bản sao khác để gửi cho Charles. Charles và Diane phải báo cho bên thứ 3 biết và bên thứ ba sẽ huỷ các thoả thuận chỉ có một chữ ký. Tuy vậy, ta vẫn muốn không phải sử dụng tới bên thứ 3 nếu có thể.

Bởi vậy, một thủ tục thoả thuận cần có hai điều:

- Cam kết. Sau một thời điểm nhất định, cả hai bên sẽ bị ràng buộc bởi bản thoả thuận cho tới một thời điểm nào đó.
- Tính không thể giả mạo. Các chữ ký trên bản thoả thuận phải chứng tỏ được là xác thực. Tức là mỗi bên phải có khả năng chứng minh rằng chữ ký của bên còn lại là xác thực.

Thủ tục phải có khả năng thu được hai vấn đề nêu trên một cách gián tiếp, không cần phải mặt đối mặt. Một thủ tục gián tiếp là thích hợp trong tình huống đã được máy tính hoá.

5.2.7.1. Với giấy và bút

Một thủ tục gián tiếp có thể hoạt động theo cách sau. Thủ tục này có thể không được sử dụng trên thực tế, nhưng nó tương tự với một thủ tục máy tính được mô tả ở phần sau. Charles và Diane sẽ chuyển tới lui ít nhất 3 bản thoả thuận. Mỗi người giữ một bản làm bằng chứng về sự việc đã xảu ra còn một bản đang được chuyển. Việc ai giữ bản sao nào cho ai để giữ một bản làm bằng chứng là rất rõ ràng, bởi vậy ta sẽ không mô tả các bản sao khác nhau lưu chuyển tới lui.

Charles sẽ viết chữ cái đầu tiên trong tên của mình lên bản thoả thuận là chữ "C" và gửi nó cho Diane. Cô ta sẽ viết chữ "D" và gửi trở lại. Charles sẽ viết một chữ cái tiếp theo trong tên của mình "h" và cứ như vậy họ tiếp tục thực hiện trao đổi bằng cách mỗi lần gắn thêm một chữ cái trong chữ ký. Chỉ với chữ "C" trên bản thoả thuận, Charles biết rằng không một ai có thể yêu cầu anh ta thực hiện bản thoả thuận. Tuy nhiên, mỗi chữ cái là một hành động chứng tỏ sự trung thực mà anh ta muốn tỏ cho Diane thấy. Diane cũng sẽ đáp ứng bằng những hành động tương tự.

Sau một vài chữ cái trên mỗi chữ ký, Diane biết rằng cô ta có thể thuyết phục Quan toà là Charles thực sự đã bị ràng buộc bởi bản thoả thuận . Cô ta giải thích các tình huống và hy vọng rằng Quan toà sẽ đồng ý. Ở đây vẫn còn có một độ bất định: Charles không bị ràng buộc bởi các chữ cái "C' mà anh ta chỉ bị ràng buộc bởi chữ ký đầy đủ của mình. Tuy nhiên, thời điểm mà anh ta bắt đầu bị ràng buộc là không rõ ràng: khi có $\frac{1}{2}$, $\frac{2}{3}$ hay $\frac{3}{4}$ số chữ cái trong chữ ký?

Bởi vì họ không biết chắc đúng vào thời điểm nào họ bị ràng buộc nên cả hai đều lo ngại họ sẽ bị ràng buộc ở mỗi lần, bởi vậy họ không có lý do nào để không tiếp tục thủ tục. Họ cũng biết rằng càng tiếp tục thì xác suất chứng tỏ rằng bên kia cũng bị ràng buộc càng lớn, bởi vậy mối quan tâm lớn nhất của họ là phải tiếp tục công việc. Ngoài ra, cả hai đều muốn ký thoả ước bằng bất cứ cách nào nhưng phải đảm bảo họ bị ràng buộc cùng lúc. Tính không chắc chắn của điểm cam kết là cơ sở của giải pháp máy tính.

5.2.7.2. Tóm lược về thủ tục gián tiếp

Thủ tục máy tính làm việc khá giống cách trên. Các bên trao đổi thông báo thoả thuận theo các mẩu bằng cách sử dụng thủ tục chuyển không nhớ. Cần nhớ lại là việc chuyển không nhớ là cách phát một trong hai thông báo để không một ai (cả người phát lẫn người nhận) biết chắc thông báo nào đã được phát. Bởi vậy, không một bên nào biết được một nửa thông báo đã được phát bởi vì với việc chuyển không nhớ, không một ai biết chính xác cái mà người khác đã nhận được.

Giả sử mỗi chữ ký của mỗi người được chia thành 4 khối và không thể ghi nhận chữ ký bằng cách nhìn vào nó. Charles và Dianne thực hiện việc chuyển không nhớ mỗi người 8 lần Charles sẽ gửi mỗi khối một lần theo i và một lần theo j . Diane cũng làm như vậy.

Trong 3 lần đầu, Charles cảm thấy an toàn vì Diane không có đủ 4 khối trong chữ ký của anh ta. Charles sẽ gửi khối thứ 4. Với khối này, Diane có thể có một chữ ký đầy đủ nhưng cô ta không thể biết điều đó. Trên thực tế, xác suất để cô ta nhận đủ 4 khối của chữ ký là khá

thấp (nhỏ hơn 0.02). Nếu cô ta dừng lại thì Charles có thể đã có $\frac{3}{4}$ chữ ký của cô ta và có thể thuyết phục Quan toà rằng Diane đã phải bội lại thoả thuận. Diane không có sự lựa chọn nào khác ngoài việc tiếp tục . Khi họ tiếp tục công việc, họ sẽ ngày càng tiến gần hơn tới việc đảm bảo thoả thuận.

Ở một thời điểm bất kỳ sau 4 vòng không một ai có thể dừng lại, bởi vì cô ta hoặc anh ta có thể đã phát đi một chữ ký đầy đủ và bởi vậy, đã bị ràng buộc bởi bản thoả thuận. Do đó, điều quan tâm lớn nhất của cả hai là phải tiếp tục công việc cho tới khi kết thúc, là khi các chữ ký đầy đủ đã được trao đổi hoàn toàn xác định.

5.2.7.3. Thủ tục ký thoả thuận gián tiếp

Sau đây là thủ tục do Even và các cộng sự đưa ra [13].

1) Charles chọn ngẫu nhiên $2n$ khoá của một hệ mật khoá đối xứng (chẳng hạn, DES). Ta ký hiệu chúng là $c_1, c_2, \ldots, c_{2n}$. Các khoá được gom thành các cặp sau (c_1, c_{n+1}), (c_2, c_{n+2}), ... Đây chỉ là một cách ký hiệu chứ không có sự phân biệt nào giữa c_i và c_{n+i}.

2) Với mỗi khoá, Charles tính $C_i = E(S, c_i)$ đối với một thông báo chuẩn nào đó mà nội dung của nó là không có liên quan. Charles sẽ gửi C_1 tới C_{2n} cho Diane. Tức là Diane có dạng đã mã hoá của thông báo chuẩn dưới mỗi khoá.

3) Charles nhất trí rằng anh ta sẽ bị ràng buộc bởi bản thoả thuận nếu Diane có thể đưa ra cả hai khoá c_i và c_{n+i} với i bất kỳ. (Chú ý: theo mô tả của Even, mỗi C_i được gọi là một thách đố S và khoá (c_i, c_{n+i}) được gọi là giải pháp).

4) Diane lặp lại 3 bước này theo cách tương tự với các khoá d_i và các thông báo đã mã D_i.

5) Charles sẽ gửi mỗi cặp c_i và c_{n+i}, $1 \le i \le n$ cho Diane qua thủ tục chuyển không nhớ. Tức là, Charles sẽ gửi hoặc c_i, hoặc c_{n+i} cho Diane nhưng cả Charles lẫn Diane đều không biết chắc khoá đã nhận được. Diane cũng làm tương tự với tất cả các d_i và d_{n+i} với $1 \le i \le n$. Tới lúc này, Charles và Diane mỗi người đã có một nửa bí mật của người khác.

6) Gọi l là độ dài của mỗi c_i hoặc d_i.

   For $1 \le j \le l$ begin

      Charles phát bit thứ j của tất cả các $c_i, 1 \le i \le n$ cho Diane

      Diane phát bit thứ j của tất cả các $d_i, 1 \le i \le n$ cho Charles

   End

7) Giả sử rằng không một ai kết thúc sớm, ở cuối thư mục cả hai đều có tất cả l bit của tất cả các bí mật của nhau và thoả thuận đã được ký.

Khi bước 6 bắt đầu, cả hai bên Charles và Diane đều có một nửa bí mật của người khác, tuy nhiên Diane vẫn không xác định được khoá c_i từ thông báo đã mã C_i của nó cho dù Diane đã

có c_{n+i}. Vì bước này đang tiếp diễn nên Diane sẽ nhận được 1 bit của c_i, rồi bit tiếp theo, rồi bit tiếp nữa... Nếu Diane sử dụng một phép tấn công mạnh để xác định c_i thì mỗi bit cô ta nhận được sẽ làm giảm nhẹ công việc của cô ta đi một nửa.

Ở thời điểm mà cô ta nghĩ rằng công việc cần làm còn lại là đủ dễ dàng để cô ta có thể xác định được c_i với i nào đó. Bởi vậy cô ta có thể ngừng việc gửi các bit cho Charles. Cũng vào lúc đó, Charles cũng có một lượng thông tin tương đương về mỗi d_i. Nếu Diane có thể dễ dàng xác định được c_i thì Charles cũng có thể dễ dàng xác định được d_k với k nào đó. Bởi vậy, thời gian mà Diane cần để gian lận cũng cho Charles thời gian để tìm ra giải pháp của cô ta để cả hai đều bị ràng buộc vào bản thoả thuận.

5.2.8. Thư tín được chứng thực

Việc gửi thư tín có chứng thực là vấn đề cuối cùng mà ta sẽ phát triển một giải pháp thủ tục. Giả sử Gina muốn gửi một thông báo cho Hal nhưng cô ta muốn chứng tỏ rằng Hal đã nhận được thông báo. Cô ta không muốn để lại thông báo cho Hal mà không nhận được phúc đáp. Dĩ nhiên là Hal sẽ không ký phúc đáp cho tới khi anh ta thực sự có thông báo. Các bưu tá và trọng tài có thể giải quyết được vấn đề này. Tuy nhiên, như thường lệ, ta muốn có một giải pháp không cần có trọng tài. Giải pháp này rất giống với thủ tục ký thoả thuận mà ta vẫn nêu. Trên thực tế, giải pháp này cũng do Even và các cộng sự đưa ra [13].

Giả sử thông báo mà Gina chuyển tới Hal là M. Cô ta sẽ phát M cho Hal ở dạng được mã hoá và sẽ chỉ trao cho Hal khoá mã hoá khi cô ta đã nhận được một thông tin xác nhận thích hợp từ Hal báo về việc anh ta đã nhận được M. Hal chỉ biết được M khi anh ta nhận được khoá cho phép anh ta thu được bản rõ M.

1) Gina chọn ngẫu nhiên $n+1$ khoá của một hệ mật đối xứng (chẳng hạn DES). Ta ký hiệu chúng là $g_0, g_1, \ldots, g_n$. Cô ta cũng tính $g_{n+i} = g_0 \oplus g_i$ với $1 \le i \le n$ (dấu $\oplus$ chỉ phép hoặc loại trừ - hay phép cộng theo modulo 2).

2) Gina tính $G = E(M, g_0)$ (mã hoá thông báo M bằng khoá g_0). Cô ta gửi G cho Hal. Vì với mỗi i:

$$\begin{aligned} g_{n+i} &= g_0 \oplus g_i \\ g_{n+i} + g_i &= g_0 \oplus g_i \oplus g_i \\ &= g_0 \oplus 0 = g_0 \end{aligned}$$

Bởi vậy Hal có thể xác định được M nếu biết một cặp (g_i, g_{n+i}) bất kỳ.

3) Gina tính và gửi cho Hal $G_i = E(S, g_i)$ với $1 \le i \le 2n$ trong đó S là một thông báo chuẩn nào đó mà nộ dung của nó là không quan trọng. Vào lúc này Hal đã có thông báo đã mã G và một thông báo tiêu chuẩn S được mã bằng $2n$ khoá khác nhau của Gina.

4) Hal chọn $2n$ khoá $h_1, h_2, \ldots, h_{2n}$. Hal tính $H_i = E(S, h_i)$ với $1 \le i \le 2n$. Hal gửi H_i cho Gina.

5) Hal gửi một thông báo cho Gina nói rằng anh ra sẽ báo việc nhận bản vẽ của G nếu Gina có thể tạo ra một trong các cặp của Hal (h_i và h_{n+i}) và tất cả các g_j với $1 \le j \le 2n$.

Khi nói điều này, Hal nói rằng nếu Gina có thời gian để xác định một trong các cặp khoá của anh ta (h_i và h_{n+i}) thì anh ta cũng có thời gian để xác định một trong các cặp khoá của cô ta (g_m và g_{n+m}). Chừng nào có được một cặp khoá, Hal có thể tính được khoá mã hoá g_0 và thu được thông báo $M = D(G, g_0)$.

Điều kiện thứ hai trong tuyên bố của Hal đảm bảo rằng Gina đã xử sự công minh, tức là mỗi cặp đều dẫn ra được g_0.

6) Cũng như trong thủ tục ký thoả thuận. Gina và Hal sẽ trao đổi các cặp (g_1, g_{n+1}), (g_2, g_{n+2}),... và (h_1, h_{n+1}), (h_2, h_{n+2}),... theo thủ tục chuyển không nhớ. Tức là, Hal sẽ nhận, hoặc g_1, hoặc g_{n+1} nhưng cả Hal lẫn Gina đều không biết là giá trị nào? Gina sẽ nhận một nửa trong mỗi cặp (h_i, h_{n+i}) mà không biết giá trị nào.

7) Cũng như trong thủ tục ký thoả thuận, giả sử l là độ dài của mỗi khoá g_i.

For $j = 1$ to l begin

Gina gửi cho Hal bit thứ j của mỗi g_i với $1 \le i \le 2n$.

Hal gửi cho Gina bit thứ j của mỗi h_i với $1 \le i \le 2n$.

end

Khối lượng công việc cần làm để phá mã của người khác cũng giống như trong việc ký thoả thuận. Bởi vậy, tốt hơn cả là tiếp tục công việc. Mặc dù cả hai thủ tục ký thoả thuận và thư tín có chứng thực là khá phức tạp đối với con người nhưng chúng lại dễ dàng thích hợp trong liên lạc máy tính.

5.3. SỬ DỤNG MÃ HOÁ NHƯ THẾ NÀO

Trong những phần trước, ta đã nghiên cứu một số ví dụ về các thủ tục để sử dụng trong máy tính. Ta đã coi các thủ tục là các dãy có thứ tự của các bước để giao tiếp giữa hai bên. Các phương pháp đã đưa ra các cách thông minh trong khi vẫn giữ được tính bí mật, tính vô danh tính hoặc tính riêng tư.

Các thủ tục sử dụng mã hoá như một công cụ để thu được tính bí mật, tính xác thực hoặc tính toàn vẹn, mặc dù trọng tâm là nằm ở trên thủ tục và các đặc tính an toàn cần đạt được chứ không phải ở bản thân phép mã hoá. Các thủ tục (không phụ thuộc vào một thuật toán mã hoá riêng nào) chỉ giả định sự tồn tại của một kiểu mã hoá (đối xứng hoặc không đối xứng) có thể có một tính chất chung (chẳng hạn như tính giao hoán).

Trong phần này ta sẽ nghiên cứu việc sử dụng các phép mã hoá trong thực tiễn. Mặc dù các ứng dụng thực tiễn này không hình thức như các thủ tục nhưng chúng sẽ hướng dẫn ta trong việc sử dụng các phép mã hoá để thực hiện các nhiệm vụ trên máy tính. Một số tình huống trong máy tính chỉ liên quan tới một người sử dụng. Ở đây chẳng cần tới một quy tắc

nào vì không có một sự không nhất trí nào về tính đúng đắn hoặc về sự tuân thủ đối với một quy tắc. Trong những trường hợp này, tính bí mật và tính xác thực là những vấn đề quan trọng. Những tình huống này được gọi là các thực tế mã hoá hay các kỹ thuật mã hoá. Chúng là chủ đề của phần còn lại trong chương này.

Chúng ta phải thông thạo về các đặc tính, các ưu nhược điểm của thuật toán mã hoá. Khi đó ta có thể chọn các thuật toán một cách đúng đắn và sử dụng những phương pháp mã hoá thích hợp. Trong phần này ta sẽ xem xét các tiêu chuẩn để phán xét các hệ thống mã hoá riêng. Hai sơ đồ chủ yếu cần xem xét là RSA và DES. Ta cũng nghiên cứu những hạn chế chung của các hệ thống khoá đối xứng và khoá không đối xứng.

Ta biết rằng, Shannon đã đề xuất các tiêu chuẩn chung cho các hệ thống mã hoá. Mặc dù các tiêu chuẩn này đã được xây dựng trước khi phổ cập các máy tính - modem nhưng chúng vẫn còn có khả năng áp dụng vào những vấn đề cần quan tâm đến ngày nay. Ta sẽ nhắc lại các tiêu chuẩn này.

- Mức độ bí mật cần thiết sẽ xác định khối lượng công việc thích hợp để mã hoá và giải mã.
- Tập các khoá không liên quan tới độ phức tạp.
- Ứng dụng phải đơn giản như có thể được.
- Các sai sót trong mã hoá phải không lan truyền và gây nên ngưng trệ những thông tin tiếp sau trong thông báo.
- Kích thước của bản mã phải không lớn hơn kích thước của thông báo gốc.

Tiêu chuẩn đầu tiên là cơ sở của các hệ mật. Tiêu chuẩn thứ hai là quan trọng hơn khi các khoá (và toàn bộ hệ thống mã hoá) được tính bằng tay. Với các máy tính để thực hiện một công việc phức tạp hay tẻ nhạt thì độ phức tạp của việc chọn khoá không phải là vấn đề. Trên thực tế, với các thuật toán toán học như RSA hay ElGamal thì các khoá là rất phức tạp theo nghĩa chỉ có những số nguyên hoặc những cặp số nguyên nhất định mới có thể là khoá. Tuy nhiên, khi việc trao đổi khoá là cần thiết hoặc mong muốn thì việc phân phối khoá tới những người sử dụng cũng là một khó khăn. Ứng dụng một sơ đồ mã hoá vẫn cần thiết phải đơn giản. Tuy nhiên, với việc sử dụng máy tính, các thuật toán trước kia không có khả năng thực hiện thì này đã trở nên có thể ứng dụng. Việc lan truyền sai được mô tả trong tiêu chuẩn thứ tư vẫn còn phải lưu tâm. Cuối cùng, tiêu chuẩn về kích thước bản mã không còn quan trọng nữa.

Ta sẽ xem xét các tiêu chuẩn này chi tiết hơn trong phần sau.

5.3.1. Mức độ bảo mật

Cuộc tranh luận liên quan tới việc xác nhận lại DES là một ví dụ tốt cho sự thích ứng của bảo mật ngày nay. Việc DES có thể bị thám nếu có đủ thời gian và tài nguyên tính toán vẫn chưa đủ để người ta loại bỏ nó. (Có thể coi DES chính thức bị loại vào cuối năm 1998, Triple - DES tạm thời thay cho DES trong khi chờ đợi chuẩn mới AES ra đời (AES - Advanced Eneryption Standard - Chuẩn mật mã tiên tiến) dự kiến đưa ra vào năm 2001, các

thảo luận trên mạng đang thúc tiến việc xây dựng AES có khoá 256 bit, chẳng hạn như các thuật toán RC-6, CAST-256,...).

Từ 1979, Hellman đã chứng tỏ rằng một thám mã có chủ đích có thể giải mã một thông báo được mã bằng DES. Mặc dù giá các chip giải mã ngày càng rẻ nhưng chi phí để xây dựng một máy giải mã DES vẫn còn lớn tới mức khó có thể có một công ty tư nhân nào, hoặc thậm chí một tổ chức tội phạm nào hay một cơ quan chính phủ nào dám bỏ ra. Một số người sử dụng mã hoá chỉ cần giữ được bí mật đối với những thám mã tình cờ chứ không nhất thiết giữ được bí mật đối với các thám mã thực sự muốn phá. Ví dụ, một công ty có thể mã hoá các dữ liệu có liên quan tới một dòng sản phẩm mới để duy trì ưu thế cạnh tranh. Các ngân hàng mã hoá các thông báo để chuyển các tài khoản nhằm tránh các sửa đổi bất hợp pháp và để xác định việc truyền giả mạo các sai chứ không nhằm bảo vệ tính riêng tư. Một người sử dụng mã hoá file chứa chương trình nguồn nhằm tránh các sửa đổi bất hợp pháp.

Trong những trường hợp này và trong nhiều trường hợp khác, giá trị của thông tin được mã hoá phải được cân nhắc với chi phí thám mã. Khi đó, DES vẫn được coi là đủ mức an toàn. Đối với hệ mật RSA, tình hình cũng tương tự như vậy. Kể từ khi được đưa vào năm 1978 tới nay, vẫn chưa có một thách đố nghiêm trọng nào được đưa ra đối với tính an toàn của nó.

Tóm lại, tính bảo mật của một hệ mật mã phải thích hợp với mức độ quan trọng của dữ liệu cần bảo vệ. Người sử dụng các hệ thống mật mã phải xem xét giá trị của dữ liệu cần bảo vệ khi chọn một hệ thống mật mã.

5.3.2. Quản lý khoá

Đối với các hệ thống mã hoá phức tạp hơn, các khoá không thể chỉ được chọn đơn giản một cách ngẫu nhiên. Các hệ thống khoá công khai thường cần 2 khoá trong một cặp được xác định rất cẩn thận. Ngay cả đối với các thuật toán đối xứng, việc tồn tại các khoá yếu cũng không cho phép tạo khoá hoàn toàn ngẫu nhiên. Vì các lý do này nên các khoá thường được tạo từ một trung tâm tạo khoá.

Khi xem xét tới vấn đề trên, việc phân phối khoá có thể là một nhiệm vụ chính. Khi tới lúc cần thay đổi khoá, mỗi một chủ sở hữu khoá phải được thông báo về việc thay đổi khoá và tất cả những người dùng phải bắt đầu sử dụng khoá mới đồng thời (để tránh có một số thông báo lại được mã hoá bằng khoá cũ thay vì bằng khoá mới và nhằm phân biệt giữa chúng).

Việc thay thế khoá cũng là một vấn đề vì các khoá cũ phải được lưu giữ để cho phép giải mã các thông báo (được mã bằng các khoá cũ) được lưu lại.

5.3.3. Các khoá bị mất (bị lộ)

Cả hai hệ thống khoá đối xứng và không đối xứng đều nhạy cảm đối với việc mất khoá, lộ khoá hoặc bị đánh cắp khoá. Cách xử lý duy nhất được biết là phải kiểm tra định kỳ. Người sử dụng phải biết khi nào một khoá có thể bị tổn hại. Nếu điều đó xảy ra thì người sử dụng phải báo cho kho trung tâm hoặc cho tất cả những ai có thể nhận các thông báo từ họ rằng khoá có khả năng đã bị lộ. Tất cả những nơi thu nhận các thông báo từ người sử dụng này phải được biết rằng họ phải nghi ngờ mọi thông báo nhận được sau ngày báo. Họ cũng phải

biết rằng họ không được gửi bất kỳ một thông báo nào nữa được mã hoá bằng khoá công khai của người đã bị lộ khoá.

Thủ tục này (không thực sự là một thủ tục tối ưu) là một thủ tục khá hữu dụng. Một người dùng muốn rút lại một thoả thuận đã ký chỉ cần tuyên bố đã lộ khoá riêng ở thời điểm trước ngày ký thoả thuận. Bởi vì khoá riêng đã được coi là lộ nên bất cứ ai cũng có thể làm như người dùng (đã bị mất khoá).

5.3.4. Độ phức tạp mã hoá

Ta sẽ xem xét tiếp tới tính phức tạp để thực hiện một phép mã hoá. Có hai vấn đề chính ở đây là độ trễ trước khi thuật toán mã hoá có thể bắt đầu tạo ra bản mã và độ chậm của bản thân thuật toán mã hoá.

5.3.4.1. Độ trễ trước khi mã hoá

Độ trễ trước khi mã hoá bắt đầu phụ thuộc vào loại mã là mã khối hay mã dòng và kích thước của khối. Như đã mô tả trong chương 2, các mã dòng là thích hợp để thuật toán mã hoá có thể mã hoá ngay mỗi khi có một ký tự xuất hiện. Kém thích hợp hơn một chút là các mã dòng làm việc trên các khối. Mỗi ký tự không thể được mã hoá ngay khi nó xuất hiện nhưng các ký tự mới sẽ được giữ cho tới khi nhận được một khối (ví dụ 8 ký tự 8 bit hay 64 bit đối với DES). Các hàm mã hoá ít thích hợp nhất là các hàm mã toàn bộ thông báo rõ phải được nhận trước khi có thể bắt đầu việc mã hoá.

Nhiều thuật toán mật mã thông thường (bao gồm RSA, Merkle - Hellman, ElGamal, DES và Skipjack) là các mật mã khối. Với DES, một khối gồm 64 bit. Với RSA, không có độ dài khối bắt buộc (mặc dù các khối ngắn sẽ có độ mật hạn chế). Độ dài khối lớn nhất cũng là một tham số lựa chọn đối với người áp dụng.

Các nhà nghiên cứu phát triển RSA khuyến nghị độ dài khối nên từ 100 tới 200 bit. Các độ dài khối của DES và RSA thích hợp cho các ứng dụng trong thương mại.

5.3.4.2. Tốc độ mã hoá

Một số thuật toán mã hoá thực hiện một khối lượng công việc không đổi đối với một ký tự được mã. (Chẳng hạn, với các mật mã thay thế, tất cả công việc chủ yếu là quá trình tra cứu bảng). DES, Skipjack, RSA, Merkle - Hellman và ElGamal mặc dù là các thuật toán phức tạp hơn nhiều nhưng đều thực hiện cùng một lượng tính toán đối với một khối. Tất cả các thuật toán này hoạt động theo một thời gian tỉ lệ với kích thước của bản rõ, chỉ có hằng số tỉ lệ là khác nhau.

Tốc độ mã hoá là một tham số quan trọng vì nó đảm bảo rằng nếu một thuật toán mã hoá có thể mã hoá xong một khối trước khi nhận được khối thứ hai thì thuật toán này sẽ không làm giảm tốc độ của ứng dụng. Luôn có các ứng dụng phần cứng (chip) đối với hầu hết các thuật toán thông dụng đảm bảo làm việc với các tốc độ hợp lý.

Các thuật toán khoá công khai chậm hơn rất nhiều so với các thuật toán khoá đối xứng. Các thuật toán đối xứng đều dựa trên các phép toán hữu hiệu như các phép tra cứu bảng, dịch, chuyển vị và các phép logic (như VÀ, HOẶC, cộng mod 2...). Tất cả các phép toán này đều có các chức năng phần cứng tiêu chuẩn. Các thuật toán không đối xứng đều xây dựng trên các

bài toán khó. Với các khoá có độ dài càng lớn thì độ phức tạp tính toán càng tăng theo tỉ lệ hàm mã. Các số liệu trong bảng B 5.1 cho ta thấy rằng mã hoá đối xứng thực hiện nhanh hơn từ 1000 tới 5000 lần so với mật mã khoá công khai.

5.3.5. Lan truyền sai

Các sai trong truyền dẫn là chủ yếu và gây tác hạn nghiêm trọng. Ta cũng biết rằng các mạng cục bộ hoặc các mạng ở xa thường chịu các sai truyền dẫn. Các sai này có thể không báo cho ta biết có kẻ thu chặn. Các thuật toán mã hoá lý tưởng phải không bị ảnh hưởng của các sai của mạng. Thông thường, mạng sẽ chịu trách nhiệm xác định các cuộc liên lạc bị sai và phát lại các thông báo có lỗi sao cho người dùng không phải quan tâm đến.

Bảng 5.1: Tốc độ mã hoá

Thuật toán	Phần cứng (bits/s)	Phần mềm (bits/s./MIPS)
Mã hoá RSA	220K	0.5
Mã hoá RSA	220K	32K
MD4	220K	1300K
DES	1,26 (1,2. 10^9)	400K

Một sai thực sự (do sự thay đổi của thám mã) phải thể hiện rõ trên bản mã sao cho một sự thay đổi bất kỳ đều dễ dàng thấy được. Không có một thuật toán thông dụng nào có cơ chế chống xâm nhập. Nhờ các cơ chế này, thuật toán mã hoá sẽ bị bóp méo tới mức thậm chí chỉ cần thay đổi 1 bit đầu ra cũng sẽ dẫn tới thông báo được giải mã bị bóp méo nghiêm trọng. Nếu bản rõ là một đoạn văn thì sự thay đổi sẽ được thấy rất rõ. Nếu bản rõ là dữ liệu nhị phân thì sự thay đổi có thể không bị phát hiện.

5.3.6. Kích thước bản mã

Kích thước của bản mã được tạo thành cũng là một tham số quan trọng. Trong một số trường hợp, bản mã có thể nằm trong cùng một không gian với bản rõ. Khi đó, việc tăng một chút về kích thước cũng là điều không thể chấp nhận được. Cần chú ý rằng các mật mã khối phải mã hoá đồng thời một khối đầy đủ. Mặc dù có thể dễ dàng độn thêm vào khối cuối cùng không đầy đủ để tăng kích thước nhưng việc độn thêm này sẽ làm tăng kích thước của văn bản ở đầu ra. DES và Skipjack tạo ra một đầu ra có kích thước đúng bằng kích thước đầu vào hoặc hơi lớn hơn một chút. (Một vector khởi tạo hoặc các bit độn làm đầy đủ cho khối cuối cùng sẽ làm tăng tối đa là một hoặc hai khối). Trong các thuật toán công khai, đầu ra có quan hệ với kích thước đầu vào nhưng không nhất thiết phải tỉ lệ với kích thước này.

5.4. CẢI THIỆN ĐỘ MẬT CỦA HỆ MẬT

DES và RSA được xem là các thuật toán mã hoá an toàn qua nhiều năm nghiên cứu của các nhà thám mã. Các thuật toán Skipjack và ElGamal cũng được coi là an toàn mặc dù chúng chưa được thử thách lâu như DES và RSA. Mặc dù thám mã khó có thể tìm được nội dung của các thông báo được mã bằng các thuật toán trên nhưng việc mô tả những tấn công có khả

năng đối với một hệ mật sẽ chứa đựng nhiều yếu điểm tiềm năng hơn việc phá bí mật của một thông báo. Trong phần này, ta sẽ xem xét một số phương pháp tấn công này.

5.4.1. Ngăn ngừa và phát hiện sai

Trong DES, mỗi khoá làmột thực thể. Một thám mã đã nắm được khuôn dạng của các thông báo của người gửi có thể sửa đổi các thông báo mà không cần phải phá mã.

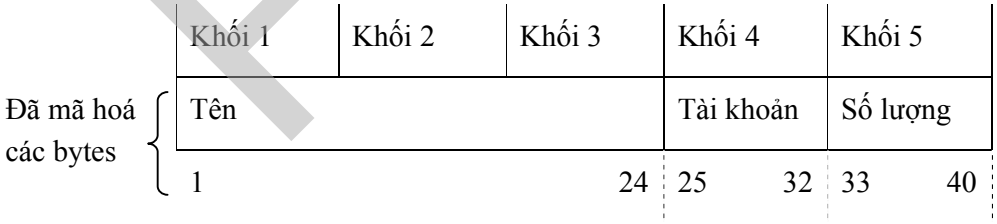
5.4.1.1. Lặp lại khối

Ví dụ, ta xét tình huống sau. Hai ngân hàng thoả thuận trao đổi điện tử các thông tin về chuyển tiền trao đổi giữa hai ngân hàng. Để bảo mật, họ sử dụng các số liệu được mã hoá. Họ thoả thuận phát các bản ghi có khuôn dạng xác định như sau:

Chủ tài khoản	Số tài khoản	Lượng tiền chuyển
24 bytes	8bytes	8 bytes

Các bản ghi này gồm 40 bytes = 320 bits = 5 khối 64 bit (xem hình 5.21).

Giả sử John có thể thu trộm kênh dữ liệu giữa hai ngân hàng. Ngày đầu tiên, John với tư cách chủ tài khoản sẽ chuyển 100 USD từ một ngân hàng này sang một ngân hàng khác. Ngày tiếp theo anh ta cũng làm như vậy. Ở cả hai ngày, anh ta vẫn đều thu chặn và ghi lại quá trình truyền dẫn từ ngân hàng này tới ngân hàng khác. Giả sử rằng cả hai lần truyền dẫn đều được gửi bằng cùng một khoá mã hoá và cả hai lần đều bắt đầu bằng một bản ghi. John biết rằng cả hai lần truyền đều có 3 khối biểu thị tên của mình, một khối biểu thị số tài khoản và một khối biểu thị lượng tiền chuyển. John cũng biết rằng 5 khối này ở cả hai ngày đều như nhau. Việc xác định dữ liệu trong các trường này chỉ là một quá trình tìm kiếm hai phần giống nhau. Công việc tẻ nhạt này có thể thực hiện dễ dàng bằng máy tính.



Hình 5.21.

Ngày tiếp theo, John kiểm tra giả định của mình bằng cách gửi qua một giao dịch khác với một lượng tiền khác. John lại tìm kiếm 4 khối giống với 4 khối trước đó, tuy nhiên các bản ghi này sẽ khác nhau ở khối thứ 5. Nếu chỉ có một tập các khối phù hợp với mẫu này thì John có dạng đã mã tên, số tài khoản của anh ta.

Bằng cách chèn thêm các số liệu vào đường truyền thay cho việc đọc các thông báo từ nó, John bây giờ có thể thay thế một người bất kỳ và số tài khoản của anh ta bằng tên và số tài khoản của mình, riêng số lượng tiền thì không đụng chạm tới. John không cần biết ai là người nhận tiền và lượng tiền được nhận là bao nhiêu, anh ta chỉ đơn giản là thay vào tên vào tên và số tài khoản của mình và quan sát ngân sách trong tài khoản của mình tăng trưởng. Kỹ thuật này được gọi là kỹ thuật lặp lại khối. Trong kỹ thuật này, các khối đã mã từ một lần truyền

dẫn sẽ được gửi lại ở lần truyền dẫn thứ hai. Để sử dụng kỹ thuật này, người thu chặn không cần phải phá mã.

Ở cuối mỗi lần truyền, các ngân hàng thường gửi tổng các chuyển khoản nhưng John sẽ không can thiệp vào, bởi vậy, tổng số sẽ cân bằng. Nếu John may mắn thì ít nhất là một trong những lần truyền đã sửa đổi sẽ làm anh ta giàu to. Các khách hàng có thể không nhận thấy thiếu một lần chuyển (khi John chuyển sang tài khoản của mình) tới một tháng sau khi John bắt đầu can thiệp. John sẽ tiếp tục trò chơi này trong vòng tối đa 1 tháng và rút tiền mặt ra khỏi tài khoản của mình.

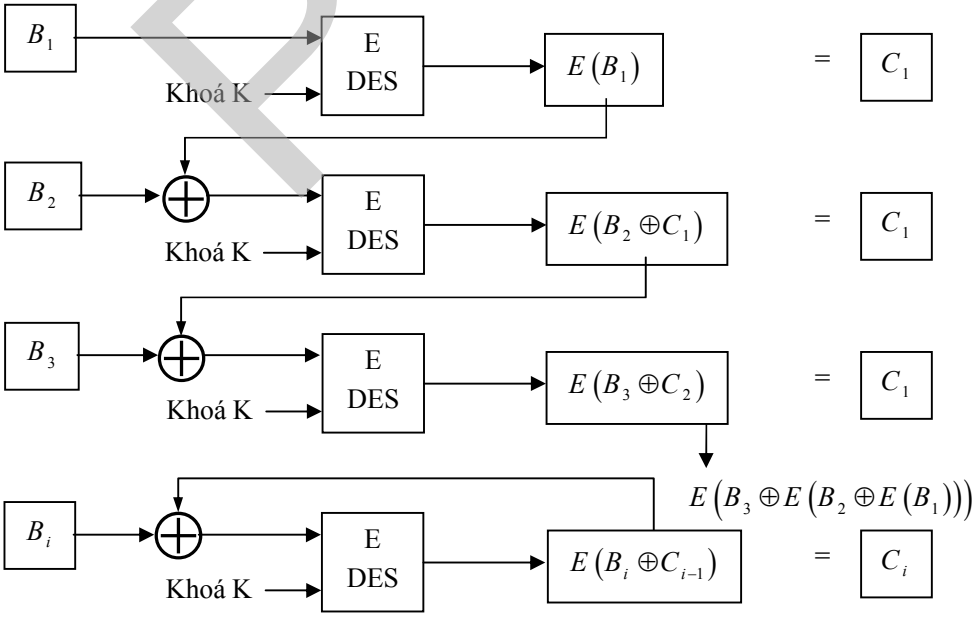
John có thể coi là người trong cuộc theo nghĩa anh ta biết khuôn dạng truyền, tần suất thay đổi khoá, các truyền dẫn này thường xảy ra như thế nào... John cũng biết cách đảm bảo an toàn khi có sự cố. Kỹ thuật được dùng ở đây đã đơn giản hoá đôi chút nhằm làm cho việc giải thích dễ dàng hơn và tránh các hướng dẫn chi tiết cho kẻ xấu lợi dụng.

Do cách xử lý mỗi khối của bản rõ là độc lập nên DES và các mật mã khối khác đều bị ảnh hưởng bởi loại tấn công này. Rất may là có một giải pháp khắc phục dễ dàng, được gọi là giải pháp xích khối.

5.4.1.2. Giải pháp xích khối

Ta biết rằng khi cộng mod 2 một xâu nhị phân với chính nó thì kết quả sẽ bằng 0 và phép "hoặc có loại trừ" này là một phép toán giao hoán. Bởi vậy, với 2 xâu bất kỳ a và b, ta có:

$$\begin{aligned}(a \oplus b) \oplus a &= (a \oplus a) \oplus b \\ &= 0 \oplus b \\ &= b\end{aligned}$$



Hình 5.22. Ví dụ về giải pháp xích khối

Với giải pháp xích khối, mỗi khối cần phát sẽ được kết hợp bằng phép cộng mod 2 với các khối trước. Nếu các khối là B_1, B_2, B_3 và các hàm mã hoá là $E(\ )$ thì các khối sau sẽ được phát:

$$C_1 = E(B_1)$$
$$C_2 = E(E(B_1) \oplus B_2) = E(C_1 + B_2)$$
$$C_3 = E(E(E(B_1) \oplus B_2) \oplus B_3) = E(C_2 + B_3)$$

Quá trình này được mô tả trên hình 5.22.

Bên nhận sẽ giải mã khối đầu tiên nhận được C_1 như thông thường. Bên nhận sẽ giải mã khối thứ hai nhận được (chính là $C_2 = E(C_1 + B_2)$) và thu được $C_1 + B_2$. Từ C_1 đã nhận được trước kia, bên thu sẽ tính $(C_1 \oplus B_2) \oplus C_1$. Ta có:

$$(C_1 \oplus B_2) \oplus C_1 = (C_1 \oplus C_1) \oplus B_2 = 0 \oplus B_2 = B_2$$

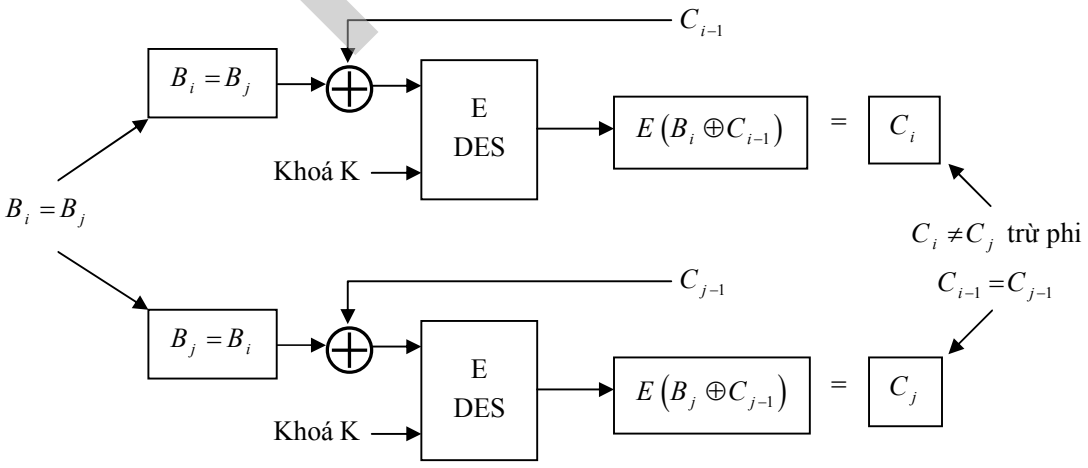
Đây là giá trị rõ của khối thứ hai. Các khối tiếp sau cũng được xử lý tương tự.

Như đã nêu ra trên hình 5.23, ta thấy rằng các khối bản rõ như nhau được phát tách biệt hoặc được phát trong cùng một lần truyền không nhất thiết tạo ra cùng một bản mã. Có sự khác nhau như vậy vì mỗi khối sẽ phụ thuộc vào tất cả các khối ở phần trước. Vì vậy, thám mã không thể lặp lại các khối nhất định từ một đường truyền này sang một lần truyền khác hoặc thậm chí là chỉ tìm các khối xuất phát từ cùng một bản rõ.

Đối với DES, một khối không làm lộ khối khác vì:

$$E(C_1 \oplus B_2) \neq E(C_1) \oplus E(B_2)$$

Thậm chí nếu thám mã biết rằng $C_2 = E(C_1 \oplus B_2)$ và đã nhận được C_1 thì thông tin này cũng không bị để lộ B_2 hoặc thậm chí chỉ là $E(B_2)$.



Hình 5.23. Mã hoá các khối bản rõ như nhau

5.4.1.3. Giá trị ban đầu

Chế độ xích khối sẽ che dấu các khối như nhau. Tuy nhiên, đối với một số thông báo một số khối đầu tiên có thể lại tuân theo một khuôn dạng định trước. Với cùng một khoá mã hoá, tất cả các thông báo dạng này sẽ lại tạo ra cùng một bản mã ở đầu ra.

Nếu điều này không thể chấp nhận được thì ta có thể dùng một giá trị ban đầu để nâng cao độ an toàn. Giá trị này là một xâu ngẫu nhiên nào đó được dùng như khối khởi đầu. Chẳng hạn, xâu này có thể là thời gian, ngày tháng hiện thời hoặc là một khối ngẫu nhiên. Xâu này phải khác nhau đối với mỗi thông báo được mã hoá. Cả bên gửi và bên nhận đều phải giải mã khối này nhưng chỉ dùng nó để giải mã cho các khối tiếp theo (khối này không chứa dữ liệu).

Bằng cách sử dụng một vector khởi tạo, thông báo $B_1B_2B_3\ldots$ sẽ trở thành thông báo $IB_1B_2B_3\ldots$, khối đầu tiên được gửi là $E(I)$, trong đó I là giá trị ban đầu. Khối tiếp theo được gửi là $E\left(E(I)\oplus B_1\right)$. Thậm chí, nếu hai khối giống nhau nằm trong các thông báo khác nhau (ví dụ B_1) thì chúng cũng không tạo ra các bản mã giống nhau (bởi vì I là khác nhau đối với mỗi thông báo nên $E\left(E(I)\oplus B_1\right)$ sẽ luôn khác nhau).

5.4.2. Mã hoá một chiều

Một số phép mã hoá phụ thuộc vào một hàm khó tính toán. Ví dụ, xét hàm lập phương $y=x^3$. Có thể dễ dàng tính x^3 bằng tay. Hàm ngược $\sqrt[3]{y}$ khó tính toán hơn nhiều. Hàm $y=x^2$ không có hàm ngược vì có hai khả năng đối với $\sqrt[2]{y}$ là $+x$ và $-x$. Các hàm như vậy (dễ tính hơn nhiều so với các hàm ngược của chúng) được gọi là các hàm một chiều.

5.4.2.1. Sử dụng mã hoá một chiều

Các hàm một chiều đặc biệt hữu ích trong xác thực. Các mật khẩu thường được dùng để kiểm tra xem một người sử dụng đang cố gắng đăng nhập có phải là một người dùng hợp lệ hay không? Việc đưa ra công khai bảng mật khẩu của các người dùng là nguy hiểm đối với hệ thống máy tính, bởi vậy nhiều hệ thống đã sử dụng hàm một chiều để mã hoá bảng mật khẩu. Hệ thống sẽ lưu $f(pw)$ khi người sử dụng nhận được một khẩu mới pw, trong đó f là hàm một chiều. Khi có một người sử dụng đăng nhập thì hệ thống sẽ hỏi mật khẩu. Người sử dụng đưa vàot và hệ thống sẽ lưu mật khẩu được mã bằng khoá là chính mật khẩu đó.

Hệ thống mật khẩu là an toàn vì không thể không có khả năng tính được f^{-1}. Thám mã phải tìm bảng mật khẩu của người dùng là $f(pw)$. Tuy nhiên, thông tin này không cho phép thám mã suy ra pw. Hơn nữa, thám mã có thể thử các mật khẩu khác nhau cho đến khi tìm thấy một từ w sao cho $f(w)=f(pw)$. Với các mật khẩu có độ dài lớn được chọn từ một bộ ký tự lớn thì kiểu tấn công này được xem là không khả thi.

5.4.2.2. Niêm phong mật mã

Một tính chất hữu dụng khác của mật mã là có khả năng bảo vệ dữ liệu khỏi những can thiệp bất hợp pháp. Ví dụ, trong một file dữ liệu thông thường, một dữ liệu nào đó, một bản ghi nào đó hoặc toàn bộ file có thể bị sửa đổi mà không bị phát hiện. Ở đây, ta chỉ xem xét tới các file mặc dù những tình huống tương tự cũng có thể áp dụng đối với một bản ghi, một trường hoặc một byte riêng lẻ.

Mã hoá thường dùng để bảo mật. Tuy nhiên, trong một số trường hợp, tính toàn vẹn lại là một yếu tố quan trọng hơn. Ví dụ, trong một hệ thống trích xuất dữ liệu, điều quan trọng là phải biết rằng bản sao được trích xuất là đúng với tư liệu đã lưu trữ. Tương tự, trong một hệ thống liên lạc an toàn, nhu cầu truyền dẫn các thông báo đúng có thể còn vượt trên các nhu cầu về bí mật. Mã hoá có thể đảm bảo được tính toàn vẹn cũng như tính bí mật.

Trong hầu hết các file thường không có sự ràng buộc nào giữa các yếu tố, tức là mỗi byte, mỗi bit hay mỗi ký tự là độc lập với nhau trong file. Việc thay đổi một một giá trị sẽ ảnh hưởng tới tính toàn vẹn của file, tuy nhiên điều này rất khó phát hiện được.

Mật mã có thể được dùng để niêm phong một file (giống như bao nó bằng một vỏ chất dẻo) sao cho bất kỳ một sự thay đổi nào đều bị phát hiện. Một trong những kỹ thuật này là tính một hàm mật mã của file (thường được gọi là hàm băm (hask) hay tổng kiểm tra). Trong chương 4, ta đã nghiên cứu mọt số thuật toán băm . Hàm này phải phụ thuộc vào tất cả các bit của file cần niêm phong sao cho bất kỳ một sự thay đổi nào thậm chí chỉ là một bit riêng lẻ cũng làm thay đổi kết quả tổng kiểm tra.

Giá trị tổng kiểm tra được lưu cùng với file. Sau đó, mỗi khi file được sử dụng thì tổng kiểm tra sẽ được tính lại. Nếu tổng kiểm tra tính được phù hợp với giá trị đã lưu thì điều đó có nghĩa là file đã không bị sửa đổi.

Một hàm mật mã (chẳng hạn DES) là đặc biệt thích hợp cho việc niêm phong cho các giá trị vì các thám mã không thể biết phải thay đổi giá trị được lưu như thế nào để phù hợp với dữ liệu cần sửa đổi. Như đã mô tả ở trên, chế độ xích khối của DES sẽ tạo ra một đầu ra mà mỗi khối sẽ phụ thuộc vào giá trị của các khối trước. Tổng kiểm tra mật mã của file có thể là khối cuối cùng trong phép mã hoá DES ở chế độ xích khối vì rằng khối này sẽ phụ thuộc vào tất cả các khối khác.

5.4.2.3. Xác thực

Ta nhớ lại một câu chuyện cổ kể về hai người cắt một đồng tiền làm đôi, mỗi người giữ một nửa và rồi ra đi. Anh ta nói rằng: "Nếu có một người đưa tin mang đến cho bạn một cái gì đó kèm theo nửa đồng tiền này thì bạn sẽ biết rằng đó là của tôi". Với mã hoá, ta có thể làm tương tự. Trong giao tiếp cá nhân, khi nói về một người mà chúng ta biết, ta có nhiều cách để đảm bảo rằng người mà chúng ta nói chính là người mà ta đã biết. Ta ghi nhận giọng nói, phong cách, các mẫu ngữ điệu. Trong giao tiếp máy tính, ta không có nhiều chứng cứ để đảm bảo cho ta việc nhận dạng.

Phương thức xác thực chủ yếu là một mật khẩu: đó là một từ hoặc một xâu mà chỉ có một người biết. Hệ thống máy tính sẽ coi rằng một người nào đó biết mật khẩu sẽ chính là

người có mật khẩu đó. Như ta sẽ thấy sau này, các mật khẩu có thể coi là an toàn, mặc dù nếu được sử dụng không đúng thì chúng chỉ bảo vệ được chút ít.

Mã hoá là một dạng khác của xác thực. Nếu bạn nhận được một thông báo đã mã hoá mà bạn có thể giải mã bằng khoá chỉ có bạn và người khác biết thì thông báo là xác thực. Trừ phi sơ đồ mã hoá đã bị khoá hoặc mã đã bị lộ, còn không thì thông báo chỉ có thể được tạo bởi người có khoá.

Khẳng định này có thể mở rộng cho toàn bộ thông báo. Với một thuật toán mã hoá mạnh thì không ai có thể thay thế một câu nào đó trong thông báo và cũng không thể dán các mẩu thông báo cũ với nhau. Chẳng hạn, chế độ xích khối mật mã đã xét ở trên không cho phép thay bất kỳ một khối nào trong thông báo. Thông báo đã nhận là được gửi bởi một người đã biết và không thể bị sửa đổi trước khi thu nó.

5.4.2.4. Các nhãn thời gian

Một vấn đề an toàn khác cần đặt ra là có thể có một thông báo đã bị thu chặn và bây giờ đang được phát lại. (Tình huống này tương tự như việc đổi tiền bằng một bản sao của séc mà người gửi không biết đó là một bản sao). Trong tình huống này, người gửi và người nhận có thể gắn với mỗi thông báo một nhãn thời gian hoặc một số thông báo.

Số thông báo là một con số được gắn vào một thông báo. Thám mã không có cách nào để biết được các bit của số này nằm ở vị trí nào trong thông báo hoặc không thể biết cách thay đổi các bit để tạo ra dạng mã hoá của số tiếp sau hoặc không thể biết cách thay đổi các bit này mà không làm gián đoạn việc giải mã phần còn lại của thông báo. Các số thông báo này không thể bị thay thế, thay đổi hoặc giả mạo. Người nhận phải duy trì việc đếm các số thông báo đã nhận được. Nếu hai người sử dụng một tập các số thì người nhận có thể ngay lập tức biết được liệu có thông báo nào trước thông báo hiện thời đã bị mất hoặc bị chậm trễ vì số được mã hoá của thông báo hiện thời phải lớn hơn số được mã hoá của thông báo trước.

Nếu người gửi có nhiều thông báo thì có thể gặp phải vấn đề số thông báo quá dài. Vì lý do này, người ta thường đặt lại bộ đếm số thông báo khi nó đạt tới giá trị quá lớn (thường là sau khi độ dài của số thông báo vượt quá 30 bit). Trong trường hợp này, tất cả các bên thu phải được thông báo rằng số thông báo được gửi tiếp theo sẽ được đặt lại về một số nhỏ (chẳng hạn là 0).

Còn một vấn đề khác là với mỗi cặp người gửi - người nhận phải có một bộ đếm riêng. Thông thường, người gửi hoặc người nhận phải duy trì nhiều bộ đếm khác nhau cho nhiều đối tượng khác nhau.

Có một vấn đề nữa là một người gửi sẽ có một bộ tạo số thông báo duy nhất cho các thông báo tới mọi đối tượng. Nếu người gửi gửi thông báo cho hai người A và B thì thông báo 1 có thể đi tới A, thông báo 2 tới B, các thông báo 3 và 4 tới A, thông báo 5 tới B...

Không một bên thu nào có thể biết được thông báo nào đã bị mất vì các số thông báo chỉ là một dãy tăng chứ không nhất thiết phải là tất cả các số trong một dải nào đó. Tuy nhiên, bất cứ lúc nào, bên thu cũng có thể yêu cầu phát lại thông báo trước. Số thông báo cần phải được mã hoá hoặc phải được bảo vệ bằng một cách nào đó để ngăn chặn việc sửa đổi.

Các nhãn thời gian và ngày tháng có thể xem là một cách mềm dẻo hơn đôi chút. Chúng là các dấu hiệu về thời gian và ngày tháng mà thông báo được gửi với mức chính xác đủ để cho không có hai thông báo nào có cùng một dấu hiệu. Các nhãn này không phải đặt lại.

Bên thu phải đồng bộ về thời gian rất chặt chẽ với bên gửi. Khi truyền nhanh, nếu các đồng bộ của bên thu và bên phát không đồng bộ thì nhãn thời gian của người gửi có thể chậm hơn thời gian hiện thời của bên nhận. Có thể cho phép có một sai lệch nhỏ đối với thời gian đồng bộ hoặc phải ghi nhận rằng các nhãn thời gian của bên gửi luôn nhanh hơn một chút.

Bây giờ, chúng ta sẽ nghiên cứu các khía cạnh khác nhau của DES trong các ứng dụng riêng trong việc đảm bảo tính toàn vẹn, tránh sử dụng lại và đảm bảo bí mật. Các ứng dụng này có thể thu được từ nhiều hệ thống mã khoá khác bởi vì chúng chỉ phụ thuộc vào những tính chất chung của phép mã hoá.

5.5. CÁC CHẾ ĐỘ MÃ HOÁ

Bây giờ chúng ta nghiên cứu một số phương pháp khác nhau mà phép mã hoá có thể sử dụng. Các nguyên tắc chung ở đây có thể áp dụng cho hầu hết các thuật toán mã hoá thông thường (kể cả đối xứng và không đối xứng). Một dạng sử dụng DES đã được trình bày trong chương 3, được gọi là chế độ quyển mã điện tử (ECB). Trong những phần sau ta sẽ trình bày các chế độ sử dụng DES khác.

5.5.1. Chế độ xích khối mật mã (CBC)

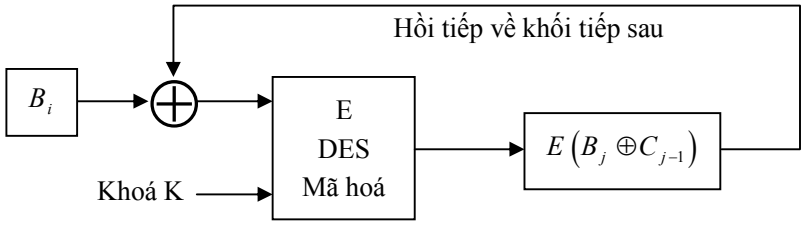
Như đã nêu ở trên, chế độ CBC bắt đầu bằng một vector khởi tạo ngẫu nhiên (IV), vector này được mã hoá và được gửi như khối 1. Sau đó, vector này sẽ được cộng mod 2 với khối bản rõ đầu tiên $\left(E\left(p_1 \oplus IV, k\right)\right)$. Kết quả này sẽ được mã làm khối 2. Sau đó, khối bản mã thứ hai này sẽ được cộng mod 2 với khối bản rõ tiếp theo trong thông báo nhằm không thể thay thế khối bản mã bằng một khối khác mà không bị phát hiện. Chế độ CBC đã được chỉ ra trên hình 5.24.

Chế độ CBC cũng có tính chất tự sửa sao cho một sự thay đổi trong khối c_i chỉ ảnh hưởng tới việc giảm mã cho các khối p_i và p_{i+1}. Tuy nhiên, các khối p_{i+2} và tiếp theo sẽ không bị tác động. Sau hai khối hàm cộng mod 2 sẽ huỷ bất kỳ một sai nào. (Cần nhớ rằng, đối với một khâu bất kỳ x, $x \oplus x = 0$). Tính chất tự sửa là thuận tiện vì một sai số trong truyền dẫn hoặc mã hoá sẽ không phá huỷ một lượng lớn bản mã.

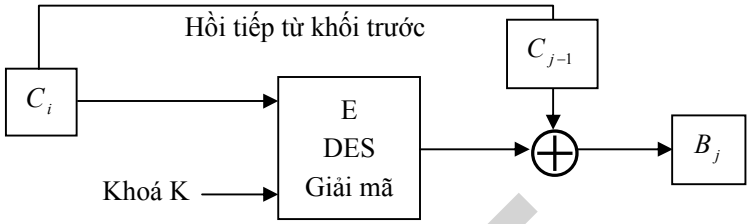
5.5.2. Chế độ hồi tiếp mật mã (CFB)

Bản chất định hướng theo khối của DES là không thuận tiện với 2 lý do sau.

- Thứ nhất là khối cuối cùng phải được độn thêm. Bởi vậy, kích thước của bản mã kết quả có thể lớn hơn một chút so với bản rõ tương ứng.
- Thứ hai là không thể thực hiện việc mã hoá nếu chưa nhận đủ 64 bit của một khối. Bởi vậy, tốc độ mã hoá cho một ký tự có thể phụ thuộc vào tốc độ nhận của một vài ký tự tiếp sau. Có một số ứng dụng đòi hỏi phải mã hoá ngay từng ký tự. Ví dụ, trong một môi trường mạng an toàn, người dùng phải phát mỗi ký tự như nó được đưa vào từ đầu cuối.



a) Mã hoá



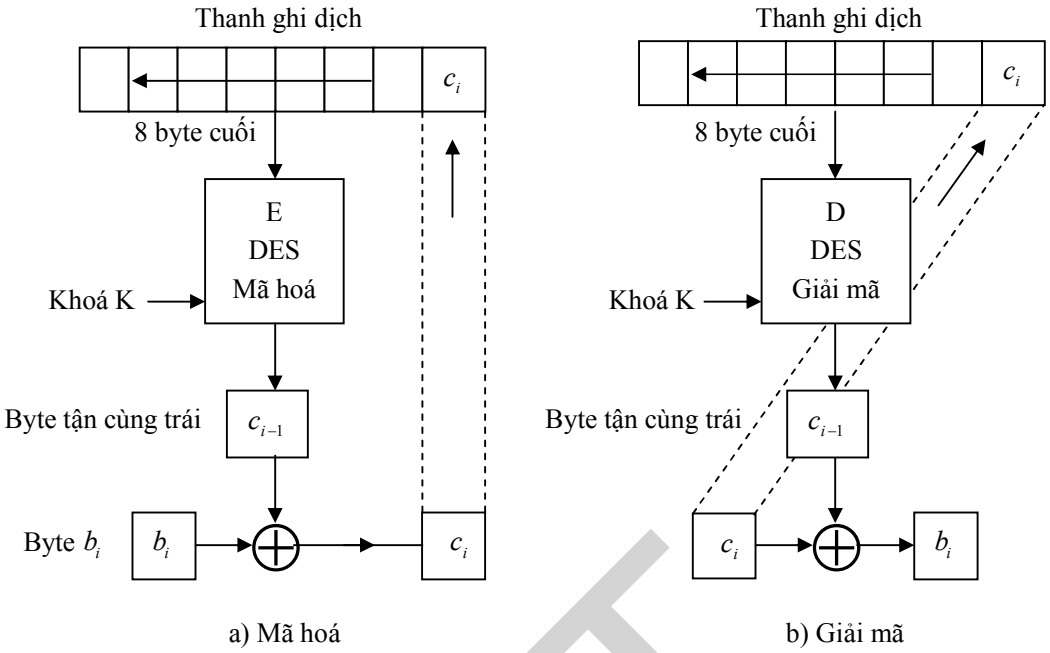
b) Giải mã

Hình 5.24. Chế độ CBC

Chế độ CFB hoạt động trên một hàng đợi 64 bit. Ban đầu, hàng đợi được làm đầy bằng vector khởi tạo (giống như vector IV trong chế độ CBC). Hàng đợi này sẽ được mã hoá và 8 bit tận cùng bên trái của kết quả sẽ được cộng mod 2 với ký tự đầu tiên cần mã hoá. 8 bit được mã hoá này sẽ được phát đi. 8 bit này cũng được chuyển vào vị trí 8 bit tận cùng bên phải của hàng đợi. Tất cả các bit khác sẽ được chuyển sang trái 8 bit và 8 bit tận cùng bên trái sẽ bị loại bỏ. Quá trình tương tự cũng được thực hiện ở đầu thu. Thủ tục này được chỉ ra trên hình 5.25.

Ưu điểm của chế độ CFB là có thể mã hoá ngay lập tức một ký tự. Cũng như trong chế độ CBC, mỗi ký tự cũng ảnh hưởng tới tất cả các ký tự tiếp sau sao cho một sự thay đổi tới một ký tự bất kỳ trong quá trình truyền cũng tác động tới các ký tự kế tiếp. Một sai trong quá trình mã hoá sẽ tác động tới ký tự đang được chuyển và cũng tác động tới 8 ký tự tiếp theo bởi vì nó sẽ nằm trong hàng đợi ở 8 phép mã hoá ký tự sau đó cho tới khi ký tự sai được đẩy ra khỏi ô tận cùng bên trái trong hàng đợi.

Hai chế độ này mở rộng cơ bản tính hiệu quả của DES. Chúng khắc phục được 2 hạn chế chủ yếu của DES: hạn chế khi mã hoá các bản rõ lặp lại và sự giữ chậm để đợi mã hoá đồng thời một khối văn bản. Không có những hạn chế này, DES có chất lượng tương đương với các thuật toán mã hoá dòng.



Hình 5.25. Chế độ CFB

5.5.3. Hai khoá cho hiệu quả tương đương một khoá 112 bit

Mặc dù độ dài khoá của DES được coi là đủ dài nhưng một số người vẫn không tin tưởng vào độ dài 56 bit của nó. Sau đây là một phương pháp nhằm tăng độ dài hiệu quả của khoá. Phương pháp này không đòi hỏi phải thay đổi bản thân thuật toán, nó thích hợp trong trường hợp thuật toán được áp dụng bởi thiết bị phần cứng hoặc trong trường hợp phần mềm có những đoạn không thể sửa đổi.

Do đã có những khảo sát về độ mật của một khoá 56 bit nên hợp lý hơn cả là nên xem xét việc sử dụng 2 khoá. Nếu bằng một cách nào đó, một phép tìm kiếm vét cạn đã tìm được một khoá thì cần phải thời gian gấp đôi để phá khoá thứ hai. Đáng tiếc là điều này không đơn giản như vậy. Merkle [14] đã chứng tỏ rằng hai khoá 56 bit dùng liên tiếp có thể bị phá bằng phép tấn công theo bản rõ chọn lọc trong 2^{57} phép thử chứ không phải là trong 2^{112} phép thử như mong muốn. Bởi vậy, phép mã hoá lần hai hầu như không làm tăng độ mật.

Tuchman [15] tính rằng hai khoá được dùng theo một cách đặc biệt sẽ làm tăng độ mật. Tuchman sử dụng một kỹ thuật được gọi là DES bội ba (Triple DES) do Matyas và Mayer đưa ra. TDES được IBM sử dụng khi mã hoá các khoá chủ trong một số hệ thống mã hoá của họ. Với hai khoá (K_1 và K_2), người gửi sẽ mã hoá bằng K_1, giải mã bằng K_2 và lại mã hoá bằng K_1. Bên thu sẽ giải mã bằng K_1, mã hoá bằng K_2 và lại giải mã bằng K_1.

Quan điểm này rất đáng xem xét trong sử dụng với một thiết bị mã hoá tự động (có thể bằng phần cứng hoặc phần mềm). Nếu thiết bị cần hai khoá và người dùng chỉ muốn dùng một khoá thì người dùng sẽ cung cấp K_1 ba lần. Thiết bị sẽ mã bằng K_1, giải mã bằng K_1 (trở về bản rõ ban đầu) và cuối cùng lại mã hoá bằng K_1. Theo cách này, một thiết bị có thể dùng cho cả các phép mã hoá đơn và kép.

5.6. TÓM LƯỢC VỀ CÁC THỦ TỤC VÀ CÁC ỨNG DỤNG THỰC TẾ

Trong chương này, chúng ta đã xem xét một số vấn đề quan trọng trong sử dụng các hệ thống liên lạc và các hệ thống máy tính. Ta đã nghiên cứu các thủ tục mà có thể tách biệt một cách có hiệu quả giữa thiết kế của một giải pháp với ứng dụng của giải pháp đó.

Chúng ta cũng đã xác định những nhiệm vụ đối với chúng, các thủ tục phải được thiết kế. Các nhiệm vụ này có ứng dụng trong thương mại điện tử, liên lạc cá nhân và những hoạt động thông thường khác mà ta muốn tự động thực hiện. Các nhiệm vụ này là:

- Ký bằng chữ ký số
- Chơi bài bằng tâm linh (phân phối khoá)
- Giao kèo khoá
- Bỏ phiếu
- Chuyển không nhớ (tung đồng tiền)
- Ký thoả thuận
- Thư tín có chứng thực

Các thủ tục đã giải quyết những vấn đề thường đòi hỏi khi giao tiếp mặt mặt giữa con người. Tuy nhiên, bằng cách thiết kế cẩn thận, ta đã có thể đảm bảo được tính bí mật và xác thực trong khi vẫn giữ được tuyệt đối trung thực.

Như đã nói ở trên, việc tách biệt thiết kế ra khỏi ứng dụng là điều đáng mong muốn. Các thủ tục (có thể phân tích và kiểm tra được) là một phương tiện thiết kế tốt bởi vì tính đúng đắn của một thủ tục có thể được xác định không liên quan tới bất kỳ một ứng dụng riêng nào của nó.

Đối với mỗi thủ tục, trước tiên ta khai thác được vấn đề cần giải quyết. Sau đó, ta xác định những yêu cầu bảo mật cần phải thoả mãn với thủ tục. Tiếp theo, ta thiết kế một thủ tục thoả mãn được các yêu cầu bảo mật. Cuối cùng, chúng ta thực hiện việc phân tích thủ tục để khẳng định rằng nó đã thoả mãn được các yêu cầu.

Chúng ta cũng xem xét những ứng dụng thực tế thích hợp khi sử dụng mã hoá. Chỉ sử dụng mã hoá thì không đủ đảm bảo được độ mật, tính riêng tư hoặc xác thực. Tuy vậy, bằng cách sử dụng đúng phép mã hoá sẽ mang lại cho ta những kết quả đó. Ta cũng đánh giá các hệ thống mã hoá DES và RSA theo các tiêu chuẩn kinh điển của Shannon và các tiêu chuẩn bảo mật khác. Cuối cùng, ta nghiên cứu các kỹ thuật mật mã nhằm hạn chế các sai sót không bị phát hiện (chẳng hạn như sử dụng lại) thông qua các kỹ thuật như nhãn thời gian...

BÀI TẬP CHƯƠNG 5

Bài 5.1: Tại sao người ta không muốn có trọng tài trong một thủ tục trao đổi bí mật?

Bài 5.2: Cho một ví dụ về một thủ tục tự ràng buộc trong đời sống thực tế?

Bài 5.3: Hàm niêm phong mật mã trước tiên được đề xuất (Tổng giá trị số của tất cả các byte trong một thông báo) có một yếu điểm nghiêm trọng: việc trao đổi vị trí của hai byte trong thông báo sẽ không được phát hiện bởi hàm này. Hãy đề xuất một hàm khác không có yếu điểm này.

Bài 5.4: Phép mã hoá xếp ba lô của Merkler Hellman không phải là một ánh xạ "vào", tức là có một số nhị phân nào đó không phải là kết quả áp dụng phép mã hoá xếp ba lô lên một đoạn bản rõ. Với những thủ tục nào thì đặc tính này sẽ gây ra vấn đề? Hãy giải thích?

Bài 5.5: Có một yếu điểm trong cơ chế niêm phong mật mã được nêu ở đây. Nếu R có thể tính f_S để kiểm tra xem tư liệu M đã nhận được có phải là tư liệu đã phát hay không thì R cũng có thể tính f_S để giả mạo một chữ ký số. Hãy đề xuất một giải pháp cho vấn đề này.

Bài 5.6: Hãy mô tả một thủ tục trao đổi các bí mật một cách công bằng theo cách thiết lập của con người (không phải theo kiểu máy tính). Hai người muốn trao đổi thông tin bí mật nhưng không một ai muốn đưa một bí mật mà không nhận được một bí mật từ phía bên kia.

a) Các yêu cầu bí mật của tình huống này là gì?

b) Thế nào là một thủ tục trao đổi bí mật công bằng?

Bài 5.7: Hãy liệt kê các yêu cầu của một sơ đồ chữ ký số khoá bí mật. Có thể có một yêu cầu nào trong các yêu cầu này cần thoả mãn với một thủ tục có trọng tài? Tại sao có, tại sao không? Có thể có một yêu cầu này cần thoả mãn trong một thủ tục ràng buộc? Tại sao có, tại sao không?

Bài 5.8: Hãy trình bày một thủ tục chữ ký số sử dụng mã hoá đối xứng sao cho người gửi và người nhận không phải để lộ nội dung thông báo của họ cho trọng tài.

Bài 5.9: Hãy giải thích tạo sao thủ tục chữ ký số dùng mã hoá khoá công khai sẽ cản trở bên thu giả mạo thông báo từ người gửi bằng cách sử dụng khoá công khai của người gửi.

Bài 5.10: Hãy chỉ ra rằng hệ mật RSA có cả hai tính chất sau: vừa là ánh xạ giao hoán, vừa là ánh xạ "vào". Tại sao hai tính chất này lại cần thiết đối với một thủ tục chữ ký số khoá công khai?

Bài 5.11: Một hàm băm (hash) sẽ rút gọn một khối số liệu (lớn) thành một tóm lược (nhỏ hơn). Điều này có nghĩa là không phải mọi sửa đổi có thể có đối với số liệu gốc đều được phát hiện bởi hàm băm. Tuy nhiên, một thuật toán nén văn bản cũng rút gọn số liệu lớn thành dạng được nén nhỏ hơn. Liệu một thuật toán nén có khả năng phát hiện được tất cả những thay đổi trong văn bản gốc? Tại sao có, tại sao không? Liệu một thuật toán nén văn bản có thể sử dụng như một hàm băm hay không? Tại sao có, tại sao không?

CHƯƠNG 6. CÁC CHUẨN VÀ ÁP DỤNG

6.1. BẢO MẬT THƯ ĐIỆN TỬ SỬ DỤNG PRETTY GOOD PRIVACY (PGP)

6.1.1. Mở đầu

PGP là một hiện tượng nổi bật. PGP chủ yếu được kiến tạo bởi Phil Zimmerman. PGP cung cấp dịch vụ xác thực và bí mật có thể dùng cho E-mail và cho việc lưu trữ file. Về cơ bản Phil Zimmerman đã làm được các công việc sau:

Chọn được các thuật toán mật mã tốt để tạo dựng các modun.

Tích hợp các thuật toán này vào một ứng dụng độc lập với hệ điều hành và bộ xử lý. Ứng dụng này sử dụng một tập nhỏ các lệnh dễ dùng.

- Tạo gói phần mềm với đầy đủ tài liệu kể cả mã nguồn cung cấp miễn phí trên mạng Internet.
- Thỏa thuận với một công ty (ViaCrypt - Hiện nay là Network Associates) để cung cấp phiên bản thương mại với giá rẻ.

PGP đã phát triển mạnh mẽ và hiện nay đang được sử dụng rộng rãi (được xem như một chuẩn bảo mật E-mail cho cộng đồng Internet).

Một số lý do thúc đẩy sự phát triển của PGP:

- Có nhiều phiên bản miễn phí có thể chạy được trên nhiều nền như: DOS/Windows, UNIX, Macintosh… Ngoài ra còn có một sản phẩm thương mại có sự trợ giúp đầy đủ cho khách hàng.
- PGP dựa trên các thuật toán đã được xem xét kỹ và được coi là cực mật như:
  + Các thuật toán mã hóa công khai RSA, DSS, Diffie-Hellman.
  + CAST-128, IDEA, TDEA cho mật mã đối xứng.
  + Hàm băm SHA-1
- Có lực lượng đông đảo những người dùng từ các công ty tới các cá nhân muốn giao tiếp an toàn với các người khác bên ngoài thế giới qua Internet và qua các mạng khác.
- Không bị khống chế và ràng buộc bởi bất kỳ một chính phủ nào hoặc bởi bất cứ một cơ quan tiêu chuẩn nào.

6.1.2. Ký hiệu

- Ks : Khóa phiên được dùng cho mã hóa đối xứng.
- KRa : Khóa riêng của user A được dùng trong sơ đồ mã hóa công khai.
- KUa : Khóa công khai của user A được dùng trong sơ đồ mã hóa công khai.

- EP : Mã hóa công khai.
- DP : Giải mã khóa công khai.
- EC : Mã hóa khóa đối xứng.
- DC : Giải mã khóa đối xứng.
- H : Hàm băm.
- || : Phép ghép.
- Z : Nén dùng thuật toán ZIP.
- R64 : Biến đổi sang khuôn dạng ASCII cơ số 64.

Chú ý:

+ Thuật ngữ khóa bí mật là khóa cùng cặp với khóa công khai trong mật mã hóa công khai .
+ Thuật ngữ khóa riêng là khóa của mật mã đối xứng.

6.1.3. Mô tả hoạt động

Không kể tới việc quản lý khóa hoạt động của PGP gồm 5 dịch vụ: Xác thực, bí mật, nén, tương thích E-mail và phân đoạn.

Bảng 6.1: Tóm lược các dịch vụ của PGP

Chức năng	Các thuật toán được dùng	Mô tả
Chữ ký số	DSS/SHA hoặc RSA/SHA	Mã băm của thông báo được tạo bằng cách dùng SHS-L. Tóm lược thông báo này được mã bằng DSS hoặc RSA với khóa riêng của người gửi và được kèm với thông báo
Mã hóa thông báo	CAST hoặc IDEA hoặc TDEA với 3 khóa với Diffie-Hellman hoặc RSA	Thông báo được mã bằng CAST-128/IDEA/ 3DES với khóa phiên một lần được tạo bởi người gửi. Khóa phiên được mã bằng Diffie-Hellman hoặc RSA với khóa công khai của bên thu và được gửi kèm với thông báo.
Nén	ZIP	Thông báo có thể được nén (để lưu giữ hoặc để truyền) bằng ZIP
Tương thích E-mail	Biến đổi cơ số 64	Để cung cấp tính trong suốt đối với các ứng dụng thư điện tử, thông báo đã mã có thể được biến đổi thành 1 xâu ASCII sử dụng phép biến đổi cơ số 64
Phân đoạn		Để phù hợp với hạn chế về kích thước lớn nhất của thông báo, PGP sẽ thực hiện việc phân đoạn và sắp xếp lại.

6.1.3.1. Xác thực

- Người gửi tạo một thông báo.
- Thuật toán SHA-1 được dùng để tạo một mã băm 160 bít của thông báo.
- Mã băm được mã hóa bằng RSA sử dụng khóa riêng của người gửi và kết quả được gắn vào thông báo.
- Bên thu sử dụng RSA với khóa công khai của người gửi để giải mã và khôi phục lại mã băm.
- Bên thu tạo một mã băm mới do thông báo và so sánh nó với mã băm đã được giải mã. Nếu hai mã này phù hợp thì thông báo được coi là xác thực.

Việc kết hợp SHA-1 và RSA tạo nền một sơ đồ chữ ký có hiệu quả. Nhờ độ mật của RSA bên thu được đảm bảo rằng chỉ có người chủ của khóa bí mật tương ứng mới có thể tạo ra chữ ký.

Nhờ thuật toán băm SHA-1 bên thu cũng đảm bảo rằng không một ai có thể tạo được một thông báo mới mà thông báo đó lại có cùng mã băm. Còn một tùy chọn khác là sử dụng DSS/SHA-1.

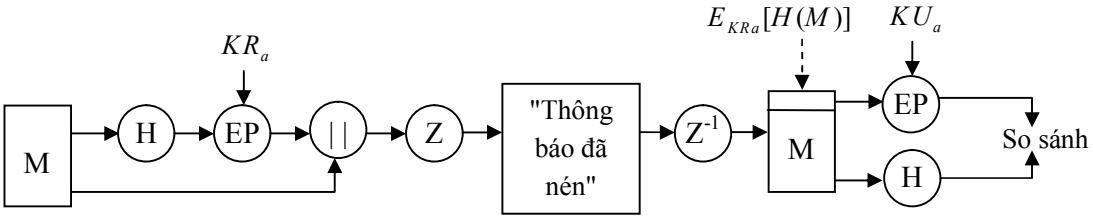
6.1.3.2. Bí mật

Bí mật là một dịch vụ khác của PGP. Nó thực hiện mã hóa các thông báo cần phát hoặc cần được lưu giữ tại chỗ như một file. Trong cả hai trường hợp có thể dùng một trong các mã pháp CAST-128, IDEA hoặc TDEA. Chế độ hồi tiếp khối mã 64 bít được sử dụng .

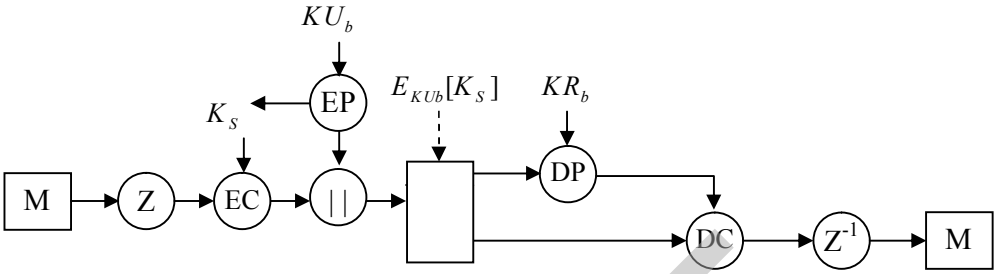
Trong PGP mỗi khóa riêng chỉ được dùng một lần. Tức là một khóa mới sẽ được tạo ra như một số ngẫu nhiên 128 bít. Vì chỉ được dùng một lần nên khóa phiên được gắn vào thông báo và được truyền cùng thông báo. Để bảo vệ khóa, nó được mã hóa bằng khóa công khai của người nhận.

Quá trình này được mô tả như sau:

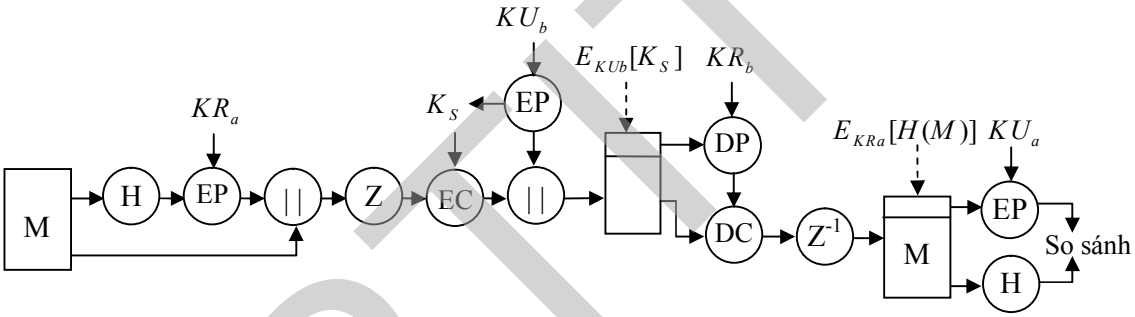
- Người gửi tạo một thông báo và một số ngẫu nhiên 128 bít được dùng làm khóa phiên cho chính thông báo này.
- Thông báo được mã hóa bằng CAST-128 (hoặcIDEA/ TDEA) với khóa phiên này.
- Khóa phiên được mã bằng RSA nhờ dùng khóa công khai của người nhận và được gắn vào thông báo.
- Bên thu dùng RSA với khóa bí mật của mình để giải mã và khôi phục lại khóa phiên.
- Khóa phiên được dùng để giải mã thông báo.



a) Xác thực



b) Bí mật



c) Bí mật và xác thực

Hình 6.1. Các chức năng của PGP

6.1.3.3. Bí mật và xác thực

Có thể sử dụng cả hai dịch vụ này cho cùng một thông báo. Trước tiên một chữ ký được tạo ra đối với một thông báo rõ và được gắn vào thông báo. Sau đó thông báo rõ và chữ ký sẽ được mã bằng CAST-128 (hoặcIDEA/ TDEA) và khóa phiên được mã bằng RSA (hoặc ElGamal) theo khóa công khai của người nhận.

6.1.3.4. Nén

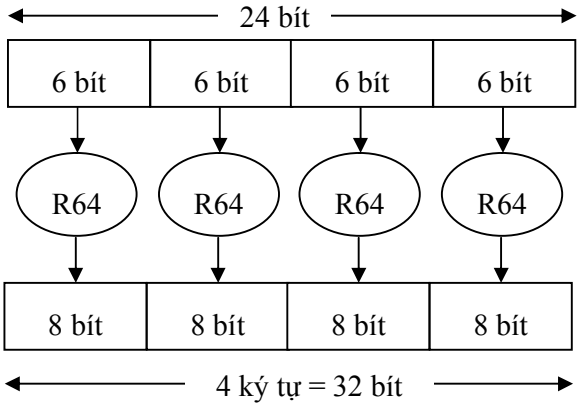
Theo ngầm định, PGP sẽ nén thông báo sau khi ký nhưng trước khi mã. Điều này nhằm tiết kiệm bộ nhớ để truyền thư điện tử hoặc để lưu trữ file. Ngoài ra, điều này còn thuận tiện cho khi cần kiểm tra.

Chú ý: Z: nén, Z^{-1} : Giải nén.

6.1.3.5. Tương thích E-mail.

Nhiều chương trình E-mail chỉ cho phép dùng các khối chứa các văn bản ASCII Nhằm phù hợp với hạn chế này PGP có cung cấp dịch vụ biến đổi dòng bít chứa các dãy 8 bít tùy ý

thành dòng bít gồm các ký tự ASCII. Sơ đồ được dùng cho phép biến đổi này là sơ đồ biến đổi cơ số 64, mỗi nhóm 3 bytes dữ liệu nhị phân được biến đổi thành 4 ký tự ASCII.



Hình 6.2.

Bảng 6.2: Mã hóa cơ số 64

Giá trị 6 bít	Mã hoá ký tự	Giá trị 6 bít	Mã hoá ký tự	Giá trị 6 bít	Mã hoá ký tự	Giá trị 6 bít	Mã hoá ký tự
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

Sử dụng phép biến đổi cơ số 64 này sẽ làm mở rộng thông báo chừng 33%. Tuy nhiên do phần khóa phiên và chữ ký không lớn, mặt khác thông báo rõ đã được nén. Trên thực tế quá trình nén đã dư sức bù cho phần mở rộng này.

Ví dụ: Khi dùng ZIP, tỷ số nén vào khoảng 2 lần. Nếu ta bỏ qua các thành phần khóa và chữ ký tương đối nhỏ thì hiệu quả chung của cả nén và mở rộng đối với 1 file có độ dài X là:

$$1,33.0,5.X = 0,665.X .$$

Như vậy file kết quả vẫn được nén 33% (1/3).

6.1.3.6. Phân đoạn và sắp xếp lại.

Các chương trình E-mail thường hạn chế độ dài lớn nhất của thông báo (Chẳng hạn 50.000 bytes).

Một thông báo dài hơn phải được phân đoạn thành các đoạn nhỏ hơn, mỗi đoạn phải được gửi tách biệt.

Để phù hợp với hạn chế này PGP sẽ tự động chia một thông báo quá lớn thành các đoạn đủ nhỏ để gửi qua E-mail.

Việc phân đoạn được thực hiện sau tất cả quá trình khác (bao gồm cả phép biến đổi cơ số 64). Bởi vậy thành phần khóa phiên và thành phần chữ ký chỉ xuất hiện một lần ở đầu của đoạn đầu trên.

Ở phía thu, PGP phải tách ra tất cả các header của E-mail và lắp ráp lại toàn bộ khối ban đầu trước khi thực hiện các công đoạn còn lại.

6.2. GIAO DỊCH ĐIỆN TỬ AN TOÀN - SET

6.2.1. Mở đầu

SET là một đặc tả về mã hóa và an toàn được xây dựng nhằm bảo vệ các giao dịch bằng thẻ tín dụng trên Internet.

Phiên bản hiện thời SETv1 được phát triển với sự tham gia của các công ty IBM, Microsoft, Netscape, RSA, Terisa và Verisign. Tới năm 1998, các phần mềm tương thích với SET đã xuất hiện. Bản thân SET không phải là một hệ thống chi trả. Nó là một tập các thủ tục an toàn và các khuôn dạng an toàn cung cấp cho những người sử dụng hạ tầng chi trả bằng thẻ tín dụng hiện thời trên mạng mở (Internet) theo một cách an toàn.

Về cơ bản SET cung cấp 3 dịch vụ:

- Cung cấp một kênh liên lạc an toàn giữa các bên thực hiện giao dịch.
- Cung cấp xác thực nhờ sử dụng các chứng chỉ số X509v.3.
- Đảm bảo tính riêng tư vì thông tin chỉ khả dụng đối với các bên vào lúc cần thiết và ở nơi cần thiết.

SET được mô tả trong 3 cuốn sách được xuất bản năm 1997:

- Q1: Mô tả thương mại (80 trang).
- Q2: Hướng dẫn cho lập trình viên (629 trang).
- Q3: Định nghĩa thủ tục hình thức (262 trang).

6.2.2. Mô tả SET

6.2.2.1. Các yêu cầu thương mại

Q1 mô tả các yêu cầu thương mại cho quá trình chi trả an toàn và các thẻ tín dụng trên Internet và các mạng khác.

- *Cung cấp sự bí mật cho các thông tin đặt hàng và chi trả.*

Điều cần thiết là phải đảm bảo cho người sở hữu thẻ tín dụng rằng thông tin này là an toàn và chỉ có bên nhận hợp pháp mới có thể truy nhập được. Tính bí mật này cũng làm giảm nguy cơ gian trá của bên thực hiện giao dịch khác hoặc của bên thứ ba ác ý. SET sử dụng mã hóa để cung cấp khả năng này.

- *Đảm bảo tính toàn vẹn của tất cả các dự liệu đã phát*

Tức là phải đảm bảo rằng không thể xảy ra bất cứ một sự thay đổi nào trong nội dung khi truyền các thông báo SET.

Ở đây chữ ký số được dùng để cung cấp tính toàn vẹn.

- *Cung cấp sự xác thực rằng chủ sở hữu thẻ tín dụng (card)* là người sử dụng hợp pháp của tài khoản.

Một cơ chế kết nối chủ sở hữu thẻ tới một số tài khoản riêng làm giảm khả năng gian lận.

Các chữ ký số và các chứng chỉ được dùng để kiểm tra rằng chủ thẻ là người dùng hợp pháp của một tài khoản hợp lệ.

- *Cung cấp sự xác thực rằng người bán (nhà buôn) có thể chấp nhận các giao dịch* dùng thẻ qua mối quan hệ của nó với một cơ quan tài chính. Chủ thẻ phải có khả năng xác định các nhà buôn mà họ thực hiện các giao dịch.

Lại một lần nữa ở đây SET sử dụng các chữ ký số và các chứng chỉ.

- *Đảm bảo sử dụng các kỹ thuật thiết kế hệ thống tốt nhất và các thực tế an toàn tốt nhất* để bảo vệ các bên hợp pháp tham gia giao dịch thương mại điện tử.

SET là một đặc tả đã được kiểm tra kỹ dựa trên các thuật toán và các thủ tục mật mã an toàn cao.

- *Tạo một thủ tục và các khuôn dạng của SET* là độc lập với phần cứng, hệ điều hành và phần mềm Web

6.2.2.2. Các đặc điểm chủ chốt của SET

- *Tính bí mật của thông tin*

Tài khoản của chủ thẻ và thông tin chi trả phải được bí mật. Một điểm quan trọng và thú vị của SET là nó tránh cho nhà buôn khỏi phải biết số thẻ tín dụng, số này chỉ dùng cho ngân hàng.

DES được dùng để đảm bảo tính bí mật

- *Tính toàn vẹn của dữ liệu*

Thông tin chi trả được gửi từ chủ thẻ tới nhà buôn bao gồm: thông tin đặt hàng, số liệu cá nhân và các lệnh chi trả.

SET đảm bảo rằng các nội dung thông báo này không bị biến đổi trong khi truyền. Các chữ ký số RSA sử dụng các mã băm SHA-1 cung cấp tính toàn vẹn cho thông báo.

Một số thông báo cũng được bảo vệ bằng HMAC dùng SHA-1.

- *Xác thực tài khoản của chủ thẻ.*

SET cho phép nhà buôn kiểm tra rằng chủ thẻ có phải là người dùng hợp pháp không và số tài khoản có hợp lệ không.

SET dùng các chứng chỉ số X509v3 với các chữ ký số RSA cho mục đích này.

- *Xác thực nhà buôn SET cung cấp cho chủ thẻ* có thể kiểm tra được rằng nhà buôn có quan hệ với cơ quan tài chính cho phép chấp nhận các thẻ chi trả.

6.2.2.3. Các bên tham gia giao dịch trong SET

- *Chủ thẻ tín dụng* .

Trong môi trường điện tử các khách hàng sẽ giao tiếp với các nhà buôn từ máy tính cá nhân trên Internet. Chủ thẻ tín dụng là chủ nhân hợp pháp của thẻ chi trả (chẳng hạn Master card và Visa card)

- *Nhà buôn:* là một người hoặc một tổ chức có hàng hóa hoặc các dịch vụ để bán cho các chủ thẻ tín dụng. Các hàng hóa và các dịch vụ này được mời chào trên trang Web hoặc qua thư điện tử.

Một nhà buôn chấp nhận các thẻ chi trả phải có mối quan hệ với một ngân hàng (kho bạc).

- *Nhà phát hành:* Đây là một cơ quan tài chính (chẳng hạn ngân hàng) cung cấp thẻ tín dụng. Nhà phát hành phải chịu trách nhiệm đối với việc chi trả các khoản của nhà buôn.

- *Cổng chi trả:* Đây là chức năng được điều hành bởi kho bạc. Cổng chi trả sẽ thực hiện giao tiếp giữa SET và các mạng chi trả. Nhà buôn sẽ trao đổi các thông báo SET với cổng chi trả qua Internet, còn cổng chi trả sẽ có một số kết nối mạng hoặc kết nối trực tiếp tới hệ thống xử lý tài chính của kho bạc.

- *Máy chủ chứng thực:* Đây là một thực thể tin cậy đưa ra các chứng chỉ khóa công khai cho các chủ thẻ tín dụng, cho các nhà buôn và cho các cổng chi trả. Sự thành công của SET sẽ tùy thuộc vào sự tồn tại của hạ tầng máy chủ chứng thực này.

Bây giờ chúng ta sẽ mô tả một cách ngắn gọn dãy <u>các biến cố cần có cho một giao dịch:</u>

- *Khách hàng mở một tài khoản:* Khách hàng sẽ nhận một tài khoản cho thẻ tín dụng.

- *Khách hàng nhận một chứng chỉ:* Sau phép kiểm tra về tính danh, khách hàng sẽ nhận một chứng chỉ số được ký bởi ngân hàng. Chứng chỉ sẽ kiểm tra khóa công khai RSA của khách hàng và ngày hết hạn của nó. Nó cũng thiết lập một quan hệ (được đảm bảo bởi ngân hàng) giữa cặp khóa của ngân hàng và thẻ tín dụng của anh ta.

- *Các nhà buôn có các chứng chỉ của riêng họ:* Một nhà buôn chấp nhận một loại card chi trả nhất định phải có hai chứng chỉ cho hai khóa công khai mà nhà buôn có: Một để ký nhận các thông báo và một để trao đổi khóa. Nhà buôn cần có một đảm bảo chứng chỉ khóa công khai của cổng chi trả.

- *Khách hàng đặt một đơn hàng:* Trước tiên khách hàng vào trang Web của nhà buôn để chọn hàng và xác định giá. Sau đó khách hàng sẽ gửi một danh sách các hạng mục cần mua

tới nhà buôn. Nhà buôn sẽ gửi trở lại một đơn hàng chứa danh sách các mặt hàng, giá của chúng, giá tổng cộng và số của đơn hàng.

- *Nhà buôn được kiểm tra thêm vào đơn hàng :* Nhà buôn sẽ gửi một bản sao chứng chỉ của nó để khách hàng có thể kiểm tra được là anh ta làm việc với một kho hàng hợp pháp.

- *Đơn hàng chi trả được gửi:* Khách hàng sẽ gửi cả thông tin về đơn hàng và thông tin chi trả tới nhà buôn cùng với chứng chỉ của mình. Đơn hàng sẽ xác nhận việc mua các mặt hàng trong đơn hàng, thông tin chi trả được mã hóa theo cách mà nhà buôn không thể đọc được. Chứng chỉ của khách hàng sẽ làm cho nhà buôn có thể kiểm tra được khách hàng.

- *Nhà buôn yêu cầu quyền chi trả:* Nhà buôn gửi thông tin chi trả tới cổng chi trả đòi hỏi quyền nhận một khoản tín dụng thích hợp của khách hàng đủ cho vụ mua bán đó..

- *Nhà buôn xác nhận đơn hàng:* Nhà buôn sẽ gửi xác nhận về đơn hàng cho khách hàng.

- *Nhà buôn cung cấp hàng hóa và dịch vu:* Nhà buôn vận chuyển hàng hóa hoặc cung cấp dịch vụ cho khách hàng.

- *Nhà buôn đòi hỏi chi trả:* Đòi hỏi này sẽ được gửi đến cổng chi trả là nơi khống chế toàn bộ quá trình chi trả.

6.2.2.4. Chữ ký kép

Mục đích của chữ ký kép là liên kết 2 thông báo gửi cho hai bên nhận khác nhau. Trong trường hợp này khách hàng muốn gửi thông tin đặt hàng (OI) tới nhà buôn và thông tin chi trả (PI) tới ngân hàng.

Nhà buôn không cần biết số thẻ tín dụng của khách hàng và ngân hàng không cần biết chi tiết đơn hàng của khách hàng.

Khách hàng cần giữ hai thông tin này tách biệt nhưng hai mục này phải kết nối theo cách để có thể được dùng để giải quyết các tranh chấp khi cần. Kết nối này cần thiết để khách hàng có thể chứng tỏ rằng chi trả này là cho đơn hàng đó chứ không phải cho các hàng hóa hoặc dịch vụ khác. Để thấy rõ hơn ta phải giả sử rằng: khách hàng phải cho nhà buôn hai thông báo: OI có ký và PI có ký. Và nhà buôn gửi PI tới ngân hàng. Nếu nhà buôn có giữ một OI khác của khách hàng thì nhà buôn có thể tuyên bố rằng đó là OI đi kèm với PI này chứ không phải là OI thực sự. Mối kết nối sẽ tránh được sự nhập nhèm đó.

$$DS = E_{KR_C}\left[H\left(H\left(PI\right)//H\left(OI\right)\right)\right]$$

KR_C : Khóa chữ ký riêng của khách hàng.

Bây giờ giả sử rằng nhà buôn có chữ ký kép (PS), có OI và tóm lược thông báo của PI (PIMD). Nhà buôn cũng có khóa công khai của khách hàng lấy từ chứng chỉ của khách hàng. Sau đó nhà buôn có thể tính hai đại lượng sau:

$$H\left(PIM//H\left(OI\right)\right)$$

và $$D_{KU_C}\left(DS\right)$$

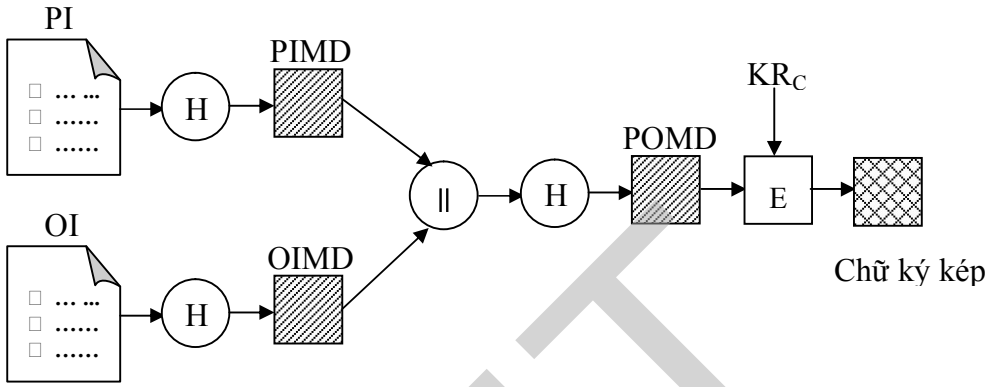
KU_C : Khóa chữ ký công khai của khách hàng.

Nếu hai đại lượng này bằng nhau thì nhà buôn đã kiểm tra được chữ ký.

Tương tự, nếu ngân hàng có $\begin{cases} DS, PI, OIMD \\ KU_C \end{cases}$

Khi đó ngân hàng có thể tính: $H(H(PI)//OIMD)$ và $D_{KU_C}(DS)$.

Nếu hai đại lượng này bằng nhau thì ngân hàng đã kiểm tra được chữ ký.



Hình 6.3. Cấu trúc của chữ ký kép

PI : Thông tin chi trả
PIMD : Tóm lược thông báo của PI
OI : Thông tin đặt hàng
OIMD : Tóm lược thông báo của OI
H : Hàm băm
POMD: Tóm lược thông báo của lệnh chi trả
|| : Phép ghép
E : Phép mã hóa (RSA)
KR_C : Khóa chữ ký riêng của khách hàng.

6.3. ỨNG DỤNG XÁC THỰC - KERBEROS

6.3.1. Mở đầu

Kerberos là một dịch vụ xác thực được xây dựng từ dự án Athena của MIT. Vấn đề mà Kerberos muốn giải quyết là: Trong một môi trường phân tán mở, ở đó các trạm làm việc muốn truy nhập các dịch vụ trên các máy chủ phân tán qua mạng. Ta muốn các máy chủ có khả năng hạn chế truy nhập đối với các người dùng hợp lệ và có thể xác thực các yêu cầu đối với mọi dịch vụ. Trong môi trường này một trạm làm việc không thể tự xác định được đúng các người dùng của nó cho các dịch vụ mạng.

Trên thực tế có ba nguy cơ sau:

- Người sử dụng có thể sử dụng một trạm làm việc nào đó và giả mạo là một người dùng khác.
- Người sử dụng (user) có thể thay đổi địa chỉ mạng của một trạm làm việc để các yêu cầu được gửi từ trạm này xuất hiện như thể từ trạm mạo danh.
- User có thể "nghe trộm" các trao đổi và sử dụng kiểu tấn công sử dụng lại để truy nhập vào một máy chủ hoặc phá vỡ hoạt động.

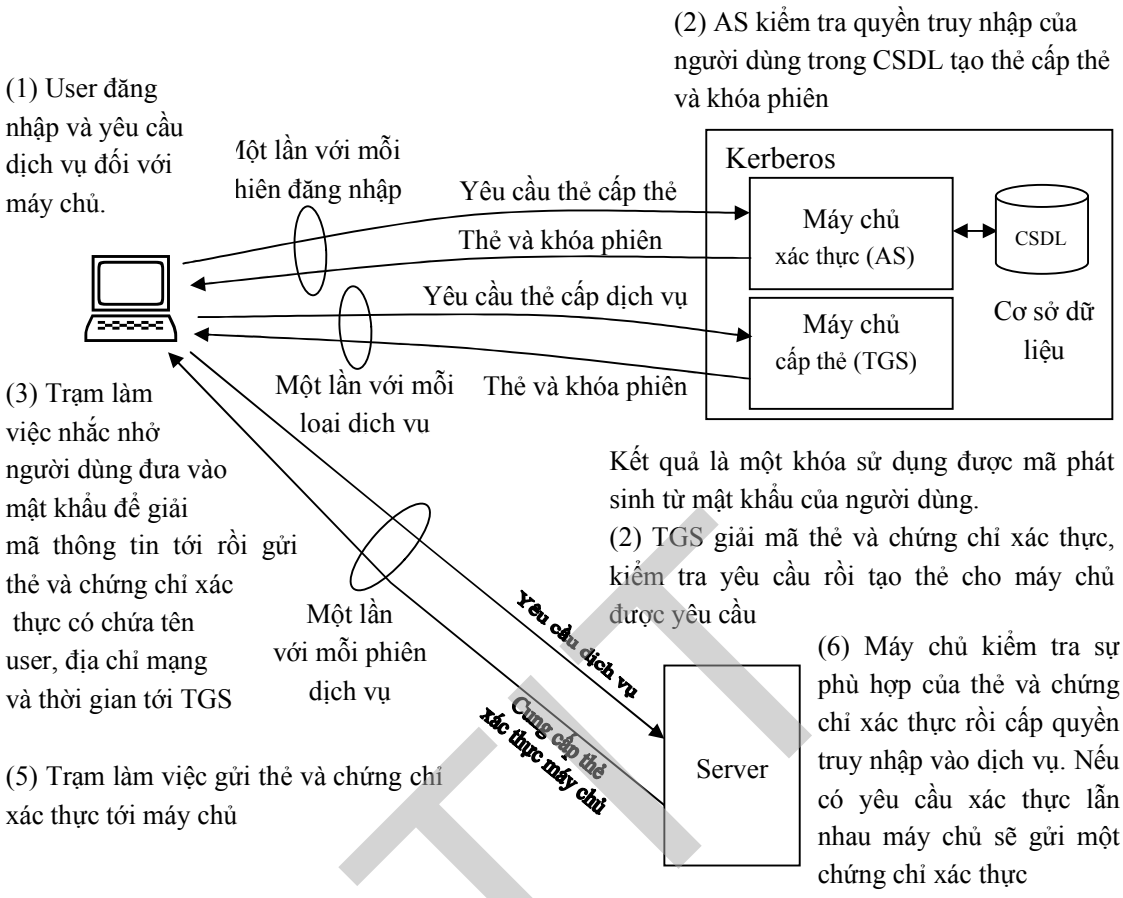
Trong các trường hợp này người dùng bất hợp pháp có thể truy nhập được tới các dịch vụ và dữ liệu mà anh ta không được phép truy nhập. Khác với việc xây dựng các thủ tục xác thực ở mỗi máy chủ, Kerberos cung cấp một dịch vụ xác thực tập trung có nhiệm vụ xác thực các user đối với máy chủ (server) và ngược lại. Cần chú ý rằng Kerberos chỉ sử dụng mật mã đối xứng mà không dùng mật mã khóa công khai.

Có hai phiên bản Kerberos là Kv.4 và Kv.5. Kv.5 được đệ trình xem như một chuẩn được khuyến nghị cho Internet.

Các yêu cầu chính đặt ra cho Kerberos:

- An toàn : Kẻ thu trộm trên mạng không có khả năng thu được thông tin cần thiết để mạo danh. Hơn nữa, Kerberos phải đủ mạnh để đối phương mạnh không thể tìm thấy được các yếu điểm.
- Tin cậy: Đối với mọi dịch vụ khống chế truy nhập có sử dụng Kerberos, việc thiếu tính sẵn sàng của dịch vụ Kerberos cũng có nghĩa là thiếu tính sẵn sàng của các dịch vụ được trợ giúp. Bởi vậy Kerberos phải rất tin cậy và phải sử dụng kiến trúc máy chủ phân tán có hệ thống dự phòng.
- Trong suốt: Về mặt lý tưởng khi người dùng đưa ra mật khẩu vào họ không thể biết được rằng quá trình xác thực đang xảy ra.
- Có khả năng mở rộng: Kerberos phải có khả năng phục vụ một số lượng lớn các máy chủ và máy trạm. Bởi vậy Kerberos phải có kiến trúc phân tán và mođun.

6.3.2. Kerberos V.4



Hình 6.4. Tóm lược các trao đổi thông báo của Kv.4

a) Trao đổi dịch vụ xác thực: Để thu nhận thẻ cấp thẻ.

(1) $C \rightarrow AS : ID_C \,//\, ID_{tgs} \,//\, TS_1$ (Nhãn thời gian).

(2) AS → C: E_{Kc} [$K_{C,\,tgs}$ // ID_{tgs} // TS_2 // LT_2 (thời gian sống) // Thẻ $_{tgs}$]

Thẻ $_{tgs} = E_{Ktgs}$ [$K_{C,\,tgs}$ // ID_C // AD_C // ID_{tgs} // TS_2 // LT_2]

b) Trao đổi dịch vụ cấp thẻ: Để thu nhận thẻ cung cấp dịch vụ.

(3) C → TGS : ID_V Thẻ $_{tgs}$ // Authenticator $_C$. (ID_V : ID của dịch vụ yêu cầu)

(4) TGS → C : $E_{K,tgs}$ [$K_{C,\,V}$ // ID_V // TS_2 // Thẻ $_V$]

Thẻ $_{tgs} = E_{Ktgs}$ [$K_{C,\,tgs}$ // ID_C // AD_C // ID_{tgs} // TS_2 // LT_2]

Thẻ$_V$ $= E_{Kv}$ [$K_{C,\,v}$ // ID_C // AD_C // ID_V // TS_2 // LT_4]

Authenticator $_C$ $= E_{K,tgs}$ [ID_C // AD_C // TS_3]

c) Trao đổi xác thực Máy trạm/ máy chủ: Để thu nhận dịch vụ.

(5) $C \rightarrow V$: Thẻ $_V$ // Authenticator $_C$.

(6) $V \rightarrow C$: $E_{Kc,v}$ [TS_5 H] (để xác thực lẫn nhau)

Thẻ$_V$ = E_{Kv} [$K_{C,v}$ // ID_C // AD_C // ID_V // TS_2 // LT_4]

Authenticator $_C$ = $E_{K,tgs}$ [ID_C // AD_C // TS_5]

Bảng trên cho ta thấy kỹ thuật phân phối khoá phiên.

Trước hết khách hàng (client) gửi một thông báo tới AS yêu cầu truy nhập vào TGS. AS trả lời bằng một thông báo được mã bằng khóa trích xuất từ mật khẩu của người dùng (K_C) có chứa thẻ. Thông báo này cũng chứa một bản sao của khoá phiên $K_{C,tgs}$ cho C và TGS, vì khóa phiên này nằm bên trong thông báo được mã bằng K_C nên chỉ có máy trạm của người dùng mới có thể đọc được nó. Khóa phiên này cũng nằm trong Thẻ $_{tgs}$ mà TGS có thể đọc được. Như vậy khóa phiên đã được phân phối an toàn cho C và TGS. Cần chú ý rằng một số thông tin phụ đã được thêm vào pha hội thoại đầu tiên. Thông báo (1) chứa nhãn thời gian TS_1 để AS biết rằng thông báo là đúng lúc. Thông báo (2) chứa một số yếu tố của thẻ ở dạng mà C có thể truy nhập. Điều này cho phép C xác nhận rằng thư này là của TGS và biết được thời gian hết hạn của nó.

Khi có thẻ và khóa phiên C đã có thể truy nhập vào TGS.

Trước hết C gửi tới TGS một thông báo chứa thẻ + ID_V (định danh của dịch vụ yêu cầu) (thông báo (3)). Hơn nữa, C cũng phát một thẻ xác thực Authenticator bao gồm ID_C và AD_C (địa chỉ của người dùng C) và một nhãn thời gian TS_3. Không giống như thẻ có thể dùng lại, thẻ xác thực chỉ dùng một lần và có thời gian sống ngắn. TGS có thể giải mã thẻ bằng khóa mà nó chia sẻ với AS. Thẻ này báo rằng người dùng C đã được cung cấp khóa phiên $K_{c,tgs}$. Thực ra, thẻ này báo rằng "Người sử dụng khóa $K_{c,tgs}$ phải là C ".

TGS dùng khóa phiên để giải mã thẻ xác thực . Sau đó TGS có thể kiểm tra tên và địa chỉ từ thẻ xác thực từ nội dung của thẻ và từ địa chỉ mạng của thông báo tới. Nếu mọi điều là phù hợp thì TGS dã được đảm bảo rằng người gửi thẻ thực sự là chủ nhân của thẻ. Thực ra thẻ này báo rằng "ở thời điểm TS_3 tôi sử dụng $K_{c,tgs}$". Cần chú ý rằng thẻ không chứng minh định danh của bất cứ ai nhưng là một phương pháp để phân phối các khóa một cách an toàn. Chính thẻ xác thực sẽ chứng minh định danh của client. Vì thẻ xác thực chỉ dùng một lần và có thời gian dùng ngắn nên đối phương khó lòng có thể ăn cắp thẻ và thẻ xác thực để dùng lại.

TGS trả lời bằng thông báo (4) có dạng như thông báo (2).

Thông báo được mã bằng khóa phiên chia sẻ giữa TGS và C, nó chứa một khóa phiên cần chia sẻ giữa C và máy chủ V, định danh của V; ID_V, nhãn thời gian của thẻ. Bản thân thẻ cũng chứa khóa phiên này.

Lúc này C có thẻ cấp dịch vụ có thể dùng lại đối với V. Khi C trình thẻ này trong thông báo (5) nó cũng gửi kèm theo một thẻ xác thực (Authentication)

Máy chủ V có thể giải mã thẻ, khôi phục khóa phiên và giải mã thẻ xác thực khi cần xác thực lẫn nhau, máy chủ có thể trả lời như thông báo (6). Máy chủ trả lại giá trị nhãn thời gian lấy từ thẻ xác thực được tăng thêm 1 và được mã bằng khóa phiên. C có thể giải mã thông báo này để khôi phục lại nhãn thời gian đã được tăng. Vì thông báo được mã bằng khóa phiên nên C được đảm bảo rằng nó chỉ có thể được tạo bởi V. Nội dung của thông báo đảm bảo với C rằng thông báo này không phải là dùng lại thông báo cũ. Cuối cùng, ở cuối quá trình, client và server cùng chia sẻ một khóa bí mật. Khóa này có thể được dùng để mã hóa các thông báo trong tương lai cho hai bên hoặc để trao đổi khóa phiên ngẫu nhiên mới.

Giải thích các yếu tố trong thủ tục Kv.4.

	a) Trao đổi dịch vụ xác thực
Thông báo (1)	Client yêu cầu thẻ cấp thẻ.
ID_C	Báo cho AS định danh của người dùng từ client này
ID_{tgs}	Báo cho AS biết rằng người dùng yêu cầu truy nhập TGS
TS_1	Cho phép AS kiểm tra rằng đồng hồ của client được đồng bộ với AS
Thông báo (2)	AS trả về thẻ cấp thẻ
E_{K_C}	Mã hóa dựa trên mật khẩu của user, cho phép AS và client kiểm tra mật khẩu và bảo vệ nội dung của thông báo (2)
$K_{C,tgs}$	Bản sao của khóa phiên được tạo bởi AS mà client có thể sử dụng để thực hiện trao đổi bí mật giữa client và TGS (không yêu cầu chúng phải chia sẻ một khóa cố định)
ID_{tgs}	Xác nhận rằng thẻ này là cho TGS
TS_2	Báo cho client về thời gian mà thẻ này đệ trình.
LT_2	Báo cho client về thời gian sống của thẻ này
Thẻ $_{tgs}$	Thẻ cần được client sử dụng để truy nhập TGS
	b) Trao đổi dịch vụ cấp thẻ
Thông báo (3)	Client yêu cầu thẻ cấp dịch vụ
ID_V	Báo cho TGS rằng user yêu cầu truy nhập tới máy chủ V.
Thẻ $_{tgs}$	Đảm bảo cho TGS rằng user này được xác thực bởi AS.
$Authenticator_C$	Được tạo bởi client để xác nhận tính hợp lệ cho thẻ.
Thông báo (4)	TGS trả về thẻ cấp dịch vụ.
$E_{K_C,tgs}$	Khóa được chia sẻ giữa C và TGS dùng bảo vệ nội dung của thông báo (4)
$K_{C,tgs}$	Bản sao của khoá phiên được tạo bởi AS.
ID_V	Xác nhận rằng thẻ này là cho máy chủ V
TS_4	Báo cho client về thời gian mà thẻ này được đệ trình.

Thẻ $_V$	Thẻ được dùng bởi client để truy nhập vào máy chủ V.
Thẻ $_{tgs}$	Có khả năng dùng lại để người dùng không phải vào lại mật khẩu.
E_{Ktgs}	Thẻ được mã hóa bằng khóa chỉ có AS và TGS biết nhằm tránh phá rối (thu trộm)
$K_{C,tgs}$	Bản sao của khóa phiên mà TGS có thể truy nhập được dùng để giải mã thẻ xác thực nhờ đó xác thực thẻ
ID_C	Báo chủ sở hữu hợp lệ của thẻ này
AD_C	Tránh việc dùng thẻ từ một trạm làm việc khác với trạm đã yêu cầu thẻ
ID_{tgs}	Đảm bảo cho máy chủ rằng nó đã giải mã đúng cho thẻ.
TS_2	Báo cho TGS về thời gian mà thẻ này được đệ trình.
LT_2	Tránh dùng lại sau khi thẻ đã hết hạn.
$Authenticator_C$	Đảm bảo cho TGS rằng người trình thẻ đúng là client mà thẻ đã được trình cho nó, có thời gian sống ngắn để tránh dùng lại.
$E_{K_C,tgs}$	Thẻ xác thực được mã bằng khóa chỉ có client và TGS biết nhằm tránh thu trộm.
ID_C	Phải phù hợp với ID trong thẻ để xác thực thẻ.
AD_C	Phải phù hợp với địa chỉ trong thẻ để xác thực thẻ
TS_2	Báo cho TGS về thời gian mà thẻ xác thực này được tạo ra.
	c) Trao đổi xác thực Máy trạm/ máy chủ (Client/Server)
Thông báo (5)	Client yêu cầu dịch vụ
Thẻ $_V$	Đảm bảo với máy chủ rằng người dùng này đã được AS xác nhận
$Authenticator_C$	Được tạo bởi client để xác nhận tính hợp lệ cho thẻ.
Thông báo (6)	Xác thực (tùy chọn) của server đối với client
$E_{K_C,v}$	Đảm bảo cho C rằng thông báo này là từ V
TS_5+1	Đảm bảo cho C rằng đây không phải là sự dùng lại của hồi đáp cũ
Thẻ $_V$	Có thể dùng lại để client không cần yêu cầu thẻ mới từ TGS đối với mỗi truy nhập tới cùng máy chủ.
E_{Kv}	Thẻ được mã hóa bằng khóa chỉ có TGS và server biết nhằm tránh thu trộm.
$K_{C,v}$	Bản sao của khóa phiên mà client có thể truy nhập, được dùng để giải mã thẻ xác thực nhằm xác thực thẻ
ID_C	Báo chủ sở hữu hợp lệ của thẻ này
AD_C	Tránh việc dùng thẻ từ một trạm làm việc khác với trạm đã yêu cầu thẻ.
ID_v	Đảm bảo cho server rằng nó đã giải mã đúng cho thẻ.
TS_4	Báo cho server về thời gian mà thẻ này được đệ trình.
LT_4	Tránh dùng lại sau khi thẻ đã hết hạn.

$Authenticator_C$	Đảm bảo cho server rằng người trình thẻ đúng là client mà thẻ đã được đưa cho nó, có thời gian sống ngắn để tránh dùng lại.
$E_{K_C,v}$	Thẻ xác thực được mã bằng khóa chỉ có server và client biết nhằm tránh thu trộm.
ID_C	Phải phù hợp với ID trong thẻ để xác thực thẻ.
AD_C	Phải phù hợp với địa chỉ trong thẻ để xác thực thẻ
TS_5	Báo cho server về thời gian mà thẻ xác thực này được tạo.

BÀI TẬP CHƯƠNG 6

Bài 6.1: Nêu ý tưởng thiết kế một hệ mật bảo vệ các file dữ liệu trên cơ sở phân tích PGP?

Bài 6.2: Cấu tạo của chữ ký kép? Ý nghĩa của chữ ký kép trong thương mại điện tử?

Bài 6.3: Vai trò của máy chủ xác thực trong Kenberos?

Bài 6.4: Hãy mô tả quá trình trao đổi dịch vụ xác thực trong K.V.4?

PHỤ LỤC 1 - MÃ NGUỒN DES

```
#define EN0   0      /* MODE == encrypt */
#define DE1   1      /* MODE == decrypt */

typedef struct {
        unsigned long ek[32];
        unsigned long dk[32];
} des_ctx;

extern void deskey(unsigned char *, short);
/*                  hexkey[8]     MODE
 * Sets the internal key register according to the hexadecimal
 * key contained in the 8 bytes of hexkey, according to the DES,
 * for encryption or decryption according to MODE.
 */

extern void usekey(unsigned long *);
/*                cookedkey[32]
 * Loads the internal key register with the data in cookedkey.
 */

extern void cpkey(unsigned long *);
/*               cookedkey[32]
 * Copies the contents of the internal key register into the storage
 * located at &cookedkey[0].
 */

extern void des(unsigned char *, unsigned char *);
/*                from[8]          to[8]
 * Encrypts/Decrypts (according to the key currently loaded in the
 * internal key register) one block of eight bytes at address `from'
 * into the block at address `to'.  They can be the same.
 */

static void scrunch(unsigned char *, unsigned long *);
static void unscrun(unsigned long *, unsigned char *);
static void desfunc(unsigned long *, unsigned long *);
static void cookey(unsigned long *);

static unsigned long KnL[32] = { 0L };
static unsigned long KnR[32] = { 0L };
static unsigned long Kn3[32] = { 0L };
static unsigned char Df_Key[24] = {
       0i01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef,
       0xfe,0xdc,0xba,0x98,0x76,0x54,0x32,0x10,
       0x89,0xab,0xcd,0xef,0i01,0x23,0x45,0x67 };

static unsigned short bytebit[8]    = {
       0200, 0100, 040, 020, 010, 04, 02, 01 };

static unsigned long bigbyte[24] = {
       0x800000L,     0x400000L,     0x200000L,     0x100000L,
       0x80000L,      0x40000L,      0x20000L,      0x10000L,
       0x8000L,       0x4000L,       0x2000L,       0x1000L,
       0x800L,        0x400L,        0x200L,        0x100L,
```

```
        0x80L,          0x40L,          0x20L,          0x10L,
        0x8L,           0x4L,           0x2L,           0x1L    };

/* Use the key schedule specified in the Standard (ANSI X3.92-1981). */

static unsigned char pc1[56] = {
        56, 48, 40, 32, 24, 16,  8,   0, 57, 49, 41, 33, 25, 17,
         9,  1, 58, 50, 42, 34, 26,  18, 10,  2, 59, 51, 43, 35,
        62, 54, 46, 38, 30, 22, 14,   6, 61, 53, 45, 37, 29, 21,
        13,  5, 60, 52, 44, 36, 28,  20, 12,  4, 27, 19, 11,  3 };

static unsigned char totrot[16] = {
        1,2,4,6,8,10,12,14,15,17,19,21,23,25,27,28 };

static unsigned char pc2[48] = {
        13, 16, 10, 23,  0,  4,       2, 27, 14,  5, 20,  9,
        22, 18, 11,  3, 25,  7,      15,  6, 26, 19, 12,  1,
        40, 51, 30, 36, 46, 54,      29, 39, 50, 44, 32, 47,
        43, 48, 38, 55, 33, 52,      45, 41, 49, 35, 28, 31 };
void deskey(key, edf)       /* Thanks to James Gillogly & Phil Karn! */
unsigned char *key;
short edf;
{
        register int i, j, l, m, n;
        unsigned char pc1m[56], pcr[56];
        unsigned long kn[32];

        for ( j = 0; j < 56; j++ ) {
                l = pc1[j];
                m = l & 07;
                pc1m[j] = (key[l >> 3] & bytebit[m]) ? 1 : 0;
                }
        for( i = 0; i < 16; i++ ) {
                if( edf == DE1 ) m = (15 - i) << 1;
                else m = i << 1;
                n = m + 1;
                kn[m] = kn[n] = 0L;
                for( j = 0; j < 28; j++ ) {
                        l = j + totrot[i];
                        if( l < 28 ) pcr[j] = pc1m[l];
                        else pcr[j] = pc1m[l - 28];
                        }
                for( j = 28; j < 56; j++ ) {
                    l = j + totrot[i];
                    if( l < 56 ) pcr[j] = pc1m[l];
                    else pcr[j] = pc1m[l - 28];
                    }
                for( j = 0; j < 24; j++ ) {
                        if( pcr[pc2[j]] ) kn[m] |= bigbyte[j];
                        if( pcr[pc2[j+24]] ) kn[n] |= bigbyte[j];
                        }
                }
        cookey(kn);
        return;
}

static void cookey(raw1)
register unsigned long *raw1;
{
        register unsigned long *cook, *raw0;
        unsigned long dough[32];
```

```
        register int i;

        cook = dough;
        for( i = 0; i < 16; i++, raw1++ ) {
                raw0 = raw1++;
                *cook   = (*raw0 & 0x00fc0000L) << 6;
                *cook  |= (*raw0 & 0x00000fc0L) << 10;
                *cook  |= (*raw1 & 0x00fc0000L) >> 10;
                *cook++        |= (*raw1 & 0i00000fc0L) >> 6;
                *cook   = (*raw0 & 0x0003f000L) << 12;
                *cook  |= (*raw0 & 0x0000003fL) << 16;
                *cook  |= (*raw1 & 0x0003f000L) >> 4;
                *cook++        |= (*raw1 & 0i0000003fL);
                }
        usekey(dough);
        return;
}

void cpkey(into)
register unsigned long *into;
{
        register unsigned long *from, *endp;
        from = KnL, endp = &KnL[32];
        while( from < endp ) *into++ = *from++;
        return;
}

void usekey(from)
register unsigned long *from;
{
        register unsigned long *to, *endp;
        to = KnL, endp = &KnL[32];
        while( to < endp ) *to++ = *from++;
        return;
}

void des(inblock, outblock)
unsigned char *inblock, *outblock;
{
        unsigned long work[2];

        scrunch(inblock, work);
        desfunc(work, KnL);
        unscrun(work, outblock);
        return;
}

static void scrunch(outof, into)
register unsigned char *outof;
register unsigned long *into;
{
        *into   = (*outof++ & 0xffL) << 24;
        *into  |= (*outof++ & 0xffL) << 16;
        *into  |= (*outof++ & 0xffL) << 8;
        *into++ |= (*outof++ & 0xffL);
        *into   = (*outof++ & 0xffL) << 24;
        *into  |= (*outof++ & 0xffL) << 16;
        *into  |= (*outof++ & 0xffL) << 8;
        *into  |= (*outof   & 0xffL);
        return;
}
```

```
static void unscrun(outof, into)
register unsigned long *outof;
register unsigned char *into;
{
       *into++ = (*outof >> 24) & 0xffL;
       *into++ = (*outof >> 16) & 0xffL;
       *into++ = (*outof >>  8) & 0xffL;
       *into++ =  *outof++           & 0xffL;
       *into++ = (*outof >> 24) & 0xffL;
       *into++ = (*outof >> 16) & 0xffL;
       *into++ = (*outof >>  8) & 0xffL;
       *into   =  *outof     & 0xffL;
       return;
}
static unsigned long SP1[64] = {
       0#01010400L, 0#00000000L, 0#00010000L, 0#01010404L,
       0#01010004L, 0#00010404L, 0#00000004L, 0#00010000L,
       0#00000400L, 0#01010400L, 0#01010404L, 0#00000400L,
       0#01000404L, 0#01010004L, 0#01000000L, 0#00000004L,
       0#00000404L, 0#01000400L, 0#01000400L, 0#00010400L,
       0#00010400L, 0#01010000L, 0#01010000L, 0#01000404L,
       0#00010004L, 0#01000004L, 0#01000004L, 0#00010004L,
       0#00000000L, 0#00000404L, 0#00010404L, 0#01000000L,
       0#00010000L, 0#01010404L, 0#00000004L, 0#01010000L,
       0#01010400L, 0#01000000L, 0#01000000L, 0#00000400L,
       0#01010004L, 0#00010000L, 0#00010400L, 0#01000004L,
       0#00000400L, 0#00000004L, 0#01000404L, 0#00010404L,
       0#01010404L, 0#00010004L, 0#01010000L, 0#01000404L,
       0#01000004L, 0#00000404L, 0#00010404L, 0#01010400L,
       0#00000404L, 0#01000400L, 0#01000400L, 0#00000000L,
       0#00010004L, 0#00010400L, 0#00000000L, 0#01010004L };

static unsigned long SP2[64] = {
       0x80108020L, 0x80008000L, 0#00008000L, 0#00108020L,
       0#00100000L, 0#00000020L, 0x80100020L, 0x80008020L,
       0x80000020L, 0x80108020L, 0x80108000L, 0x80000000L,
       0x80008000L, 0#00100000L, 0#00000020L, 0x80100020L,
       0#00108000L, 0#00100020L, 0x80008020L, 0#00000000L,
       0x80000000L, 0#00008000L, 0#00108020L, 0x80100000L,
       0#00100020L, 0x80000020L, 0#00000000L, 0#00108000L,
       0#00008020L, 0x80108000L, 0x80100000L, 0#00008020L,
       0#00000000L, 0#00108020L, 0x80100020L, 0#00100000L,
       0x80008020L, 0x80100000L, 0x80108000L, 0#00008000L,
       0x80100000L, 0x80008000L, 0#00000020L, 0x80108020L,
       0#00108020L, 0#00000020L, 0#00008000L, 0x80000000L,
       0#00008020L, 0x80108000L, 0#00100000L, 0x80000020L,
       0#00100020L, 0x80008020L, 0x80000020L, 0#00100020L,
       0#00108000L, 0#00000000L, 0x80008000L, 0#00008020L,
       0x80000000L, 0x80100020L, 0x80108020L, 0#00108000L };

static unsigned long SP3[64] = {
       0#00000208L, 0#08020200L, 0#00000000L, 0#08020008L,
       0#08000200L, 0#00000000L, 0#00020208L, 0#08000200L,
       0#00020008L, 0#08000008L, 0#08000008L, 0#00020000L,
       0#08020208L, 0#00020008L, 0#08020000L, 0#00000208L,
       0#08000000L, 0#00000008L, 0#08020200L, 0#00000200L,
       0#00020200L, 0#08020000L, 0#08020008L, 0#00020208L,
       0#08000208L, 0#00020200L, 0#00020000L, 0#08000208L,
       0#00000008L, 0#08020208L, 0#00000200L, 0#08000000L,
       0#08020200L, 0#08000000L, 0#00020008L, 0#00000208L,
```

```
        0#00020000L, 0#08020200L, 0#08000200L, 0#00000000L,
        0#00000200L, 0#00020008L, 0#08020208L, 0#08000200L,
        0#08000008L, 0#00000200L, 0#00000000L, 0#08020008L,
        0#08000208L, 0#00020000L, 0#08000000L, 0#08020208L,
        0#00000008L, 0#00020208L, 0#00020200L, 0#08000008L,
        0#08020000L, 0#08000208L, 0#00000208L, 0#08020000L,
        0#00020208L, 0#00000008L, 0#08020008L, 0#00020200L };

static unsigned long SP4[64] = {
        0#00802001L, 0#00002081L, 0#00002081L, 0#00000080L,
        0#00802080L, 0#00800081L, 0#00800001L, 0#00002001L,
        0#00000000L, 0#00802000L, 0#00802000L, 0#00802081L,
        0#00000081L, 0#00000000L, 0#00800080L, 0#00800001L,
        0#00000001L, 0#00002000L, 0#00800000L, 0#00802001L,
        0#00000080L, 0#00800000L, 0#00002001L, 0#00002080L,
        0#00800081L, 0#00000001L, 0#00002080L, 0#00800080L,
        0#00002000L, 0#00802080L, 0#00802081L, 0#00000081L,
        0#00800080L, 0#00800001L, 0#00802000L, 0#00802081L,
        0#00000081L, 0#00000000L, 0#00000000L, 0#00802000L,
        0#00002080L, 0#00800080L, 0#00800081L, 0#00000001L,
        0#00802001L, 0#00002081L, 0#00002081L, 0#00000080L,
        0#00802081L, 0#00000081L, 0#00000001L, 0#00002000L,
        0#00800001L, 0#00002001L, 0#00802080L, 0#00800081L,
        0#00002001L, 0#00002080L, 0#00800000L, 0#00802001L,
        0#00000080L, 0#00800000L, 0#00002000L, 0#00802080L };

static unsigned long SP5[64] = {
        0#00000100L, 0#02080100L, 0#02080000L, 0x42000100L,
        0#00080000L, 0#00000100L, 0x40000000L, 0#02080000L,
        0x40080100L, 0#00080000L, 0#02000100L, 0x40080100L,
        0x42000100L, 0x42080000L, 0#00080100L, 0x40000000L,
        0#02000000L, 0x40080000L, 0x40080000L, 0#00000000L,
        0x40000100L, 0x42080100L, 0x42080100L, 0#02000100L,
        0x42080000L, 0x40000100L, 0#00000000L, 0x42000000L,
        0#02080100L, 0#02000000L, 0x42000000L, 0#00080100L,
        0#00080000L, 0x42000100L, 0#00000100L, 0#02000000L,
        0x40000000L, 0#02080000L, 0x42000100L, 0x40080100L,
        0#02000100L, 0x40000000L, 0x42080000L, 0#02080100L,
        0x40080100L, 0#00000100L, 0#02000000L, 0x42080000L,
        0x42080100L, 0#00080100L, 0x42000000L, 0x42080100L,
        0#02080000L, 0#00000000L, 0x40080000L, 0x42000000L,
        0#00080100L, 0#02000100L, 0x40000100L, 0#00080000L,
        0#00000000L, 0x40080000L, 0#02080100L, 0x40000100L };

static unsigned long SP6[64] = {
        0x20000010L, 0x20400000L, 0#00004000L, 0x20404010L,
        0x20400000L, 0#00000010L, 0x20404010L, 0#00400000L,
        0x20004000L, 0#00404010L, 0#00400000L, 0x20000010L,
        0#00400010L, 0x20004000L, 0x20000000L, 0#00004010L,
        0#00000000L, 0#00400010L, 0x20004010L, 0#00004000L,
        0#00404000L, 0x20004010L, 0#00000010L, 0x20400010L,
        0x20400010L, 0#00000000L, 0#00404010L, 0x20404000L,
        0#00004010L, 0#00404000L, 0x20404000L, 0x20000000L,
        0x20004000L, 0#00000010L, 0x20400010L, 0#00404000L,
        0x20404010L, 0#00400000L, 0#00004010L, 0x20000010L,
        0#00400000L, 0x20004000L, 0x20000000L, 0#00004010L,
        0x20000010L, 0x20404010L, 0#00404000L, 0x20400000L,
        0#00404010L, 0x20404000L, 0#00000000L, 0x20400010L,
        0#00000010L, 0#00004000L, 0x20400000L, 0#00404010L,
        0#00004000L, 0#00400010L, 0x20004010L, 0#00000000L,
        0x20404000L, 0x20000000L, 0#00400010L, 0x20004010L };
```

```
static unsigned long SP7[64] = {
        0#00200000L, 0#04200002L, 0#04000802L, 0#00000000L,
        0#00000800L, 0#04000802L, 0#00200802L, 0#04200800L,
        0#04200802L, 0#00200000L, 0#00000000L, 0#04000002L,
        0#00000002L, 0#04000000L, 0#04200002L, 0#00000802L,
        0#04000800L, 0#00200802L, 0#00200002L, 0#04000800L,
        0#04000002L, 0#04200000L, 0#04200800L, 0#00200002L,
        0#04200000L, 0#00000800L, 0#00000802L, 0#04200802L,
        0#00200800L, 0#00000002L, 0#04000000L, 0#00200800L,
        0#04000000L, 0#00200800L, 0#00200000L, 0#04000802L,
        0#04000802L, 0#04200002L, 0#04200002L, 0#00000002L,
        0#00200002L, 0#04000000L, 0#04000800L, 0#00200000L,
        0#04200800L, 0#00000802L, 0#00200802L, 0#04200800L,
        0#00000802L, 0#04000002L, 0#04200802L, 0#04200000L,
        0#00200800L, 0#00000000L, 0#00000002L, 0#04200802L,
        0#00000000L, 0#00200802L, 0#04200000L, 0#00000800L,
        0#04000002L, 0#04000800L, 0#00000800L, 0#00200002L };

static unsigned long SP8[64] = {
        0x10001040L, 0#00001000L, 0#00040000L, 0x10041040L,
        0x10000000L, 0x10001040L, 0#00000040L, 0x10000000L,
        0#00040040L, 0x10040000L, 0x10041040L, 0#00041000L,
        0x10041000L, 0#00041040L, 0#00001000L, 0#00000040L,
        0x10040000L, 0x10000040L, 0x10001000L, 0#00001040L,
        0#00041000L, 0#00040040L, 0x10040040L, 0x10041000L,
        0#00001040L, 0#00000000L, 0#00000000L, 0x10040040L,
        0x10000040L, 0x10001000L, 0#00041040L, 0#00040000L,
        0#00041040L, 0#00040000L, 0x10041000L, 0#00001000L,
        0#00000040L, 0x10040040L, 0#00001000L, 0#00041040L,
        0x10001000L, 0#00000040L, 0x10000040L, 0x10040000L,
        0x10040040L, 0x10000000L, 0#00040000L, 0x10001040L,
        0#00000000L, 0x10041040L, 0#00040040L, 0x10000040L,
        0x10040000L, 0x10001000L, 0x10001040L, 0#00000000L,
        0x10041040L, 0#00041000L, 0#00041000L, 0#00001040L,
        0#00001040L, 0#00040040L, 0x10000000L, 0x10041000L };

static void desfunc(block, keys)
register unsigned long *block, *keys;
{
        register unsigned long fval, work, right, leftt;
        register int round;

        leftt = block[0];
        right = block[1];
        work = ((leftt >> 4) ^ right) & 0#0f0f0f0fL;
        right ^= work;
        leftt ^= (work << 4);
        work = ((leftt >> 16) ^ right) & 0#0000ffffL;
        right ^= work;
        leftt ^= (work << 16);
        work = ((right >> 2) ^ leftt) & 0x33333333L;
        leftt ^= work;
        right ^= (work << 2);
        work = ((right >> 8) ^ leftt) & 0#00ff00ffL;
        leftt ^= work;
        right ^= (work << 8);
        right = ((right << 1) | ((right >> 31) & 1L)) & 0xffffffffL;
        work = (leftt ^ right) & 0xaaaaaaaaL;
        leftt ^= work;
        right ^= work;
```

```
        leftt = ((leftt << 1) | ((leftt >> 31) & 1L)) & 0xffffffffL;

        for( round = 0; round < 8; round++ ) {
               work  = (right << 28) | (right >> 4);
               work ^= *keys++;
               fval  = SP7[ work             & 0x3fL];
               fval |= SP5[(work >>  8) & 0x3fL];
               fval |= SP3[(work >> 16) & 0x3fL];
               fval |= SP1[(work >> 24) & 0x3fL];
               work  = right ^ *keys++;
               fval |= SP8[ work             & 0x3fL];
               fval |= SP6[(work >>  8) & 0x3fL];
               fval |= SP4[(work >> 16) & 0x3fL];
               fval |= SP2[(work >> 24) & 0x3fL];
               leftt ^= fval;
               work  = (leftt << 28) | (leftt >> 4);
               work ^= *keys++;
               fval  = SP7[ work             & 0x3fL];
               fval |= SP5[(work >>  8) & 0x3fL];
               fval |= SP3[(work >> 16) & 0x3fL];
               fval |= SP1[(work >> 24) & 0x3fL];
               work  = leftt ^ *keys++;
               fval |= SP8[ work             & 0x3fL];
               fval |= SP6[(work >>  8) & 0x3fL];
               fval |= SP4[(work >> 16) & 0x3fL];
               fval |= SP2[(work >> 24) & 0x3fL];
               right ^= fval;
               }

        right = (right << 31) | (right >> 1);
        work = (leftt ^ right) & 0xaaaaaaaaL;
        leftt ^= work;
        right ^= work;
        leftt = (leftt << 31) | (leftt >> 1);
        work = ((leftt >> 8) ^ right) & 0#00ff00ffL;
        right ^= work;
        leftt ^= (work << 8);
        work = ((leftt >> 2) ^ right) & 0x33333333L;
        right ^= work;
        leftt ^= (work << 2);
        work = ((right >> 16) ^ leftt) & 0#0000ffffL;
        leftt ^= work;
        right ^= (work << 16);
        work = ((right >> 4) ^ leftt) & 0#0f0f0f0fL;
        leftt ^= work;
        right ^= (work << 4);
        *block++ = right;
        *block = leftt;
        return;
}
/* Validation sets:
 *
 * Single-length key, single-length plaintext -
 * Key    : 0123 4567 89ab cdef
 * Plain  : 0123 4567 89ab cde7
 * Cipher : c957 4425 6a5e d31d
 *
 ********************************************************************/

void des_key(des_ctx *dc, unsigned char *key){
        deskey(key,EN0);
```

```
        cpkey(dc->ek);
        deskey(key,DE1);
        cpkey(dc->dk);
}
/* Encrypt several blocks in ECB mode.  Caller is responsible for
   short blocks. */
void des_enc(des_ctx *dc, unsigned char *data, int blocks){
        unsigned long work[2];
        int i;
        unsigned char *cp;
        cp = data;
        for(i=0;iek);
                unscrun(work,cp);
                cp+=8;
        }
}

void des_dec(des_ctx *dc, unsigned char *data, int blocks){
        unsigned long work[2];
        int i;
        unsigned char *cp;

        cp = data;
        for(i=0;idk);
                unscrun(work,cp);
                cp+=8;
        }
}
void main(void){
        des_ctx dc;
        int i;
        unsigned long data[10];
        char *cp,key[8] = {0#01,0x23,0x45,0x67,0x89,0xab,0xcd,0xef};
        char x[8] = {0#01,0x23,0x45,0x67,0x89,0xab,0xcd,0xe7};

        cp = x;

        des_key(&dc,key);
        des_enc(&dc,cp,1);
        printf("Enc(0..7,0..7) = ");
        for(i=0;i<8;i++) printf("%02x ", ((unsigned int) cp[i])&0#00ff);
        printf("\n");

        des_dec(&dc,cp,1);

        printf("Dec(above,0..7) = ");
        for(i=0;i<8;i++) printf("%02x ",((unsigned int)cp[i])&0#00ff);
        printf("\n");

        cp = (char *) data;
        for(i=0;i<10;i++)data[i]=i;

        des_enc(&dc,cp,5); /* Enc 5 blocks. */
        for(i=0;i<10;i+=2) printf("Block %01d = %08lx %08lx.\n",
                                i/2,data[i],data[i+1]);

        des_dec(&dc,cp,1);
        des_dec(&dc,cp+8,4);
        for(i=0;i<10;i+=2) printf("Block %01d = %08lx %08lx.\n",
                                i/2,data[i],data[i+1]);
}
```