Chương 1: TỔNG QUAN VỀ AN TOÀN VÀ BẢO MẬT THÔNG TIN

1.1. Nội dung của an toàn và bảo mật thông tin

Khi nhu cầu trao đổi thông tin dữ liệu ngày càng lớn và đa dạng, các tiến bộ về điện tử - viễn thông và công nghệ thông tin không ngừng được phát triển ứng dụng để nâng cao chất lượng và lưu lượng truyền tin thì các quan niệm ý tưởng và biện pháp bảo vệ thông tin dữ liệu cũng được đổi mới. Bảo vệ an toàn thông tin dữ liệu là một chủ đề rộng, có liên quan đến nhiều lĩnh vực và trong thực tế có thể có rất nhiều phương pháp được thực hiện để bảo vệ an toàn thông tin dữ liệu. Các phương pháp bảo vệ an toàn thông tin dữ liệu có thể được quy tụ vào ba nhóm sau:

- Bảo vệ an toàn thông tin bằng các biện pháp hành chính.
- Bảo vệ an toàn thông tin bằng các biện pháp kỹ thuật (phần cứng).
- Bảo vệ an toàn thông tin bằng các biện pháp thuật toán (phần mềm).

Ba nhóm trên có thể được ứng dụng riêng rẽ hoặc phối kết hợp. Môi trường khó bảo vệ an toàn thông tin nhất và cũng là môi trường đối phương dễ xân nhập nhất đó là môi trường mạng và truyền tin. Biện pháp hiệu quả nhất và kinh tế nhất hiện nay trên mạng truyền tin và mạng máy tính là biện pháp thuật toán.

An toàn thông tin bao gồm các nội dung sau:

- Tính bí mật: tính kín đáo riêng tư của thông tin
- Tính xác thực của thông tin, bao gồm xác thực đối tác(bài toán nhận danh), xác thực thông tin trao đổi.
- Tính trách nhiệm: đảm bảo người gửi thông tin không thể thoái thác trách nhiệm về thông tin mà mình đã gửi.

Để đảm bảo an toàn thông tin dữ liệu trên đường truyền tin và trên mạng máy tính có hiệu quả thì điều trước tiên là phải lường trước hoặc dự đoán trước các khả năng không an toàn, khả năng xâm phạm, các sự cố rủi ro có thể xảy ra đối với thông tin dữ liệu được lưu trữ và trao đổi trên đường truyền tin cũng như

trên mạng. Xác định càng chính xác các nguy cơ nói trên thì càng quyết định được tốt các giải pháp để giảm thiểu các thiệt hại.

Có hai loại hành vi xâm phạm thông tin dữ liệu đó là: *vi phạm chủ động và vi phạm thụ động*. Vi phạm thụ động chỉ nhằm mục đích cuối cùng là nắm bắt được thông tin (đánh cắp thông tin). Việc làm đó có khi không biết được nội dung cụ thể nhưng có thể dò ra được người gửi, người nhận nhờ thông tin điều khiển giao thức chứa trong phần đầu các gói tin. Kẻ xâm nhập có thể kiểm tra được số lượng, độ dài và tần số trao đổi. Vì vậy vi pham thụ động không làm sai lệch hoặc hủy hoại nội dung thông tin dữ liệu được trao đổi. Vi phạm thụ động thường khó phát hiện nhưng có thể có những biện pháp ngăn chặn hiệu quả. Vi phạm chủ động là dạng vi phạm có thể làm thay đổi nội dung, xóa bỏ, làm trễ, xắp xếp lại thứ tự hoặc làm lặp lại gói tin tại thời điểm đó hoặc sau đó một thời gian. Vi phạm chủ động có thể thêm vào một số thông tin ngoại lai để làm sai lệch nội dung thông tin trao đổi. Vi phạm chủ động dễ phát hiện nhưng để ngăn chặn hiệu quả thì khó khăn hơn nhiều.

Một thực tế là không có một biện pháp bảo vệ an toàn thông tin dữ liệu nào là an toàn tuyệt đối. Một hệ thống dù được bảo vệ chắc chắn đến đâu cũng không thể đảm bảo là an toàn tuyệt đối.

1.2. Các chiến lượt an toàn hệ thống :

a. Giới hạn quyền hạn tối thiểu (Last Privilege):

Đây là chiến lược cơ bản nhất theo nguyên tắc này bất kỳ một đối tượng nào cùng chỉ có những quyền hạn nhất định đối với tài nguyên mạng, khi thâm nhập vào mạng đối tượng đó chỉ được sử dụng một số tài nguyên nhất định.

b. Bảo vệ theo chiều sâu (Defence In Depth):

Nguyên tắc này nhắc nhở chúng ta : Không nên dựa vào một chế độ an toàn nào dù cho chúng rất mạnh, mà nên tạo nhiều cơ chế an toàn để tương hỗ lẫn nhau.

c. Nút thắt (Choke Point) :

Tạo ra một "cửa khẩu" hẹp, và chỉ cho phép thông tin đi vào hệ thống của mình bằng con đường duy nhất chính là "cửa khẩu" này. => phải tổ chức một cơ cấu kiểm soát và điều khiển thông tin đi qua cửa này.

d. Điểm nối yếu nhất (Weakest Link) :

Chiến lược này dựa trên nguyên tắc: " Một dây xích chỉ chắc tại mắt duy nhất, một bức tường chỉ cứng tại điểm yếu nhất"

Kẻ phá hoại thường tìm những chỗ yếu nhất của hệ thống để tấn công, do đó ta cần phải gia cố các yếu điểm của hệ thống. Thông thường chúng ta chỉ quan tâm đến kẻ tấn công trên mạng hơn là kẻ tiếp cận hệ thống, do đó an toàn vật lý được coi là yếu điểm nhất trong hệ thống của chúng ta.

e. Tính toàn cục:

Các hệ thống an toàn đòi hỏi phải có tính toàn cục của các hệ thống cục bộ. Nếu có một kẻ nào đó có thể bẻ gãy một cơ chế an toàn thì chúng có thể thành công bằng cách tấn công hệ thống tự do của ai đó và sau đó tấn công hệ thống từ nội bộ bên trong.

f. **Tính đa dạng bảo vệ :**Cần phải sử dụng nhiều biện pháp bảo vệ khác nhau cho hệ thống khác nhau, nếu không có kẻ tấn công vào được một hệ thống thì chúng cũng dễ dàng tấn công vào các hệ thống khác.

1.3 Các mức bảo vệ trên mạng :

Vì không thể có một giải pháp an toàn tuyệt đối nên người ta thường phải sử dụng đồng thời nhiều mức bảo vệ khác nhau tạo thành nhiều hàng rào chắn đối với các hoạt động xâm phạm. Việc bảo vệ thông tin trên mạng chủ yếu là bảo vệ thông tin cất giữ trong máy tính, đặc biệt là các server trên mạng. Bởi thế ngoài một số biện pháp nhằm chống thất thoát thông tin trên đường truyền mọi cố gắng tập trung vào việc xây dựng các mức rào chắn từ ngoài vào trong cho các hệ thống kết nối vào mạng. Thông thường bao gồm các mức bảo vệ sau:

a. Quyền truy nhập

Lớp bảo vệ trong cùng là quyền truy nhập nhằm kiểm soát các tài nguyên của mạng và quyền hạn trên tài nguyên đó. Dĩ nhiên là kiểm soát được các cấu trúc dữ liệu càng chi tiết càng tốt. Hiện tại việc kiểm soát thường ở mức tệp.

b. Đăng ký tên /mật khẩu.

Thực ra đây cũng là kiểm soát quyền truy nhập, nhưng không phải truy nhập ở mức thông tin mà ở mức hệ thống. Đây là phương pháp bảo vệ phổ biến nhất vì nó đơn giản ít phí tổn và cũng rất hiệu quả. Mỗi người sử dụng muốn được tham gia vào mạng để sử dụng tài nguyên đều phải có đăng ký tên và mật khẩu trước. Người quản trị mạng có trách nhiệm quản lý, kiểm soát mọi hoạt động của mạng và xác định quyền truy nhập của những người sử dụng khác theo thời gian và không gian (nghĩa là người sử dụng chỉ được truy nhập trong một khoảng thời gian nào đó tại một vị trí nhất định nào đó).

Về lý thuyết nếu mọi người đều giữ kín được mật khẩu và tên đăng ký của mình thì sẽ không xảy ra các truy nhập trái phép. Song điều đó khó đảm bảo trong thực tế vì nhiều nguyên nhân rất đời thường làm giảm hiệu quả của lớp bảo vệ này. Có thể khắc phục bằng cách người quản mạng chịu trách nhiệm đặt mật khẩu hoặc thay đổi mật khẩu theo thời gian.

c. Mã hoá dữ liệu

Để bảo mật thông tin trên đường truyền người ta sử dụng các phương pháp mã hoá. Dữ liệu bị biến đổi từ dạng nhận thức được sang dạng không nhận thức

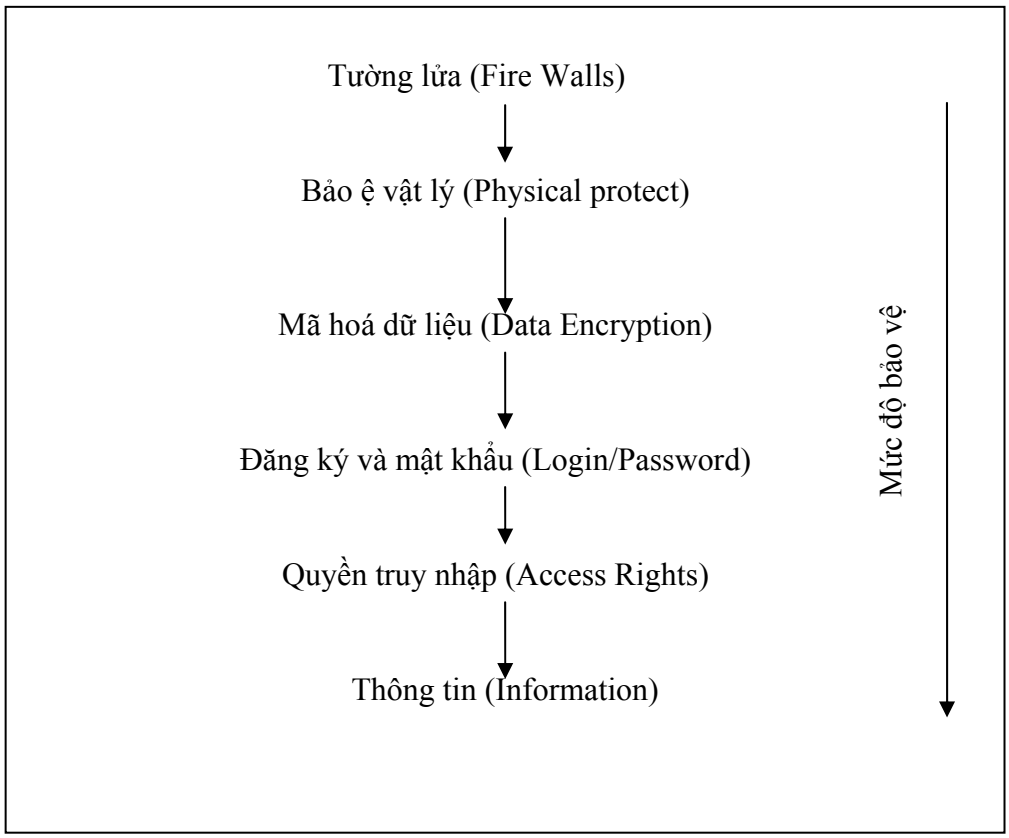
được theo một thuật toán nào đó và sẽ được biến đổi ngược lại ở trạm nhận (giải mã). Đây là lớp bảo vệ thông tin rất quan trọng.

d. Bảo vệ vật lý

Ngăn cản các truy nhập vật lý vào hệ thống. Thường dùng các biện pháp truyền thống như ngăn cấm tuyệt đối người không phận sự vào phòng đặt máy mạng, dùng ổ khoá trên máy tính hoặc các máy trạm không có ổ mềm.

e. Tường lửa

Ngăn chặn thâm nhập trái phép và lọc bỏ các gói tin không muốn gửi hoặc nhận vì các lý do nào đó để bảo vệ một máy tính hoặc cả mạng nội bộ (intranet)



f. Quản trị mạng.

Trong thời đại phát triển của công nghệ thông tin, mạng máy tính quyết định toàn bộ hoạt động của một cơ quan, hay một công ty xí nghiệp. Vì vậy việc bảo đảm cho hệ thống mạng máy tính hoạt động một cách an toàn, không xảy ra sự cố là một công việc cấp thiết hàng đầu. Công tác quản trị mạng máy tính phải được thực hiện một cách khoa học đảm bảo các yêu cầu sau :

- Toàn bộ hệ thống hoạt động bình thường trong giờ làm việc.

- Có hệ thống dự phòng khi có sự cố về phần cứng hoặc phần mềm xảy ra.

- Backup dữ liệu quan trọng theo định kỳ.

- Bảo dưỡng mạng theo định kỳ.

- Bảo mật dữ liệu, phân quyền truy cập, tổ chức nhóm làm việc trên mạng.

1.4. An toàn thông tin bằng mật mã

Mật mã là một ngành khoa học chuyên nghiên cứu các phương pháp truyền tin bí mật. Mật mã bao gồm : Lập mã và phá mã. Lập mã bao gồm hai quá trình: mã hóa và giải mã.

Để bảo vệ thông tin trên đường truyền người ta thường biến đổi nó từ dạng nhận thức được sang dạng không nhận thức được trước khi truyền đi trên mạng, quá trình này được gọi là mã hoá thông tin (encryption), ở trạm nhận phải thực hiện quá trình ngược lại, tức là biến đổi thông tin từ dạng không nhận thức được (dữ liệu đã được mã hoá) về dạng nhận thức được (dạng gốc), quá trình này được gọi là giải mã. Đây là một lớp bảo vệ thông tin rất quan trọng và được sử dụng rộng rãi trong môi trường mạng.

Để bảo vệ thông tin bằng mật mã người ta thường tiếp cận theo hai hướng:

- Theo đường truyền (Link_Oriented_Security).

- Từ nút đến nút (End_to_End).

Theo cách thứ nhất thông tin được mã hoá để bảo vệ trên đường truyền giữa hai nút mà không quan tâm đến nguồn và đích của thông tin đó. Ở đây ta lưu ý rằng thông tin chỉ được bảo vệ trên đường truyền, tức là ở mỗi nút đều có quá trình giải mã sau đó mã hoá để truyền đi tiếp, do đó các nút cần phải được bảo vệ tốt.

Ngược lại theo cách thứ hai thông tin trên mạng được bảo vệ trên toàn đường truyền từ nguồn đến đích. Thông tin sẽ được mã hoá ngay sau khi mới tạo ra và chỉ được giải mã khi về đến đích. Cách này mắc phải nhược điểm là

chỉ có dữ liệu của người □ung thì mới có thể mã hóa được còn dữ liệu điều khiển thì giữ nguyên để có thể xử lý tại các nút.

1.5. Vai trò của hệ mật mã

Các hệ mật mã phải thực hiện được các vai trò sau:

- Hệ mật mã phải che dấu được nội dung của văn bản rõ (PlainText) để đảm bảo sao cho chỉ người chủ hợp pháp của thông tin mới có quyền truy cập thông tin (Secrety), hay nói cách khác là chống truy nhập không đúng quyền hạn.

- Tạo các yếu tố xác thực thông tin, đảm bảo thông tin lưu hành trong hệ thống đến người nhận hợp pháp là xác thực (Authenticity).

- Tổ chức các sơ đồ chữ ký điện tử, đảm bảo không có hiện tượng giả mạo, mạo danh để gửi thông tin trên mạng.

Ưu điểm lớn nhất của bất kỳ hệ mật mã nào đó là có thể đánh giá được độ phức tạp tính toán mà "kẻ địch" phải giải quyết bài toán để có thể lấy được thông tin của dữ liệu đã được mã hoá. Tuy nhiên mỗi hệ mật mã có một số ưu và nhược điểm khác nhau, nhưng nhờ đánh giá được độ phức tạp tính toán mà ta có thể áp dụng các thuật toán mã hoá khác nhau cho từng ứng dụng cụ thể tuỳ theo dộ yêu cầu về đọ an toàn.

Các thành phần của một hệ mật mã :

Định nghĩa :

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

- P là một tập hợp hữu hạn các bản rõ (PlainText), nó được gọi là không gian bản rõ.

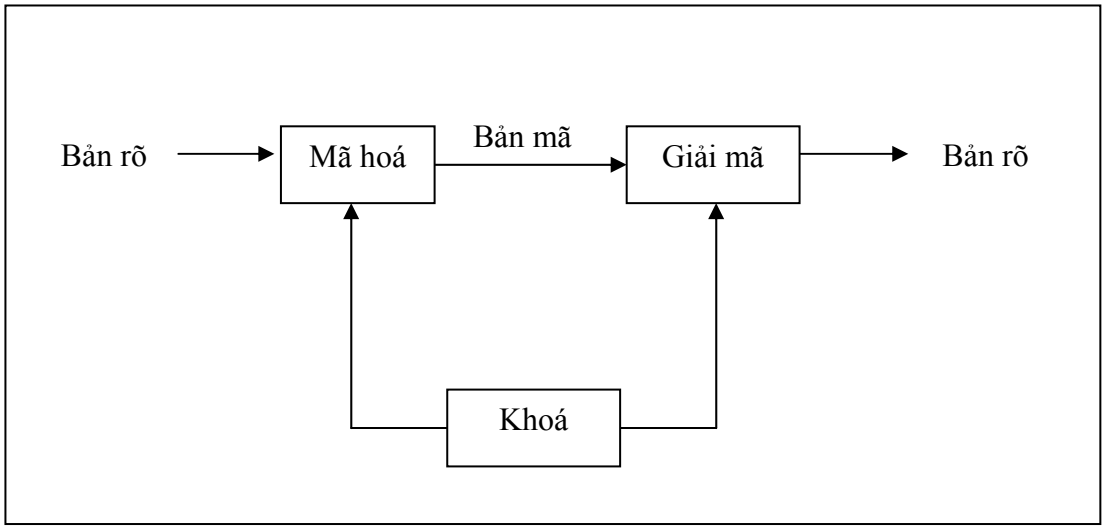
- C là tập các hữu hạn các bản mã (Crypto), nó còn được gọi là không gian các bản mã. Mỗi phần tử của C có thể nhận được bằng cách áp dụng phép mã hoá E_k lên một phần tử của P, với $k \in K$.

- K là tập hữu hạn các khoá hay còn gọi là không gian khoá. Đối với mỗi phần tử k của K được gọi là một khoá (Key). Số lượng của không gian khoá

phải đủ lớn để "kẻ địch: không có đủ thời gian để thử mọi khoá có thể (phương pháp vét cạn).

- Đối với mỗi $k \in K$ có một quy tắc mã $e_K: P \rightarrow C$ và một quy tắc giải mã tương ứng $d_K \in D$. Mỗi $e_K: P \rightarrow C$ và $d_K: C \rightarrow P$ là những hàm mà:

$d_K (e_K(x))=x$ với mọi bản rõ $x \in P$.



Mã hoá với khoá mã và khoá giải giống nhau

1.6. Phân loại hệ mật mã

Có nhiều cách để phân loại hệ mật mã. Dựa vào cách truyền khóa có thể phân các hệ mật mã thành hai loại:

- Hệ mật đối xứng (hay còn gọi là mật mã khóa bí mật): là những hệ mật dung chung một khoá cả trong quá trình mã hoá dữ liệu và giải mã dữ liệu. Do đó khoá phải được giữ bí mật tuyệt đối.

- Hệ mật mã bất đối xứng (hay còn gọi là mật mã khóa công khai) : Hay còn gọi là hệ mật mã công khai, các hệ mật này dùng một khoá để mã hoá sau đó dùng một khoá khác để giải mã, nghĩa là khoá để mã hoá và giải mã là khác nhau. Các khoá này tạo nên từng cặp chuyển đổi ngược nhau và không có khoá nào có thể suy được từ khoá kia. Khoá dùng để mã hoá có thể công khai nhưng khoá dùng để giải mã phải giữ bí mật.

Ngoài ra nếu dựa vào thời gian đưa ra hệ mật mã ta còn có thể phân làm hai loại: Mật mã cổ điển (là hệ mật mã ra đời trước năm 1970) và mật mã hiện đại (ra đời sau năm 1970). Còn nếu dựa vào cách thức tiến hành mã thì hệ mật mã còn được chia làm hai loại là mã dòng (tiến hành mã từng khối dữ liệu, mỗi khối lại dựa vào các khóa khác nhau, các khóa này được sinh ra từ hàm sinh khóa, được gọi là dòng khóa) và mã khối (tiến hành mã từng khối dữ liệu với khóa như nhau)

1.7. Tiêu chuẩn đánh giá hệ mật mã

Để đánh giá một hệ mật mã người ta thường đánh giá thông qua các tính chất sau:

a, Độ an toàn: Một hệ mật được đưa vào sử dụng điều đầu tiên phải có độ an toàn cao. Ưu điểm của mật mã là có thể đánh giá được độ an toàn thông qua độ an toàn tính toán mà không cần phải cài đặt. Một hệ mật được coi là an toàn nếu để phá hệ mật mã này phải dùng n phép toán. Mà để giải quyết n phép toán cần thời gian vô cùng lớn, không thể chấp nhận được.

Một hệ mật mã được gọi là tốt thì nó cần phải đảm bảo các tiêu chuẩn sau:

- Chúng phải có phương pháp bảo vệ mà chỉ dựa trên sự bí mật của các khoá, công khai thuật toán.

- Khi cho khoá công khai e_K và bản rõ P thì chúng ta dễ dàng tính được $e_K(P) = C$. Ngược lại khi cho d_K và bản mã C thì dễ dàng tính được $d_K(M)=P$. Khi không biết d_K thì không có khả năng để tìm được M từ C, nghĩa là khi cho hàm f: $X \rightarrow Y$ thì việc tính y=f(x) với mọi $x \in X$ là dễ còn việc tìm x khi biết y lại là vấn đề khó và nó được gọi là hàm một chiều.

- Bản mã C không được có các đặc điểm gây chú ý, nghi ngờ.

b, Tốc độ mã và giải mã: Khi đánh giá hệ mật mã chúng ta phải chú ý đến tốc độ mã và giải mã. Hệ mật tốt thì thời gian mã và giải mã nhanh.

c, Phân phối khóa: Một hệ mật mã phụ thuộc vào khóa, khóa này được truyền công khai hay truyền khóa bí mật. Phân phối khóa bí mật thì chi phí sẽ cao hơn so với các hệ mật có khóa công khai. Vì vậy đây cũng là một tiêu chí khi lựa chọn hệ mật mã.

Chương 2: CÁC PHƯƠNG PHÁP MÃ HÓA CỔ ĐIỂN

2.1. Các hệ mật mã cổ điển

2.1.1. Mã dịch vòng (shift cipher)

Phần này sẽ mô tả mã dịch (MD) dựa trên số học theo modulo. Trước tiên sẽ điểm qua một số định nghĩa cơ bản của số học này.

Định nghĩa

Giả sử a và b là các số nguyên và m là một số nguyên dương. Khi đó ta viết a ≡ b (mod m) nếu m chia hết cho b-a. Mệnh đề a ≡ b (mod m) được gọi là " a đồng dư với b theo modulo m". Số nguyên m được gọi là mudulus.

Giả sử chia a và b cho m và ta thu được phần thương nguyên và phần dư, các phần dư nằm giữa 0 và m-1, nghĩa là $a = q_1m + r_1$ và $b = q_2m + r_2$ trong đó $0 \leq r_1 \leq m-1$ và $0 \leq r_2 \leq m-1$. Khi đó có thể dễ dàng thấy rằng $a \equiv b \pmod m$ khi và chỉ khi $r_1 = r_2$. Ta sẽ dùng ký hiệu a mod m (không dùng các dấu ngoặc) để xác định phần dư khi a được chia cho m (chính là giá trị r_1 ở trên). Như vậy: $a \equiv b \pmod m$ khi và chỉ khi a mod m = b mod m. Nếu thay a bằng a mod m thì ta nói rằng a được rút gọn theo modulo m.

Nhận xét: Nhiều ngôn ngữ lập trình của máy tính xác định a mod m là phần dư trong dải - m+1,.. ., m-1 có cùng dấu với a. Ví dụ -18 mod 7 sẽ là -4, giá trị này khác với giá trị 3 là giá trị được xác định theo công thức trên. Tuy nhiên, để thuận tiện ta sẽ xác định a mod m luôn là một số không âm.

Bây giờ ta có thể định nghĩa số học modulo m: Z_m được coi là tập hợp {0,1,. . .,m-1} có trang bị hai phép toán cộng và nhân. Việc cộng và nhân trong Z_m được thực hiện giống như cộng và nhân các số thực ngoài trừ một điểm là các kết quả được rút gọn theo modulo m.

Ví dụ tính 11× 13 trong Z_{16} . Tương tự như với các số nguyên ta có 11 ×13 = 143. Để rút gọn 143 theo modulo 16, ta thực hiện phép chia bình thường: 143 = 8 × 16 + 15, bởi vậy 143 mod 16 = 15 trong Z_{16} .

Các định nghĩa trên phép cộng và phép nhân Z_m thảo mãn hầu hết các quy tắc quen thuộc trong số học. Sau đây ta sẽ liệt kê mà không chứng minh các tính chất này:

1. Phép cộng là đóng, tức với bất kì a,b $\in Z_m$,a +b $\in Z_m$
2. Phép cộng là giao hoán, tức là với a,b bất kì $\in Z_m$

a+b = b+a

3. Phép cộng là kết hợp, tức là với bất kì a,b,c $\in Z_m$

(a+b)+c = a+(b+c)

4. 0 là phần tử đơn vị của phép cộng, có nghĩa là với a bất kì $\in Z_m$

a+0 = 0+a = a

5. Phần tử nghịch đảo của phép cộng của phần tử bất kì (a $\in Z_m$) là m-a, nghĩa là a+(m-a) = (m-a)+a = 0 với bất kì a $\in Z_m$.
6. Phép nhân là đóng , tức là với a,b bất kì $\in Z_m$, ab $\in Z_m$.
7. Phép nhân là giao hoán , nghĩa là với a,b bất kì $\in Z_m$, ab = ba
8. Phép nhân là kết hợp, nghĩa là với a,b,c $\in Z_m$, (ab)c = a(cb)
9. 1 là phần tử đơn vị của phép nhân, tức là với bất kỳ a $\in Z_m$

a×1 = 1×a = a

10. Phép nhân có tính chất phân phối đối với phép cộng, tức là đối với a,b,c $\in Z_m$, (a+b)c = (ac)+(bc) và a(b+c) = (ab) + (ac)

Các tính chất 1,3-5 nói lên rằng Z_m lập nên một cấu trúc đại số được gọi là một nhóm theo phép cộng. Vì có thêm tính chất 4 nhóm được gọi là nhóm Aben (hay nhóm giao hoán).

Các tính chất 1-10 sẽ thiết lập nên một vành Z_m . Một số ví dụ quen thuộc của vành là các số nguyên Z, các số thực R và các số phức C. Tuy nhiên các vành này đều vô hạn, còn mối quan tâm của chúng ta chỉ giới hạn trên các vành hữu hạn.

Vì phần tử ngược của phép cộng tồn tại trong Z_m nên cũng có thể trừ các phần tử trong Z_m . Ta định nghĩa a-b trong Z_m là a+m-b mod m. Một cách tương tự có thể tính số nguyên a-b rồi rút gon theo modulo m.

Ví dụ : Để tính 11-18 trong Z_{31}, ta tính 11+31 – 18 mod 31= 11+13 mod 31 = 24. Ngược lại, có thể lấy 11-18 được -7 rồi sau đó tính -7 mod 31 =31-7= 24.

Mã dịch vòng được xác định trên Z_{26} (do có 26 chữ cái trên bảng chữ cái tiếng Anh) mặc dù có thể xác định nó trên Z_m với modulus m tuỳ ý. Dễ dàng thấy rằng, MDV sẽ tạo nên một hệ mật như đã xác định ở trên, tức là $d_K(e_K(x)) = x$ với mọi $x \in Z_{26}$. Ta có sơ đồ mã như sau:

Giả sử $P = C = K = Z_{26}$ với $0 \le k \le 25$, định nghĩa:

$e_K(x) = x + K \bmod 26$

và $d_K(x) = y - K \bmod 26$

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

Nhận xét: Trong trường hợp K = 3, hệ mật thường được gọi là mã Caesar đã từng được Julius Caesar sử dụng.

Ta sẽ 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 kí tự và các thặng dư theo modulo 26 như sau: A $\leftrightarrow$ 0,B $\leftrightarrow$ 1, . . ., Z $\leftrightarrow$ 25. Vì phép tương ứng này còn dùng trong một vài ví dụ nên ta sẽ ghi lại để còn tiện dùng sau này:

Sau đây là một ví dụ nhỏ để minh hoạ

Ví dụ 1.1:

Giả sử khoá cho MDV là K = 11 và bản rõ là:

wewillmeetatmidnight

Trước tiên biến đổi bản rõ thành dãy các số nguyên nhờ dùng phép tương ứng trên. Ta có:

22	4	22	8	11	11	12	4	4	19
0	19	12	8	3	13	8	6	7	19

sau đó cộng 11 vào mỗi giá trị rồi rút gọn tổng theo modulo 26

7	15	7	19	22	22	23	15	15	4
11	4	23	19	14	24	19	17	18	4

Cuối cùng biến đổi dãy số nguyên này thành các kí tự thu được bản mã sau:

HPHTWWXPPELEXTOYTRSE

Để giả mã bản mã này, trước tiên, Bob sẽ biến đổi bản mã thành dãy các số nguyên rồi trừ đi giá trị cho 11 (rút gọn theo modulo 26) và cuối cùng biến đổi lại dãy này thành các ký tự.

Nhận xét: Trong ví dụ trên, ta đã dùng các chữ in hoa cho bản mã, các chữ thường cho bản rõ để tiện phân biệt. Quy tắc này còn tiếp tục sử dụng sau này.

Nếu một hệ mật có thể sử dụng được trong thực tế thì nó phảo thoả mãn một số tính chất nhất định. Ngay sau đây sẽ nêu ra hai trong số đó:

1. Mỗi hàm mã hoá e_K và mỗi hàm giải mã d_K phải có khả năng tính toán được một cách hiệu quả.

2. Đối phương dựa trên xâu bản mã phải không có khả năng xác định khoá K đã dùng hoặc không có khả năng xác định được xâu bản rõ x.

Tính chất thứ hai xác định (theo cách khá mập mờ) ý tưởng "bảo mật". Quá trình thử tính khoá K (khi đã biết bản mã y) được gọi là mã thám (sau này khái niệm này sẽ được làm chính xác hơn). Cần chú ý rằng, nếu Oscar có thể xác định được K thì anh ta có thể giải mã được y như Bob bằng cách dùng d_K. Bởi vậy, việc xác định K chí ít cũng khó như việc xác định bản rõ x.

Nhận xét rằng, MDV (theo modulo 26) là không an toàn vì nó có thể bị thám theo phương pháp vét cạn. Do chỉ có 26 khoá nên dễ dàng thử mọi khoá d_K

có thể cho tới khi nhận được bản rõ có nghĩa. Điều này được minh hoạ theo ví dụ sau:

Ví du 1.2

Cho bản mã

JBCRCLQRWCRVNBJENBWRWN

ta sẽ thử liên tiếp các khoá giải mã d_0 ,d_1 .. . và y thu được:

j b c r c l q r w c r v n b j e n b w r w n

i a b q b k p q v b q u m a i d m a v q v m

h z a p a j o p u a p t l z h c l z u p u l

g y z o z i n o t z o s k y g b k y t o t k

j x y n y h m n s y n r j e x f a j x s n s j

e w x m x g l m r x m q i w e z i w r m r i

d v w l w f k l q w l p h v o d y h v q l q h

c u v k v e j k p v k o g u c x g u p k p g

b t u j u d i j o u j n f t b w f o j o f

a s t i t c h i n t i m e s a v e s n i n e

Tới đây ta đã xác định được bản rõ và dừng lại. Khoá tương ứng K = 9.

Trung bình có thể tính được bản rõ sau khi thử 26/2 = 13 quy tắc giải mã.

Như đã chỉ ra trong ví dụ trên, điều kiện để một hệ mật an toàn là phép tìm khoá vét cạn phải không thể thực hiện được, tức không gian khoá phải rất lớn. Tuy nhiên, một không gian khoá lớn vẫn chưa đủ đảm bảo độ mật.

2.1.2. Mã thay thế

Một hệ mật nổi tiếng khác là hệ mã thay thế. Hệ mật này đã được sử dụng hàng trăm năm. Trò chơi đố chữ "*cryptogram*" trong các bài báo là những ví dụ về MTT.

Trên thực tế MTT có thể lấy cả P và C đều là bộ chữ cái tiếng anh, gồm 26 chữ cái. Ta dùng Z_{26} trong MDV vì các phép mã và giải mã đều là các phép toán đại số. Tuy nhiên, trong MTT, thích hợp hơn là xem phép mã và giải mã như các hoán vị của các kí tự.

Mã thay thế

Cho $P=C=Z_{26}$. K chứa mọi hoán vị có thể của 26 kí hiệu 0,1, . . . ,25
Với mỗi phép hoán vị $\pi \in K$, 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 (cũng như trước, các ký hiệu của bản rõ được viết bằng chữ thường còn các ký hiệu của bản mã là chữ in hoa).

Như vậy, e_π (a) = X, e_π (b) = N,. . . . 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 nhận được:

Bởi vậy d_π (A) = d, d_π(B) = 1, . . .

Ví dụ: Hãy giải mã bản mã:

M G Z V Y Z L G H C M H J M Y X S S E M N H A H Y C D L M H A.

Mỗi khoá của MTT là một phép hoán vị của 26 kí tự. Số các hoán vị này là 26!, lớn hơn 4×10^{26} là một số rất lớn. Bởi vậy, phép tìm khoá vét cạn không thể thực hiện được, thậm chí bằng máy tính. Tuy nhiên, sau này sẽ thấy rằng MTT có thể dễ dàng bị thám bằng các phương pháp khác.

2.1.3. 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 mod 26

a, b $\in Z_{26}$. 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 $\in Z_{26}$, ta muốn có đồng nhất thức sau:

ax + b ≡ y (mod 26)

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

ax ≡ y-b (mod 26)

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

$ax \equiv y \pmod{26}$ $(y \in Z_{26})$.

Ta biết rằng, phương trình này có một nghiệm duy nhất đối với mỗi y khi và chỉ khi UCLN(a,26) = 1 (ở đây hàm UCLN là ước chung lớn nhất của các biến của nó). Trước tiên ta giả sử rằng, UCLN(a,26) = d >1. Khi đó, đồng dư thức $ax \equiv 0 \pmod{26}$ sẽ có ít nhất hai nghiệm phân biệt trong Z_{26} là x = 0 và x = 26/d. Trong trường hợp này, e(x) = ax + b mod 26 không phải là một hàm đơn ánh và bởi vậy nó không thể là hàm mã hoá hợp lệ.

Ví dụ, do UCLN(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 kì $x \in Z_{26}$.

Ta giả thiết UCLN(a,26) = 1. Giả sử với x_1 và x_2 nào đó thảo mãn:

$ax_1 \equiv ax_2 \pmod{26}$

Khi đó

$a(x_1 - x_2) \equiv 0 \pmod{26}$

bởi vậy

$26 \mid a(x_1 - x_2)$

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

$26 \mid (x_1 - x_2)$

tức là

$x_1 \equiv x_2 \pmod{26}$

Tới đây ta chứng tỏ rằng, nếu UCLN(a,26) = 1 thì một đồng dư thức dạng $ax \equiv y \pmod{26}$ chỉ có (nhiều nhất) một nghiệm trong Z_{26} . Do đó, nếu ta cho x thay đổi trên Z_{26} thì ax mod 26 sẽ nhận được 26 giá trị khác nhau theo modulo 26 và đồng dư thức $ax \equiv y \pmod{26}$ chỉ có một nghiệm y duy nhất.

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

Định lí

Đồng dư thức $ax \equiv b \bmod m$ chỉ có một nghiệm duy nhất $x \in Z_m$ với mọi $b \in Z_m$ khi và chỉ khi UCLN(a,m) = 1.

Vì $26 = 2 \times 13$ nên các giá trị $a \in Z_{26}$ thoả mãn UCLN(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 Z_{26} . Như vậy, mã Affine có $12 \times 26 = 312$ khoá có thể (dĩ nhiên con số này quá nhỉ 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

Giả sử $a \geq 1$ và $m \geq 2$ là các số nguyên. UCLN(a,m) = 1 thì ta nói rằng a và m là nguyên tố cùng nhau. Số các số nguyên trong Z_m nguyên tố cùng nhau với m thường được ký hiệu là $\phi(m)$ (hàm này được gọi là hàm Euler).

Một kết quả quan trọng trong lý thuyết số cho ta giá trị của $\phi(m)$ 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 $98 = 2 \times 7^2$).

Số khoá trong mã Affine trên Z_m bằng $\phi(m)$, trong đó $\phi(m)$ 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, $\phi(60)=\phi(5.2^2.3)=\phi(5).\phi(2^2).\phi(3) = 2 \times 2 \times 4 = 16$ và số các khoá trong mã Affine là 960. (xem tính chất của hàm phi euler chương 4)

Bây giờ ta sẽ xét xem các phép toán giải mã trong mật mã Affine với modulo m = 26. Giả sử UCLN(a,26) = 1. Để giải mã cần giải phương trình đồng dư $y \equiv ax+b \pmod{26}$ theo x. Từ thảo luận trên thấy rằng, phương trình này có

một nghiệm duy nhất trong 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:

Giả sử $a \in Z_m$. Phần tử nghịch đảo (theo phép nhân) của a là phần tử $a^{-1} \in Z_m$ sao cho $aa^{-1} \equiv a^{-1}a \equiv 1 \pmod m$.

Bằng các lý luận tương tự như trên, có thể chứng tỏ rằng a có nghịch đảo theo modulo m khi và chỉ khi UCLN(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 Z_P đều có nghịch đảo. Một vành trong đó mọi phần tử đều có nghịch đảo được gọi là một trường.

Trong phần sau sẽ mô tả một thuật toán hữu hiệu để tính các nghịch đảo của Z_m với m tuỳ ý. Tuy nhiên, trong 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 $y \equiv ax+b \pmod{26}$. Phương trình này tương đương với

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

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

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

Áp dụng tính kết hợp của phép nhân modulo:

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

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

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

Cho mô tả đầy đủ về mã Affine. Sau đây là một ví dụ nhỏ

Cho $P = C = Z_{26}$ và giả sử

$P = \{ (a,b) \in Z_{26} \times Z_{26} : UCLN(a,26) = 1 \}$

Với $K = (a,b) \in K$, ta định nghĩa:

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

và

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

$x,y \in Z_{26}$

Mật mãA ffine

Ví dụ:

Giả sử K = (7,3). Như đã nêu ở trên, $7^{-1} \bmod 26 = 15$. 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 Z_{26}. Ta sẽ kiểm tra liệu $d_K(e_K(x)) = x$ với mọi $x \in Z_{26}$ không? Dùng các tính toán trên 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 du 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 3 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.

2.1.4. Mã Vigenère

Trong cả hai hệ MDV và MTT (một khi khoá đã được chọn) mỗi ký tự sẽ được ánh xạ vào một ký tự duy nhất. Vì lý do đó, các hệ mật còn được gọi hệ thay thế đơn biểu. Bây giờ ta sẽ trình bày một hệ mật không phải là bộ chữ đơn, đó là hệ mã Vigenère nổi tiếng. Mật mã này lấy tên của Blaise de Vigenère sống vào thế kỷ XVI.

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

Xét một ví dụ:

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

Thiscryptosystemisnotsecure

Cho m là một số nguyên dương cố định nào đó. Định nghĩa $P = C = K = (Z_{26})^m$. 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 Z_{26}

Mật mã Vigenère

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

19	7	8	18	2	17	24	15	19	14	18	24
2	8	15	7	4	17	2	8	15	7	4	17
21	15	23	25	6	8	0	23	8	21	22	15
18	19	4	12	8	18	13	14	19	18	4	2
2	8	15	7	4	17	2	8	15	7	4	17
20	1	19	19	12	9	15	22	8	15	8	19
	20	17	4								
	2	8	15								
	22	25	19								

Bởi vậy, dãy ký tự tương ứng của xâu bản mã sẽ là:V P X Z G I A X I V W

P U B T T M J P W I Z I T W Z T

Để giải mã ta có thể dùng cùng từ khoá nhưng thay cho cộng, ta trừ cho nó

theo modulo 26.

Ta thấy rằng các từ khoá có thể với số độ dài m trong mật mã Vigenère là 26^m, bởi vậy, thậm chí với các giá trị m khá nhỏ, phương pháp tìm kiếm vét cạn cũng yêu cầu thời gian khá lớn. Ví dụ, nếu m = 5 thì không gian khoá cũng có kích thước lớn hơn $1{,}1 \times 10^7$. Lượng khoá này đã đủ lớn để ngăn ngừa việc tìm khoá bằng tay (chứ không phải dùng máy tính).

Trong hệ mật Vigenère có từ khoá độ dài m, mỗi ký tự có thể được ánh xạ vào trong m ký tự có thể có (giả sử rằng từ khoá chứa m ký tự phân biệt). Một hệ mật như vậy được gọi là hệ mật thay thế đa biểu (polyalphabetic). Nói chung, việc thám mã hệ thay thế đa biểu sẽ khó khăn hơn so việc thám mã hệ đơn biểu.

2.1.5. 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 $P = C = (Z_{26})^m$. 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_1cũng như y_2 đều là một tổ hợp tuyến tính của x_1 và x_2. Chẳng hạn, có thể lấy

$y_1 = 11x_1 + 3x_2$

$y_2 = 8x_1 + 7x_2$

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

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

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

$$(y_1, \ldots, y_m) = (x_1, \ldots, x_m) \begin{pmatrix} k_{1,1} & k_{1,2} & \ldots & k_{1,m} \\ k_{2,1} & k_{2,2} & \ldots & k_{2,m} \\ \ldots & \ldots & \ldots & \ldots \\ k_{m,1} & k_{m,2} & \ldots & k_{m,m} \end{pmatrix}$$

Nói một 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 giả mã. Bản mã được giải mã bằng công thức $y K^{-1}$.

Sau đây là một số định nghĩa về những khái niệm cần thiết lấy từ đại số tuyến tính. Nếu $A = (x_{i,j})$ là một ma trận cấp l × m và $B = (b_{1,k})$ là một ma trận

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

cấp m × n thì tích ma trận $AB = (c_{1,k})$ được định nghĩa theo công thức:

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 l × n.

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

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

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

I_m được gọi là ma trận đơn vị vì $AI_m = A$ với mọi ma trận cấp l × m và $I_mB = B$ với mọi ma trận cấp m × n. Ma trận nghịch đảo của ma trận A cấp m × m (nếu tồn tại) là ma trận A^{-1} sao cho $AA^{-1} = A^{-1}A = I_m$. 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:

$yK^{-1} = (xK)K^{-1} = x(KK^{-1}) = xI_m = x$

(Chú ý sử dụng tính chất kết hợp)

$$\begin{bmatrix} 12 & 8 \\ 3 & 7 \end{bmatrix}^{-1} = \begin{bmatrix} 8 & 18 \\ 23 & 11 \end{bmatrix}$$

Có thể thấy rằng, ma trận mã hoá ở trên có nghịch đảo trong Z_{26}: Vì

$$\begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix}\begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix} = \begin{bmatrix} 11\times7+8\times23 & 11\times18+8\times11 \\ 3\times7+7\times23 & 3\times18+7\times11 \end{bmatrix}$$

$$= \begin{bmatrix} 261 & 286 \\ 182 & 131 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(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:

Giả sử khoá K $= \begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix}$

$$K^{-1} = \begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix}$$

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

Từ các tính toán trên ta có:

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:

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

Như vậy Bob đã nhận được bản đúng.

$$(3,4) \begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix} = (9,20)$$

Cho tới lúc này ta đã chỉ ra rằng có thể thực hiện phép giải mã nếu K có

$$(11,22) \begin{bmatrix} 7 & 18 \\ 23 & 11 \end{bmatrix} = (11,24)$$

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, chúng ta chỉ quan tâm tới các ma trận K khả nghich.

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

Định thức của ma trận $A = (a_{i\,j})$ *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 mxm có thể được tính theo các phép toán hằng sơ cấp (xem một giáo trình bất kỳ về đại số tuyến tính)

Hai tính chất quan trọng của định thức là $\det I_m = 1$ và quy tắc nhân $\det(AB) = \det A \times \det B$.

Một ma trận thức 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 Z_{26}. Kết quả tương ứng là ma trận K có nghịch đảo theo modulo 26 khi và chỉ khi UCLN(det K,26) = 1.

Sau đây sẽ chứng minh ngắn gọn kết quả này.

Trước tiên, giả sử rằng UCLN(det K,26) = 1. Khi đó det K có nghịch đảo trong Z_{26} . Với $1 \le i \le m$, $1 \le j \le m$, định nghĩa $K_{i\,j}$ 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 $(-1)\ \det K_{j\,i}$ (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 Z_{26} .

Nhận xét: Công thức đối với 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 hằng sơ cấp.

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

Định lý

Giả sử $A = (a_{i,j})$ là một ma trận cấp 2×2 trên Z_{26} sao cho det $A = a_{1,1}a_{2,2} - a_{1,2}\, a_{2,1}$ có nghịch đảo. Khi đó

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

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

$$\det \begin{bmatrix} 11 & 8 \\ 3 & 7 \end{bmatrix} = 11 \times 7 - 8 \times 3 \bmod 2$$
$$= 77 - 24 \bmod 26 = 53 \bmod 26$$
$$= 1$$

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

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

Đây chính là ma trận đã có ở trên.

Bây giờ ta sẽ mô tả chính xác mật mã Hill trên Z_{26} (hình 1.6)

Mật mã HILL

Cho m là một số nguyên dương có định. Cho $P = C = (Z_{26})^m$ và cho
K = { các ma trận khả nghịch cấp m × m trên Z_{26}}
Với một khoá $K \in K$ ta xác định
$$e_K(x) = xK$$
và $$d_K(y) = yK^{-1}$$
Tất cả các phép toán được thực hiện trong Z_{26}

2.1.6. Các hệ 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:

$y = y_1y_2 \ldots = e_K(x_1)\, e_K(x_2) \ldots$

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 $z = z_1z_2 \ldots$ và dùng nó để mã hoá một xâu bản rõ $x = x_1x_2 \ldots$ theo quy tắc:

$y = y_1y_2 \ldots = e_{z1}(x_1)\, e_{z2}(x_1) \ldots$

Mã dòng hoạt động như sau. Giả sử $K \in K$ là khoá và $x = x_1x_2 \ldots$ 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:

$z_i = f_i (K, x_1, \ldots, x_{i-1})$

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

Việc giải mã xâu bản mã $y_1y_2 \ldots$ 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

Mật mã dòng là một bộ (P,C,K,L,F,E,D) thoả mãn dược các điều kiện sau:

1. *P là một tập hữu hạn các bản rõ có thể.*
2. *C là tập hữu hạn các bản mã có thể.*
3. *K là tập hữu hạn các khoá có thể (không gian khoá)*
4. *L là tập hữu hạn các bộ chữ của dòng khoá.*
5. *$F = (f_1 f_2 \ldots)$ là bộ tạo dòng khoá. Với $i \geq 1$*

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

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

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 $i \geq 1$.

Sau đây là một số dạng đặc biệt của mã dòng cùng với các ví dụ minh hoạ. Mã dòng được gọi là đồng bộ nếu dòng khoá không phụ thuộc vào xâu bản rõ, tức là nếu dòng khoá được tạo ra chỉ là hàm của khoá K. Khi đó ta coi K là một "mần" để mở rộng thành dòng khoá $z_1 z_2 \ldots$

Một hệ mã dòng được gọi là tuần hoàn với chu kỳ d nếu $z_{i+d} = z_i$ với số nguyên $i \geq 1$. Mã Vigenère với độ dài từ khoá m có thể coi là mã dòng tuần hoàn với chu kỳ m. Trong trường hợp này, khoá là $K = (k_1, \ldots k_m)$. Bản thân K sẽ tạo m phần tử đầu tiên của dòng khoá: $z_i = k_i$, $1 \leq i \leq m$. Sau đó dòng khoá sẽ tự lặp lại. Nhận thấy rằng, trong mã dòng tương ứng với mật mã Vigenère, các hàm mã và giải mã được dùng giống như các hàm mã và giải mã được dùng trong MDV:

$e_z(x) = x+z$ và $d_z(y) = y-z$

Các mã dòng thường được mô tả trong các bộ chữ nhi phân tức là $P = C = L = Z_2$. Trong trường hợp này, các phép toán mã và giải mã là phép cộng theo modulo 2.

$e_z(x) = x + z \bmod 2$ và $d_z(x) = y + z \bmod 2$.

Nếu ta coi "0" biểu thị giá trị "sai" và "1" biểu thị giá trị "đúng" trong đại số Boolean thì phép cộng theo moulo 2 sẽ ứng với phép hoặc có loại trừ. Bởi vậy phép mã (và giải mã) dễ dàng thực hiện bằng mạch cứng.

Ta xem xét một phương pháp tạo một dòng khoá (đồng bộ) khác. Giả sử bắt đầu với $(k_1, \ldots, k_m)$ và $z_i = k_i$, $1 \leq i \leq m$ (cũng giống như trước đây), tuy nhiên bây giờ ta tạo dòng khoá theo một quan hệ đệ quy tuyến tính cấp m:

$$z_{i+m} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod$$

trong đó $c_0, \ldots, c_{m-1} \in Z_2$ là các hằng số cho trước.

Nhận xét:

Phép đệ quy được nói là có bậc m vì mỗi số hạng phụ thuộc vào m số hạng đứng trước. Phép đệ quy này là tuyến tính bởi vì Z_{i+m} là một hàm tuyến tính của các số hạng đứng trước. Chú ý ta có thể lấy c_0= 1 mà không làm mất tính tổng quát. Trong trường hợp ngược lại phép đệ quy sẽ là có bậc m-1.

Ở đây khoá K gồm 2m giá trị $k_1, \ldots, k_m, c_0, \ldots, c_{m-1}$. Nếu $(k_1, \ldots, k_m)$= (0,...,0) thì dòng khoá sẽ chứa toàn các số 0. Dĩ nhiên phải tránh điều này vì khi đó bản mã sẽ đồng nhất với bản rõ. Tuy nhiên nếu chọn thích hợp các hằng số $c_0,..,c_{m-1}$ thì một véc tơ khởi đầu bất kì khác $(k_1, \ldots, k_m)$ sẽ tạo nên một dòng khoá có chu kỳ 2^m -1. Bởi vậy một khoá ngắn sẽ tạo nên một dòng khoá có chu kỳ rất lớn. Đây là một tính chất rất đáng lưu tâm vì ta sẽ thấy ở phần sau, mật mã Vigenère có thể bị thám nhờ tận dụng yếu tố dòng khoá có chu kỳ ngắn.

Sau đây là một ví dụ minh hoạ:

Ví dụ:

Giả sử m = 4 và dòng khoá được tạo bằng quy tắc:

$z_{i+4} = z_i + z_{i+1} \bmod 2$

Nếu dòng khoá bắt đầu một véc tơ bất kỳ khác với véc tơ (0,0,0,0) thì ta thu được dòng khoá có chu kỳ 15. Ví dụ bắt đầu bằng véc tơ (1,0,0,0), dòng khoá sẽ là:

1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1

Một véc tơ khởi đầu khác không bất kỳ khác sẽ tạo một hoán vị vòng (cyclic) của cùng dòng khoá.

Một hướng đáng quan tâm khác của phương pháp tạo dòng khoá hiệu quả bằng phần cứng là sử dụng bộ ghi dịch hồi tiếp tuyến tính (hay LFSR). Ta dùng một bộ ghi dịch có m tầng. Véc tơ $(k_1, \ldots, k_m)$ sẽ được dùng để khởi tạo (đặt các giá trị ban đầu) cho thanh ghi dịch. Ở mỗi đơn vị thời gian, các phép toán sau sẽ được thực hiện đồng thời.

1. k_1 được tính ra dùng làm bit tiếp theo của dòng khoá.
2. $k_2, \ldots, k_m$ sẽ được dịch một tầng về phía trái.

3. Giá trị mới của ki sẽ được tính bằng:

m-1

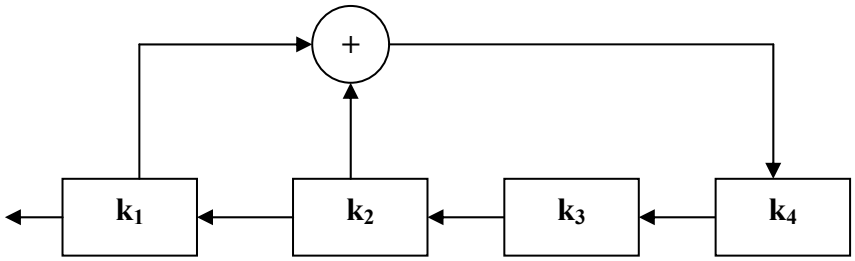
$\sum c_j k_{j+1}$

j=0

(đây là hồi tiếp tuyến tính)

Ta thấy rằng thao tác tuyến tính sẽ được tiến hành bằng cách lấy tín hiệu ra từ một số tầng nhất định của thanh ghi (được xác định bởi các hằng số c_j có giá trị "1") và tính tổng theo modulo 2 (là phép hoặc loại trừ). Mô tả của LFSR dùng để tạo dòng khoá

Thanh ghi dịch hồi tiếp tuyến tính (LFSR)



Một ví dụ về mã dòng không đồng bộ là mã khoá tự sinh như sau: (mật mã này do Vigenère đề xuất).

Mật mã khoá tự sinh

Cho $P = C = K = L = Z_{26}$
Cho $z_1 = K$ và $z_i = x_{i-1}$ $(i \geq 2)$
Với $0 \leq z \leq 25$ ta xác định
$e_z(x) = x + z \bmod 26$
$d_z(y) = y - z \bmod 26$
$(x,y \in Z_{26})$

Lý do sử dụng thuật ngữ "khoá tự sinh" là ở chỗ: bản rõ được dùng làm khoá (ngoài "khoá khởi thuỷ" ban đầu K).

Sau đây là một ví dụ minh hoạ

Giả sử khoá là k = 8 và bản rõ là *rendezvous*. Trước tiên ta biến đổi bản rõ thành dãy các số nguyên:

17 4 13 3 4 25 21 14 20 18

Dòng khoá như sau:

8 17 4 13 3 4 25 21 14 20

Bây giờ ta cộng các phần tử tương ứng rồi rút gọn theo modulo 26:

25 21 17 16 7 3 20 9 8 12

Bản mã ở dạng ký tự là: ZVRQHDUJIM

Bây giờ ta xem Alice giải mã bản mã này như thế nào. Trước tiên Alice biến đổi xâu kí tự thành dãy số:

25 21 17 16 7 3 20 9 8 12

Sau đó cô ta tính:

$x_1 = d_8(25) = 25 - 8 \bmod 26 = 17$

và $x_2 = d_{17}(21) = 21 - 17 \bmod 26 = 4$

và cứ tiếp tục như vậy. Mỗi khi Alice nhận được một ký tự của bản rõ, cô ta sẽ dùng nó làm phần tử tiếp theo của dòng khoá.

Dĩ nhiên là mã dùng khoá tự sinh là không an toàn do chỉ có 26 khoá.

Trong phần sau sẽ thảo luận các phương pháp thám các hệ mật mã mà ta đã trình bày.

2.2. Mã thám các hệ mã cổ điển

Trong phần này ta sẽ bàn tới một vài kỹ thuật mã thám. Giả thiết chung ở đây là luôn coi đối phương Oscar đã biết hệ mật đang dùng. Giả thiết này được gọi là nguyên lý Kerekhoff. Dĩ nhiên, nếu Oscar không biết hệ mật được dùng thì nhiệm vụ của anh ta sẽ khó khăn hơn. Tuy nhiên ta không muốn độ mật của một hệ mật lại dựa trên một giả thiết không chắc chắn là Oscar không biết hệ

mật được sử dụng. Do đó, mục tiêu trong thiết kế một hệ mật là phải đạt được độ mật dưới giả thiết Kerekhoff.

Trước tiên ta phân biệt các mức độ tấn công khác nhau vào các hệ mật. Sau đây là một số loại thông dụng nhất.

Chỉ có bản mã:

Thám mã chỉ có xâu bản mã y.

Bản rõ đã biết:

Thám mã có xâu bản rõ x và xâu bản mã tương ứng y.

Bản rõ được lựa chọn:

Thám mã đã nhận được quyền truy nhập tạm thời vào cơ chế mã hoá. Bởi vậy, thám mã có thể chọn một xâu bản rõ x và tạo nên xâu bản mã y tương ứng.

Bản mã được lựa chọn:

Thám mã có được quyền truy nhập tạm thời vào cơ chế giải mã. Bởi vậy thám mã có thể chọn một bản mã y và tạo nên xâu bản rõ x tương ứng.

Trong mỗi trường hợp trên, đối tượng cần phải xác định chính là khoá đã sử dụng. Rõ ràng là 4 mức tấn công trên đã được liệt kê theo độ tăng của sức mạnh tấn công. Nhận thấy rằng, tấn công theo bản mã được lựa chọn là thích hợp với các hệ mật khoá công khai mà ta sẽ nói tới ở chương sau.

Trước tiên, ta sẽ xem xét cách tấn công yếu nhất, đó là tấn công chỉ có bản mã. Giả sử rằng, xâu bản rõ là một văn bản tiếng Anh thông thường không có chấm câu hoặc khoảng trống (mã thám sẽ khó khăn hơn nếu mã cả dấu chấm câu và khoảng trống).

Có nhiều kỹ thuật thám mã sử dụng các tính chất thống kê của ngôn ngữ tiếng Anh. Nhiều tác giả đã ước lượng tần số tương đối của 26 chữ cái theo các tính toán thống kê từ nhiều tiểu thuyết, tạp chí và báo. Các ước lượng trong bảng dưới đây lấy theo tài liệu của Beker và Piper.

Xác suất xuất hiện của 26 chữ cái:

Kí tự	Xác suất	Kí tự	Xác suất	Kí tự	Xác suất
A	.082	J	.002	S	.063
B	.015	K	.008	T	.091
C	.028	L	.040	U	.028
D	.043	M	.024	V	.010
E	.0127	N	.067	W	.023
F	.022	O	.075	X	.001
G	.020	P	.019	Y	.020
H	.061	Q	.001	Z	.001
I	.070	R	.060		

Từ bảng trên, Beker và Piper phân 26 chữ cái thành 5 nhóm như sau:

1. E: có xác suất khoảng 1,120
2. T, A, O, I, N, S, H, R : mỗi ký tự có xac suất khoảng 0,06 đến 0,09
3. D, L : mỗi ký tự có xác suất chừng 0,04
4. C, U, M, W, F, G, Y, P, B: mỗi ký tự có xác suất khoảng 0,015 đến 0,023
5. V, K, J, X, Q, Z mỗi ký tự có xác suất nhỏ hơn 0,01

Việc xem xét các dãy gồm 2 hoặc 3 ký tự liên tiếp (được gọi là bộ đôi-diagrams và bộ ba – Trigrams) cũng rất hữu ích. 30 bộ đôi thông dụng nhất (theo thứ tự giảm dần) là: TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI và OF. 12 bộ ba thông dụng nhất (theo thứ tự giảm dần) là: THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR và DTH.

2.2.1. Thám hệ mã Affine

Mật mã Affine là một ví dụ đơn giản cho ta thấy cách thám hệ mã nhờ dùng các số liệu thống kê. Giả sử Oscar đã thu trộm được bản mã sau:

Bảng 1.2: Tần suất xuất hiện của 26 chữ cái của bản mã

Kí tự	Tần suất	Kí tự	Tần suất	Kí tự	Tần suất	Kí tự	Tần suất
A	2	H	5	O	1	U	2
B	1	I	0	P	3	V	4
C	0	J	0	Q	0	W	0
D	6	K	5	R	8	X	2
E	5	L	2	S	3	Y	1
F	4	M	2	T	0	Z	0
G	0	N	1				

Bản mã nhận được từ mã Affine:

FMXVEDRAPHFERBNDKRXRSREFMORUDSDKDVSHVUFEDKPK DLYEVLRHHRH

Phân tích tần suất của bản mã này được cho ở bảng dưới

Bản mã chỉ có 57 ký tự. Tuy nhiên độ dài này cũng đủ phân tích thám mã đối với hệ Affine. Các ký tự có tần suất cao nhất trong bản mã là: R (8 lần xuất hiện), D (6 lần xuất hiện), E, H, K (mỗi ký tự 5 lần) và F, S, V (mỗi ký tự 4 lần).

Trong phỏng đoán ban đầu, ta giả thiết rằng R là ký tự mã của chữ e và D là kí tự mã của t, vì e và t tương ứng là 2 chữ cái thông dụng nhất. Biểu thị bằng số ta có: $e_K(4) = 17$ và $e_K(19) = 3$. Nhớ lại rằng $e_K(x) = ax + b$ trong đó a và b là các số chưa biết. Bởi vậy ta có hai phương trình tuyến tính hai ẩn:

$4a + b = 17$

19a + b = 3

Hệ này có duy nhất nghiệm a = 6 và b = 19 (trong Z_{26}). Tuy nhiên đây là một khoá không hợp lệ do UCLN(a,26) = 2 1. Bởi vậy giả thiết của ta là không đúng. Phỏng đoán tiếp theo của ta là: R là ký tự mã của e và E là mã của t. Thực hiện như trên, ta thu được a =13 và đây cũng là một khoá không hợp lệ. Bởi vậy ta phải thử một lần nữa: ta coi rằng R là mã hoá của e và H là mã hoá của t. Điều này dẫn tới a = 8 và đây cũng là một khoá không hợp lệ. Tiếp tục, giả sử rằng R là mã hoá của e và K là mã hoá của t. Theo giả thiết này ta thu được a = 3 và b = 5 là khóa hợp lệ.

Ta sẽ tính toán hàm giải mã ứng với K = (3,5) và giải mã bản mã để xem liệu có nhận được xâu tiếng Anh có nghĩa hay không. Điều này sẽ khẳng định tính hợp lệ của khoá (3,5). □auk hi thực hiện các phép toán này, ta có d_K (y) = 9y – 19 và giải mã bản mã đã cho, ta được:

algorithmsarequitegeneraldefinitionsof

arithmeticprocesses

Như vậy khoá xác định trên là khoá đúng.

2.2.2. Thám hệ mã thay thế

Sau đây ta phân tích một tình huống phức tạp hơn, đó là thay thế bản mã sau: Ví dụ:

Bản mã nhận được từ MTT là:

YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

NZUCDRJXỷYMTMEYIFZWDYVZVYFZUMRZCRWNZDZJT

XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDINZDIR

Phân tích tần suất của bản mã này được cho ở bảng dưới đây:

Tần suất xuất hiện của 26 chữ cái trong bản mã.

Ký tự	Tần suất	Ký tự	Tần suất	Ký tự	Tần suất	Ký tự	Tần suất
A	0	H	4	O	0	U	5
B	1	I	5	P	1	V	5
C	15	J	11	Q	4	W	8
D	13	K	1	R	10	X	6
E	7	L	0	S	3	Y	10
F	11	M	16	T	2	Z	20
G	1	N	9				

Do Z xuất hiện nhiều hơn nhiều so với bất kỳ một ký tự nào khác trong bản mã nên có thể phỏng đoán rằng, $d_Z(Z) = e$. các ký tự còn lại xuất hiện ít nhất 10 lần (mỗi ký tự) là C, D, F, J, R, M, Y. Ta hy vọng rằng, các ký tự này là mã khoá của (một tập con trong) t, a, c, o, i, n, s, h, r, tuy nhiên sự khác biệt về tần suất không đủ cho ta có được sự phỏng đoán thích hợp.

Tới lúc này ta phải xem xét các bộ đôi, đặc biệt là các bộ đôi có dạng -Z hoặc Z- do ta đã giả sử rằng Z sẽ giải mã thành e. Nhận thấy rằng các bộ đôi thường gặp nhất ở dạng này là DZ và ZW (4 lần mỗi bộ); NZ và ZU (3 lần mỗi bộ); và RZ, HZ, XZ, FZ, ZR, ZV, ZC, ZD và ZJ (2 lần mỗi bộ). Vì ZW xuất hiện 4 lần còn WZ không xuất hiện lần nào và nói chung W xuất hiện ít hơn so với nhiều ký tự khác, nên ta có thể phỏng đoán là $d_K(W) = d$. Vì DZ xuất hiện 4 lần và ZD xuất hiện 2 lần nên ta có thể nghĩ rằng $d_K(D) \in \{r,s,t\}$, tuy nhiên vẫn còn chưa rõ là ký tự nào trong 3 ký tự này là ký tự đúng.

Nêu tiến hành theo giả thiết $d_K(Z) = e$ và $d_K(W) = d$ thì ta phải nhìn trở lại bản mã và thấy rằng cả hai bộ ba ZRW và RZW xuất hiện ở gần đầu của bản mã

và RW xuất hiện lại sau đó vì R thường xuất hiện trong bản mã và nd là một bộ đôi thường gặp nên ta nên thử $d_K(R) = n$ xem là một khả năng thích hợp nhất.

Tới lúc này ta có:

- - - - - - end - - - - - - - - - e - - - - ned- - - e - - - - - - - - -

YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

- - - - - - - - e- - - - e - - - - - - - - n - - d - - - en - - - - e - - - -e

NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

- e - - - n - - - - - - n - - - - - - - ed - - - e - - - - - - ne - nd- e- e - -

NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

- ed - - - - - n - - - - - - - - - - - e - - - ed - - - - - - - d - - - e - - n

XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR

Bước tiếp theo là thử $d_K(N) = h$ vì NZ là một bộ đôi thường gặp còn ZN không xuất hiện. Nếu điều này đúng thì đoạn sau của bản rõ ne - ndhe sẽ gợi ý rằng $d_K(C) = a$. Kết hợp các giả định này, ta có:

- - - - - -end- - - - - a- - -e -a - - nedh- -e- - - - - -a - - - - -

YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

h - - - - - - - a- - - e - a- - - a - - - nhad - a - -en -a - e - h- -e

NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he - a - n- - - - - - n - - - - - - ed - - - e- - - e - - neandhe -e - -

NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

- ed - a - - -nh - - - ha - - - a- e - - - - ed - - - - -a -d - - he- -n

XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR

Bây giờ ta xét tới M là ký tự thường gặp nhất sau Z. Đoạn bản mã RNM mà ta tin là sẽ giải mã thành nh- gợi ý rằng h- sẽ bắt đầu một từ, bởi vậy chắc là M sẽ biểu thị một nguyên âm. Ta đã sử dụng a và e, bởi vậy, phỏng đoán rằng $d_K(M) = i$ hoặc o. Vì ai là bộ đôi thường gặp hơn ao nên bộ đôi CM trong bản mã gợi ý rằng, trước tiên nên thử $d_K(M) = i$. Khi đó ta có:

- - - - -iend- - - - - a -i - e -a -inedhi - e- - - - - -a - - -i -

YIFQFMZRWQFYVECFMDZPCVMRZWNMDZVEJBTXCDDUMJ

h - - - - - i - ea - i - e -a - - -a - i -nhad -a - en - -a - e -hi -e

NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREXCHZUNMXZ

he - a - n - - - - -in -i - - - - ed - - -e - - - e - ineandhe - e - -

NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

- ed - a - - inhi - - hai - - a - e - i- -ed- - - - - a - d - - he - -n

XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR

Tiếp theo thử xác định xem chữ nào được mã hoá thành o. Vì o là một chữ thường gặp nên giả định rằng chữ cái tương ứng trong bản mã là một trong các ký tự D,F,J,Y. Y có vẻ thích hợp nhất, nếu không ta sẽ có các xâu dài các nguyên âm, chủ yếu là aoi (từ CFM hoặc CJM). Bởi vậy giả thiết rằng $d_K(Y) =$ o.

Ba ký tự thường gặp nhất còn lại trong bản mã là D,F,J, ta phán đoán sẽ giải mã thành r,s,t theo thứ tự nào đó. Hai lần xuất hiện của bộ ba NMD gợi ý rằng $d_K(D) =$ s ứng với bộ ba his trong bản rõ (điều này phù hợp với giả định trước kia là $d_K(D) \in \{r,s,t\}$). Đoạn HNCMF có thể là bản mã của chair, điều này sẽ cho $d_K(F) =$ r (và $d_K(H) =$ c) và bởi vậy (bằng cách loại trừ) sẽ có $d_K(J) =$ t.

Ta có:

o- r - riend - ro - - arise - a - inedhise - - t - - - ass - it

YIFQFMZRWQFYVECFMDZPCVMRZNMDZVEJBTXCDDUMJ

hs - r - riseasi - e - a - orationhadta - - en - -ace - hi - e

NDIFEFMDZCDMQZKCEYFCJMYRNCWJCSZREZCHZUNMXZ

he - asnt - oo - in - i - o - redso - e - ore - ineandhesett

NZUCDRJXYYSMRTMEYIFZWDYVZVYFZUMRZCRWNZDZJJ

- ed - ac - inhischair - aceti - ted - - to - ardsthes - n

XZWGCHSMRNMDHNCMFQCHZJMXJZWIEJYUCFWDJNZDIR

Bây giờ việc xác định bản rõ và khoá cho ở ví dụ trên không còn gì khó khăn nữa. Bản rõ hoàn chỉnh như sau:

Our friend from Pais examined his empty glass with surprise, as if evaporation had taen place while he wasn't looking. I poured some more wine and he settled back in his chair, face tilted up towards the sun.

2.2.3. Thám hệ mã Vigenère

Trong phần này chúng ta sẽ mô tả một số phương pháp thám hệ mã Vigenère. Bước đầu tiên là phải xác định độ dài từ khoá mà ta ký hiệu là m. ở đây dùng hai kỹ thuật. Kỹ thuật thứ nhất là phép thử Kasiski và kỹ thuật thứ hai sử dụng chỉ số trùng hợp.

Phép thử Kasiski lần đầu tiên được Kasiski Friendrich mô tả vào năm 1863. Kỹ thuật này được xây dựng trên nhận xét là: hai đoạn giống nhau của bản rõ sẽ được mã hoá thành cùng một bản mã khi chúng xuất hiện trong bản rõ cách nhau x vị trí, trong đó $x \equiv o \bmod m$. Ngược lại, nếu ta thấy hai đoạn giống nhau của bản mã (mỗi đoạn có độ dài ít nhất là 3) thì đó là một dấu hiệu tốt để nói rằng chúng tương ứng với các đoạn bản rõ giống nhau.

Phép thử Kasiski như sau. Ta tìm trong bản mã các cặp gồm các đoạn như nhau có độ dài tối thiểu là 3 và ghi lại khoảng cách giữa các vị trí bắt đầu của hai đoạn. Nếu thu được một vài giá trị d_1, d_2 ,. . . thì có thể hy vọng rằng m sẽ chia hết cho ước chung lớn nhất của các d_i.

Việc xác minh tiếp cho giá trị của m có thể nhận được bằng chỉ số trùng hợp. Khái niệm này đã được Wolfe Friedman đưa ra vào 1920 như sau:

Định nghĩa:

Giả sử $x = x_1x_2 \ldots x_n$ là một xâu ký tự. Chỉ số trùng hợp của x (ký hiệu là $I_c(x)$) được định nghĩa là xác suất để hai phần tử ngẫu nhiên của x là đồng nhất. Nếu ký hiệu các tần suất của A,B,C,. . . ,Z trong x tương ứng là f_0,f_1 ,. . . f_{25} , có thể chọn hai phần tử của x theo ??? cách. Với mỗi i, $0 \le i \le 25$, có ??? cách chọn hai phần tử là i.

Bây giờ, giả sử x là một xâu văn bản tiếng Anh. Ta kí hiệu các xác suất xuất hiện của các kí tự A,B,. . .,Z trong bảng 1.1 là p_0,...p_{25}. Khi đó:

do xác suất để hai phần tử ngẫu nhiên đều là A là p_0^2, xác suất để cả hai phần tử này đều bằng B bằng p_1^2 . . . Tình hình tương tự cũng xảy ra nếu x là một bản mã nhận được theo một hệ mã thay thế đơn bất kì. Trong trường hợp này, từng xác suất riêng rẽ sẽ bị hoán vị nhưng tổng ??? sẽ không thay đổi.

Bây giờ giả sử có một bản mã $y = y_1y_2. . .y_n$ được cấu trúc theo mật mã Vigenère. Ta xác định các xâu con m của $y(y_1,y_2,. . .,y_m)$ bằng cách viết ra bản mã thành một hình chữ nhật có kích thước $m\times(n/m)$. Các hàng của ma trận này là các xâu con y_i, $1 \le i \le m$. Nếu m thực sự là độ dài khoá thì mỗi $I_c(y_i)$ phải xấp xỉ bằng 0,065. Ngược lại, nếu m không phải là độ dài khoá thì các xâu con y_i sẽ có vẻ ngẫu nhiên hơn vì chúng nhận được bằng cách mã dịch vòng với các khoá khác nhau. Xét thấy rằng, một xâu hoàn toàn ngẫu nhiên sẽ có:

Hai giá trị 0,065 và 0,038 đủ cách xa nhau để có thể xác định được độ dài từ khoá đúng (hoặc xác nhận giả thuyết đã được làm theo phép thử Kasiski). Hai kỹ thuật này sẽ được minh hoạ qua ví dụ dưới đây:

Ví dụ:

Bản mã nhận được từ mật mã Vigenère.

CHEEVOAHMAERATBTAXXWTNXBEEOPHBSBQMQEQERBW
RVXUOAKXAOSXXWEAHBWGJMMQMNKGRFVGXWTRZXWIAK

LXFPSKAUTEMNDCMGTSXMXBTUIADNGMGPSRELXNJELX

RVPRTULHDNQWTWDTYGBPHXTFEALJHASVBFXNGLLCHR

ZBWELEKMSJIKNBHWRJGNMGJSGLXFEYPHAGNRBIEQJT

MRVLCRRREMNDGLXRRIMGNSNRWCHRQHAEYEVTAQEBBI

EEWEVKAKOEWADREMXMTBHHCHRTKDNVRZCHRCLQOHP

WQAIIWXNRMGWOIIFKEE

Trước tiên, ta hãy thử bằng phép thử Kasiski xâu bản mã CHR xuất hiện ở bốn vị trí trong bản mã, bắt đầu ở các vị trí 1, 166,236 và 286. Khoảng cách từ

lần xuất hiện đầu tiên tới 3 lần xuất hiện còn lại tương ứng là 165,235 và 285. UCLN của 3 số nguyên này là 5, bởi vậy giá trị này rất có thể là độ dài từ khoá.

Ta hãy xét xem liệu việc tính các chỉ số trùng hợp có cho kết luận tương tự không. Với m = 1 chỉ số trùng hợp là 0,045. Với m = 2, có 2 chỉ số là 0,046 và 0,041. Với m = 3 ta có 0,043; 0,050; 0,047. Với m = 4 các chỉ số là 0,042; 0,039; 0,046; 0,040. Với m = 5 ta có các giá trị 0,063; 0,068; 0,069; 0,061 và 0,072. Điều này càng chứng tỏ rằng độ dại từ khoá là 5.

Với giả thiết trên, làm như thế nào để xác định từ khoá? Ta sẽ sử dụng khái niệm chỉ số trùng hợp tương hỗ của hai xâu sau:

Định nghĩa.

Giả sử $x = x_1x_2. . .x_n$ và $y = y_1y_2. . .y_{n'}$ là các xâu có n và n' kí tự anphabet tương ứng. Chỉ số trùng hợp tương hỗ của x và y (kí hiệu là $MI_c(x,y)$) được xác định là xác suất để một phần tử ngẫu nhiên của x giống với một phần tử ngẫu nhiên của y. Nếu ta kí hiệu các tần suất của A,B,. . .,Z trong x và y tương ứng là $f_0,f_1,. . .,f_{25}$ .Với các giá trị m đã xác định, các xâu con y_i thu được bằng mã dịch vòng bản rõ. Giả sử $K = (k_1,k_2,. . .,k_m)$ là từ khoá. Ta sẽ xem xét có thể đánh giá $MI_c(y_i,y_j)$ như thế nào. Xét một kí tự ngẫu nhiên trong y_i và một kí tự ngẫu nhiên trong y_j . Xác suất để cả hai kí tự là A bằng $p_{-ki}\, p_{-kj}$, xác suất để cả hai là B bằng $p_{1-ki}\, p_{1-kj},. . .$(Cần chú ý rằng tất cả các chỉ số dưới đều được rút gọn theo modulo 26). Bởi vậy có thể ước lượng rằng:

$$MI_c(y_i, y_i) \approx \sum_{h=0}^{25} p_{h-ki}\, p_{h-kj} = \sum_{h=0}^{25} p_h\, p_{h+ki-kj}$$

Ta thấy rằng, giá trị ước lượng này chỉ phụ thuộc vào kiếu hiệu k_i-k_j mod 26 (được gọi là độ dịch tương đối của y_i và y_j). Cũng vậy, ta thấy rằng:

$$\sum_{h=0}^{25} p_h\, p_{h+1} = \sum_{h=0}^{25} p_h\, p_{h-1}$$

Bởi vậy độ dịch tương đối *l* sẽ dẫn đến cùng một ước lượng MI_c như độ dịch tương đối 26-*l* .

Ta lập bảng các ước lượng cho độ dịch tương đối trong phạm vi từ 0 đến 13.(Xem bảng).

Các chỉ số trùng hợp tương hỗ tính được.

Độ dịch tương đối	**Giá trị tính được của MI_c**
0	0.065
1	0,039
2	0,032
3	0,034
4	0,044
5	0,033
6	0,036
7	0,039
8	0,034
9	0,034
10	0,038
11	0,045
12	0,039
13	0,043

Xét thấy rằng, nếu độ dịch tương đối khác 0 thì các ước lượng này thay đổi trong khoảng từ 0.031 đến 0,045; ngược lại nếu độ dịch tương đối bằng 0 thì ước lượng bằng 0,065. Có thể dùng nhận xét này để tạo nên một phỏng đoán thích hợp cho $l = k_i - k_j$ (độ dịch tương đối của y_i và y_j) như sau: Giả sử cố định y_i

$$MI_c(x, y^g) = \frac{\sum_{i=0}^{25} f_i f'_{i-g}}{n.n'}$$

và xét việc mã hoá y_j bảng e_0,e_1,e_2. . . Ta kí hiệu các kết quả bằng y_j^0,y_j^1,. . . Dễ dàng dùng các chỉ số $MI_c(y_i,y_j^g)$, $0 \le g \le 25$ theo công thức sau:

Khi g = *l* thì MI_c phải gần với giá trị 0,065 vì độ dịch tương đối của y_i và y_j bằng 0. Tuy nhiên, với các giá trị $g \ne l$ thì MI_c sẽ thay đổi giữa 0,031 và 0,045.

Bằng kỹ thuật này, có thể thu được các độ dịch tương đối của hai xâu con y_i bất kỳ. Vấn đề còn lại chỉ là 26 từ khoá có thể và điều này dễ dàng tìm được bằng phương pháp tìm kiếm vét cạn.

Trở lại ví dụ trên để minh hoạ.

Ở trên đã giả định rằng, độ dài từ khoá là 5. Bây giờ ta sẽ thử tính các độ dịch tương đối. Nhờ máy tính, dễ dàng tính 260 giá trị $MI_c(y_i,y_j^g)$, trong đó $1 \le i \le j \le 5$; $0 \le g \le 25$. Các giá trị này được cho trên bảng. Với mỗi cặp (i,j), ta tìm các giá trị của $MI_c(y_i,y_j^g)$ nào gần với 0,065. Nếu có một giá trị duy nhất như vậy (Đối với mỗi cặp (i,j) cho trước), thì có thể phán đoán đó chính là giá trị độ dịch tương đối.

Trong bảng dưới có 6 giá trị như vậy được đóng khung. Chúng chứng tỏ khá rõ ràng là độ dịch tương đối của y_1 và y_2 bằng 9; độ dịch tương đối của y_2 và y_3 bằng 13; độ dịch tương đối của y_2 và y_5 bằng 7; độ dịch tương đối của y_3 và y_5 bằng 20; của y_4 và y_5 bằng 11. Từ đây có các phương trình theo 5 ẩn số K_1, K_2, K_3, K_4, K_5 như sau:

$K_1 - K_2 = 9$

$K_1 - K_2 = 16$

$K_2 - K_3 = 13$

$K_2 - K_5 = 17$

$K_3 - K_5 = 20$

$K_4 - K_5 = 11$

Điều này cho phép biểu thị các K_i theo K_1 ;

$K_2 = K_1 + 17$

$K_3 = K_1 + 4$

$K_4 = K_1 + 21$

$K_5 = K_1 + 10$

Như vậy khoá có khả năng là (K_1, K_1+17, K_1+4, K_1+21, K_1+10) với giá trị K_1 nào đó $\in Z_{26}$. Từ đây ta hy vọng rằng, từ khoá là một dịch vòng nào đó của AREVK. Bây giờ, không tốn nhiều công sức lắm cũng có thể xác định được từ khoá là JANET. Giải mã bản mã theo khoá này, ta thu được bản rõ sau:

The almond tree was in tentative blossom. The days were longer often ending with magnificient evenings of corrugated pink skies. The hunting seasun was over, with hounds and guns put away for six months. The vineyards were busy again as the well-organized farmers treated their vinesand the more lackadaisical neighbors hurried to do the pruning they have done in November.

. Các chỉ số trùng hợp tương hỗ quan sát được.

		Giá trị của $MI_c(y_j,y_j^g)$
		0,028 0,027 0,028 0,034 0,039 0,037 0,026 0,025 0,052 0,068 0,044 0,026 0,037 0,043 0,037 0,043 0,037 0,028 0,041 0,041 0,041 0,034 0,037 0,051 0,045 0,042 0,036
		0,039 0,033 0,040 0,034 0,028 0,053 0,048 0,033 0,029 0,056 0,050 0,045 0,039 0,040 0,036 0,037 0,032 0,027 0,037 0,047 0,032 0,027 0,039 0,037 0,039 0,035

		0,034 0,043 0,025 0,027 0,038 0,049 0,040 0,032 0,029 0,034 0,039 0,044 0,044 0,034 0,039 0,045 0,044 0,037 0,055 0,047 0,032 0,027 0,039 0,037 0,039 0,035
		0,043 0,033 0,028 0,046 0,043 0,044 0,039 0,031 0,026 0,030 0,036 0,040 0,041 0,024 0,019 0,048 **0,070** 0,044 0,028 0,038 0,044 0,043 0,047 0,033 0,026
		0,046 0,048 0,041 0,032 0,036 0,035 0,036 0,020 0,024 0,039 0,034 0,029 0,040 **0,067** 0,061 0,033 0,037 0,045 0,033 0,033 0,027 0,033 0,045 0,052 0,042 0,030
		0,046 0,034 0,043 0,044 0,034 0,031 0,040 0,045 0,040 0,048 0,044 0,033 0,024 0,028 0,042 0,039 0,026 0,034 0,050 0,035 0,032 0,040 0,056 0,043 0,028 0,028
		0,033 0,033 0,036 0,046 0,026 0,018 0,043 **0,080** 0,050 0,029 0,031 0,045 0,039 0,037 0,027

		0,026 0,031 0,039 0,040 0,037 0,041 0,046 0,045 0,043 0,035 0,030
		0,038 0,036 0,040 0,033 0,036 0,060 0,035 0,041 0,029 0,058 0,035 0,035 0,034 0,053 0,030 0,032 0,035 0,036 0,036 0,028 0,043 0,032 0,051 0,032 0,034 0,030
		0,035 0,038 0,034 0,036 0,030 0,043 0,043 0,050 0,025 0,041 0,051 0,050 0,035 0,032 0,033 0,033 0,052 0,031 0,027 0,030 **0,072** 0,035 0,034 0,032 0,043 0,027
		0,052 0,038 0,033 0,038 0,041 0,043 0,037 0,048 0,028 0,028 0,036 **0,061** 0,033 0,033 0,032 0,052 0,034 0,027 0,039 0,043 0,033 0,027 0,030 0,039 0,048 0,035

2.2.4.Tấn công với bản rõ đã biết trên hệ mật Hill.

Hệ mã Hill là một hệ mật khó pha hơn nếu tấn công chỉ với bản mã. Tuy nhiên hệ mật này dễ bị phá nếu tấn công bằng bản rõ đã biết. Trước tiên, giả sử rằng, thám mã đã biết được giá trị m đang sử dụng. Giả sử thám mã có ít nhất m cặp véc tơ khác nhau xj = ($x_{1,j}$, $x_{2,j}$, , . . ., $x_{m,j}$) và y_j = ($y_{1,j}$, $y_{2,j}$,...,$y_{m,j}$) ($1 \le j \le m$)

sao cho $y_j = e_K(x_j)$, $1 \le j \le m$. Nếu xác định hai ma trận: $X = (x_{i,j})$ $Y = (y_{i,j})$ cấp $m \times m$ thì ta có phương trình ma trận $Y = XK$, trong đó ma trận K cấp $m \times m$ là khoá chưa biết. Với điều kiện ma trận Y là khả nghịch. Oscar có thể tính $K = X^{-1}Y$ và nhờ vậy phá được hệ mật. (Nếu Y không khả nghịch thì cần phải thử các tập khác gồm m cặp rõ - mã).

Ví dụ

Giả sử bản rõ *Friday* được mã hoá bằng mã Hill với m = 2, bản mã nhận được là PQCFKU.

Ta có $e_K(5,17) = (15,16)$, $e_K(8,3) = (2,5)$ và $e_K(0,24) = (10,20)$. Từ hai cặp rõ - mã đầu tiên, ta nhận được phương trình ma trận:

$$\begin{pmatrix} 15 & 16 \\ 2 & 5 \end{pmatrix} = \begin{pmatrix} 5 & 17 \\ 8 & 3 \end{pmatrix} K$$

Dùng định lý dễ dàng tính được:

$$\begin{pmatrix} 5 & 17 \\ 8 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 9 & 1 \\ 2 & 15 \end{pmatrix}$$

Bởi vậy:

$$K = \begin{pmatrix} 9 & 1 \\ 2 & 15 \end{pmatrix} \begin{pmatrix} 15 & 16 \\ 2 & 5 \end{pmatrix} = \begin{pmatrix} 7 & 19 \\ 8 & 3 \end{pmatrix}$$

Ta có thể dùng cặp rõ - mã thứ 3 để kiểm tra kết quả này.

Vấn đề ở đây là thám mã phải làm gì nếu không biết m?. Giả sử rằng m không quá lớn, khi đó thám má có thể thử với m = 2,3,. . . cho tới khi tìm được khoá. Nếu một giá trị giả định của m không đúng thì mà trận $m \times m$ tìm được theo thuật toán đã mô tả ở trên sẽ không tương thích với các cặp rõ - mã khác. Phương pháp này, có thể xác định giá trị m nếu chưa biết.

2.2.5. Thám mã hệ mã dòng xây dựng trên LFSR.

Ta nhớ lại rằng, bản mã là tổng theo modulo 2 của bản rõ và dòng khoá, tức $y_i = x_i + z_i \bmod 2$. Dòng khóa được tạo từ $(z_1,z_2,. . .,z_m)$ theo quan hệ đệ quy tuyến tính:

$$z_{m+1} = \sum_{j=0}^{m-1} c_j z_{i+1} \bmod 2$$

trong đó $c_0, \ldots, c_m \in Z_2$ (và $c_0 = 1$)

Vì tất cả các phép toán này là tuyến tính nên có thể hy vọng rằng, hệ mật này có thể bị phá theo phương pháp tấn công với bản rõ đã biết như trường hợp mật mã Hill. Giả sử rằng, Oscar có một xâu bản rõ $x_1x_2 \ldots x_n$ và xâu bản mã tương ứng $y_1y_2 \ldots y_n$. Sau đó anh ta tính các bít dòng khoá $z_i = x_i+y_i \bmod 2$, $1 \le i \le n$. Ta cũng giả thiết rằng Oscar cũng đã biết giá trị của m. Khi đó Oscar chỉ cần tính $c_0, \ldots, c_{m-1}$ để có thể tái tạo lại toàn bộ dòng khoá. Nói cách khác, Oscar cần phải có khả năng để xác định các giá trị của m ẩn số.

Với $i \ge 1$ bất kì ta có :

$$z_{m+1} = \sum_{j=0}^{m-1} c_j z_{i+j} \bmod 2$$

là một phương trình tuyến tính n ẩn. Nếu $n \ge 2n$ thì có m phương trình tuyến tính m ẩn có thể giải được.

Hệ m phương trình tuyến tính có thể viết dưới dạng ma trận như sau:

$$(z_{m+1}, z_{m+2}, \ldots, z_{2m}) = (c_0, c_1, \ldots, c_{m-1}) \begin{bmatrix} z_1 & z_2 & \ldots & z_m \\ z_2 & z_3 & \ldots & z_{m+1} \\ \ldots & \ldots & \ldots & \ldots \\ z_m & z_{m+1} & \ldots & z_{2m-1} \end{bmatrix}$$

Nếu ma trận hệ số có nghịch đảo (theo modulo 2)thì ta nhận được nghiệm:

$$(c_0, c_1, \ldots, c_{m-1}) = (z_{m+1}, z_{m+2}, \ldots, z_{2m}) \begin{bmatrix} z_1 & z_2 & \ldots & z_m \\ z_2 & z_3 & \ldots & z_{m+1} \\ \ldots & \ldots & \ldots & \ldots \\ z_m & z_{m+1} & \ldots & z_{2m-1} \end{bmatrix}^{-1}$$

Trên thực tế, ma trận sẽ có nghịch đảo nếu bậc của phép đệ quy được dùng để tạo dòng khoá là m.(xem bài tập). Minh hoạ điều này qua một ví dụ.

Ví dụ :

Giả sử Oscar thu được xâu bản mã

101101011110010

tương ứng với xâu bản rõ

011001111111001

Khi đó anh ta có thể tính được các bít của dòng khoá:

110100100001010

Ta cũng giả sử rằng, Oscar biết dòng khoá được tạo từ một thanh ghi dịch phản hồi (LFSR) có 5 tầng. Khi đó, anh ta sẽ giải phương trình mà trận sau (nhận được từ 10 bít đầu tiên của dòng khoá):

$$(0,1,0,0,0) = (c_0, c_1, c_2, c_3, c_4)\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Có thể kiểm tra được rằng:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Từ đó ta có:

$$(c_0, c_1, c_2, c_3, c_4) = (0,1,0,0,0)\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$= (1, 0, 0, 1, 0)$$

Như vậy phép đệ quy được dùng để tạo dòng khoá là:

$z_{i+5} = z_i + z_{i+3} \bmod 2$

Các chú giải và tài liệu dẫn

Nhiều tài liệu về mật mã cổ điển đã có trong các giáo trình, chẳng hạn như giáo trình của Beker và Piper [BP82] và Denning [DE82]. Xác suất đánh giá cho 26 kí tự được lấy của Beker và Piper. Cũng vậy, việc phân tích mã Vigenère được sửa đổi theo mô tả của Beker và Piper. Rosen [Ro93] là một tài liệu tham khảo tốt về lý thuyết số. Cơ sở của Đại số tuyến tính sơ cấp có thể tìm thấy trong sách của Anton [AN91]. Cuốn " Những người mã thám " của Kahn [KA67] là một cấu chuyện hấp dẫn và phong phú về mật mã cho tới năm 1967, trong đó Kahn khẳng định rằng mật mã Vigenère thực sự không phải là phát minh của Vigenère.

Mật mã Hill lần đầu tiên được mô tả trong [HI29]. Các thông tin về mật mã dòng có thể tìm được trong sách của Rueppel [RU86].

Chương 3: Chuẩn mã dữ liệu DES
(Data Encryption Standard)

3.1. Giới thiệu chung về DES

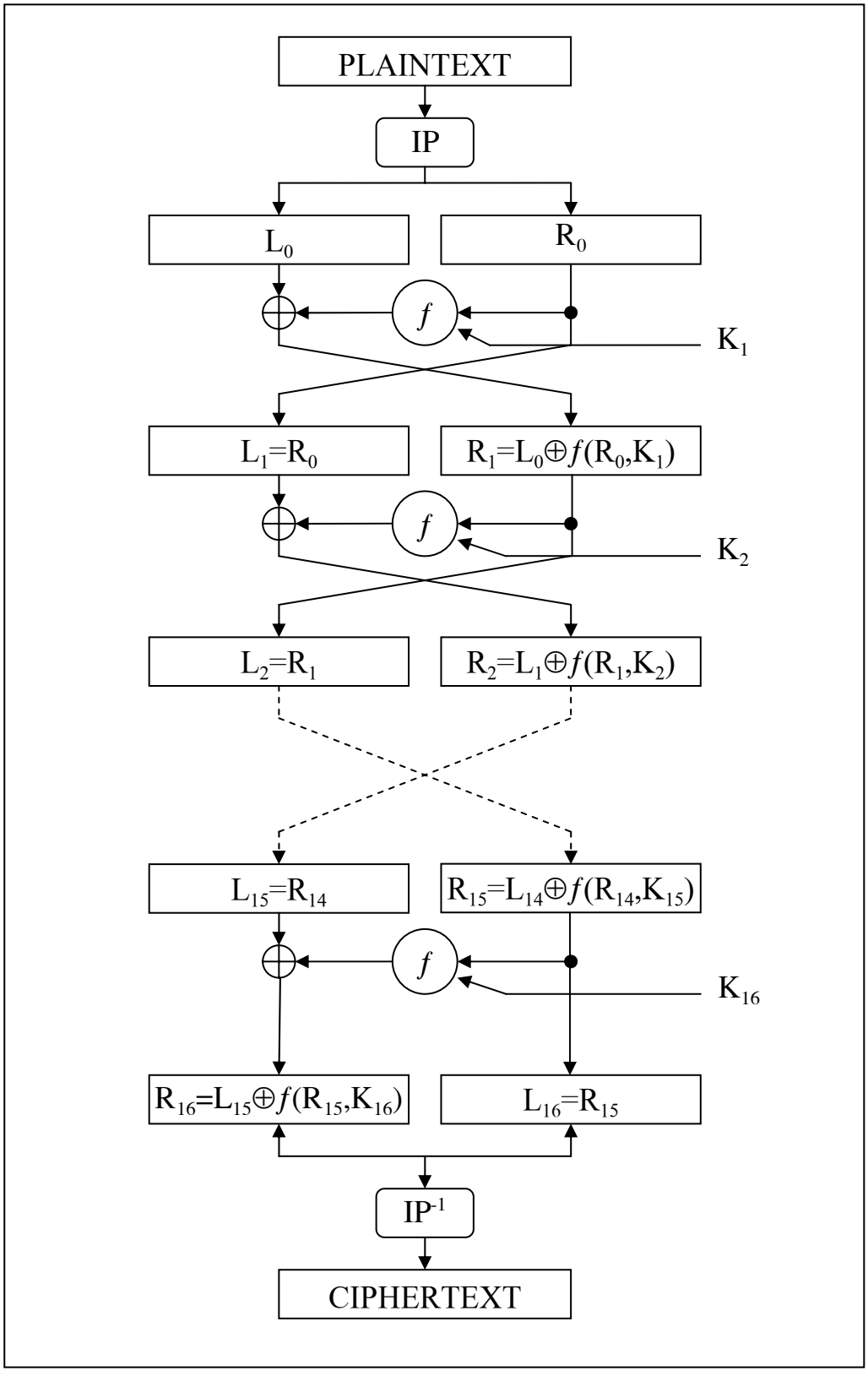
Chuẩn mã hoá dữ liệu DES được Văn phòng tiêu chuẩn của Mỹ (U.S National Bureau for Standards) công bố năm 1971 để sử dụng trong các cơ quan chính phủ liên bang. Giải thuật được phát triển tại Công ty IBM dựa trên hệ mã hoá LUCIFER của Feistel.

DES là thuật toán mã hoá khối (block algrithm), với cỡ của một khối là 64 bít. Một khối 64 bít bản rõ được đưa vào, sau khi mã hoá dữ liệu đưa ra là một khối bản mã 64 bít. Cả mã hoá và giải mã đều sử dụng cùng một thuật toán và khoá.

Khoá mã có độ dài 64 bít, trong đó có 8 bít chẵn lẻ được sử dụng để kiểm soát lỗi. Các bít chẵn lẻ nằm ở các vị trí 8, 16, 24,... , 64. Tức là cứ 8 bít khoá thì trong đó có 1 bít kiểm soát lỗi, bít này qui định số bít có giá trị "1" của khối 8 bít đó theo tính bù chẵn.

Nền tảng để xây dựng khối của DES là sự kết hợp đơn giản của các kỹ thuật thay thế và hoán vị bản rõ dựa trên khoá. Đó là các vòng lặp. DES sử dụng 16 vòng lặp, nó áp dụng cùng một kiểu kết hợp của các kỹ thuật trên khối bản rõ 16 lần (Như hình vẽ)

Thuật toán chỉ sử dụng các phép toán số học và lôgíc trên các số 64 bít, vì vậy nó dễ dàng thực hiện vào những năm 1970 trong điều kiện về công nghệ phần cứng lúc bấy giờ. Ban đầu, sự thực hiện các phần mềm kiểu này rất thô sơ, nhưng hiện tại thì việc đó đã tốt hơn, và với đặc tính lặp đi lặp lại của thuật toán đã tạo nên ý tưởng sử dụng chíp với mục đích đặc biệt này.


PLAINTEXT
IP
L_0
R_0
f
K_1
$L_1=R_0$
$R_1=L_0 \oplus f(R_0,K_1)$
f
K_2
$L_2=R_1$
$R_2=L_1 \oplus f(R_1,K_2)$
$L_{15}=R_{14}$
$R_{15}=L_{14} \oplus f(R_{14},K_{15})$
f
K_{16}
$R_{16}=L_{15} \oplus f(R_{15},K_{16})$
$L_{16}=R_{15}$
IP^{-1}
CIPHERTEXT

Sơ đồ mã DES

Tóm lại DES có một số đặc điểm sau:

- Sử dụng khoá 56 bít.
- Xử lý khối vào 64 bít, biến đổi khối vào thành khối ra 64 bít.
- Mã hoá và giải mã được sử dụng cùng một khoá.
- DES được thiết kế để chạy trên phần cứng.

DES thường được sử dụng để mã hoá các dòng dữ liệu mạng và mã hoá dữ liệu được lưu trữ trên đĩa.

3.2. Mô tả thuật toán

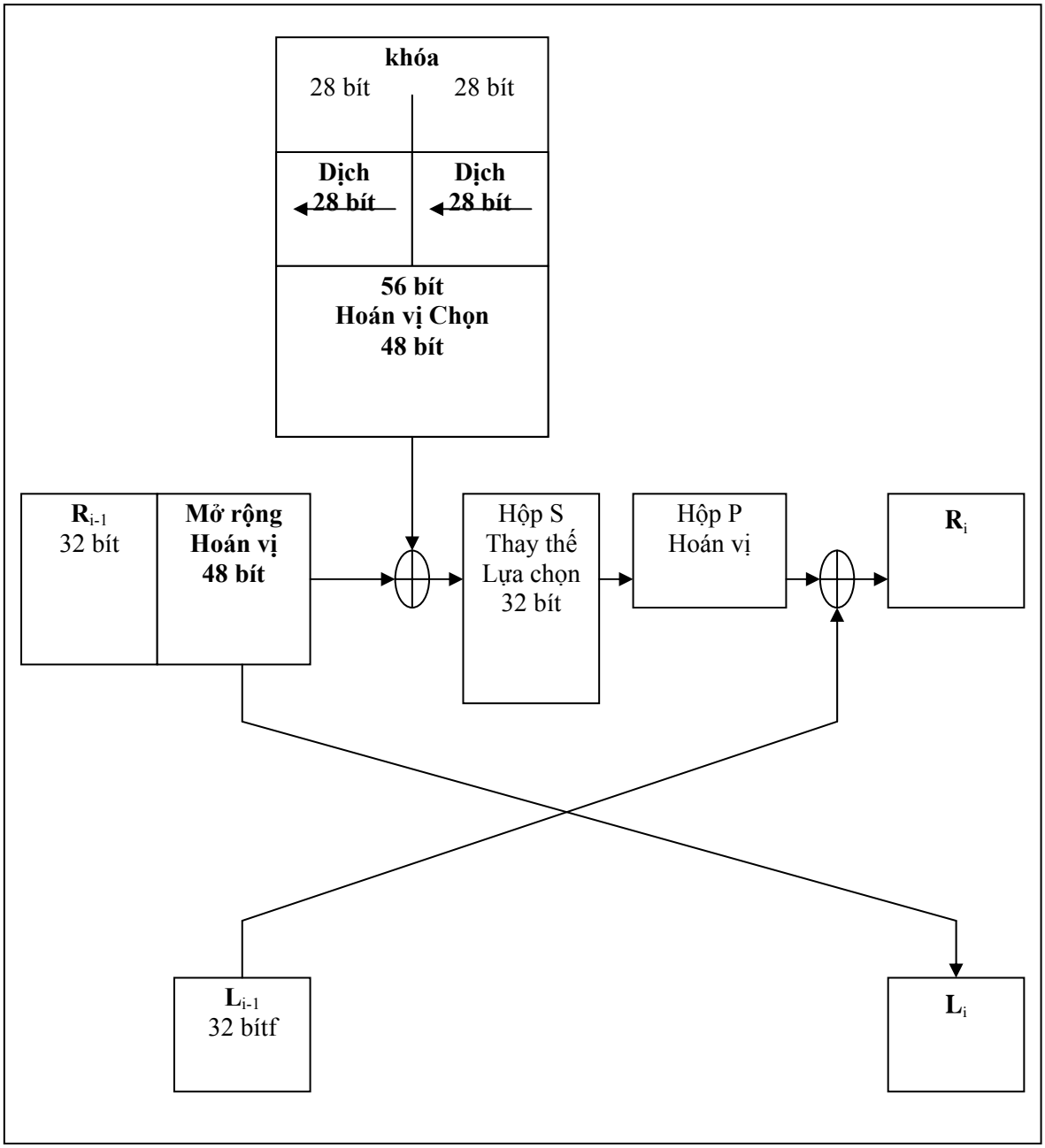
DES thực hiện trên từng khối 64 bít bản rõ. Sau khi thực hiện hoán vị khởi đầu, khối dữ liệu được chia làm hai nửa trái và phải, mỗi nửa 32 bít. Tiếp đó, có 16 vòng lặp giống hệt nhau được thực hiện, được gọi là các hàm *f*, trong đó dữ liệu được kết hợp với khoá. Sau 16 vòng lặp, hai nửa trái và phải được kết hợp lại và hoán vị cuối cùng (hoán vị ngược) sẽ kết thúc thuật toán.

Trong mỗi vòng lặp, các bít của khoá được dịch đi và có 48 bít được chọn ra từ 56 bít của khoá. Nửa phải của dữ liệu được mở rộng thành 48 bít bằng một phép hoán vị mở rộng, tiếp đó khối 48 bít này được kết hợp với khối 48 bít đã được thay đổi và hoán vị của khoá bằng toán tử XOR. Khối kết quả của phép tính XOR được lựa chọn ra 32 bít bằng cách sử dụng thuật toán thay thế và hoán vị lần nữa. Đó là bốn thao tác tạo nên hàm *f*. Tiếp đó, đầu ra của hàm *f* được kết hợp với nửa trái bằng một toán tử XOR. Kết quả của các bước thực hiện này trở thành nửa phải mới; nửa phải cũ trở thành nửa trái mới. Sự thực hiện này được lặp lại 16 lần, tạo thành 16 vòng của DES (Hình 10).

Nếu B_i là kết quả của vòng thứ i, L_i và R_i là hai nửa trái và phải của B_i, K_i là khoá 48 bít của vòng thứ i, và *f* là hàm thực hiện thay thế, hoán vị và XOR với khoá, ta có biểu diễn của một vòng sẽ như sau:

$$L_i=R_{i-1}$$

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



Một vòng lặp DES

3.3.Hoán vị khởi đầu

Hoán vị khởi đầu đổi chỗ khối dữ liệu vào, thay đổi vị trí của các bít trong khối dữ liệu vào, như được mô tả trong Bảng 1. Bảng này, và tất cả các bảng khác sau này, được đọc từ trái qua phải, từ trên xuống dưới. Ví dụ, hoán vị khởi đầu chuyển bít 1 thành bít 58, bít 2 thành bít 50, bít 3 thành bít 42,...

Bảng 1. Hoán vị khởi đầu.

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

Hoán vị khởi đầu và tương ứng là hoán vị ngược không làm ảnh hưởng đến sự an toàn của DES.

3.4. Khoá chuyển đổi

Đầu tiên, khoá 64 bít được giảm xuống thành một khoá 56 bít bằng cách bỏ qua 8 bít chẵn lẻ. Sự loại bỏ được thực hiện theo Bảng sau:

Bảng khoá chuyển đổi:

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

Các bít chẵn lẻ này có thể được sử dụng để đảm bảo rằng không có lỗi nào xảy ra khi đưa khoá vào. Sau khi khoá 56 bít được trích ra, một khoá khác 48 bít được sinh ra cho mỗi vòng của DES. Những khoá này, k_i, được xác định bằng cách:

+ Đầu tiên, khoá 56 bít được chia làm hai phần mỗi phần 28 bít. Sau đó, các phần này được dịch trái một hoặc hai bít, phụ thuộc vào vòng đó. Số bít được dịch được cho trong Bảng sau:

Bảng số bít dịch của một vòng

Vòng	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Số bít dịch	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

+ Sau khi được dịch, 48 bít được lựa chọn ra từ 56 bít. Bởi vì sự thực hiện này đổi chỗ thứ tự các bít như là sự lựa chọn một tập con các bít, nó được gọi là hoán vị nén (compression permutation), hoặc hoán vị lựa chọn (permuted choice). Sự thực hiện này cung cấp một tập hợp các bít cùng cỡ với đầu ra của hoán vị mở rộng. Bảng 4 định nghĩa hoán vị nén (cũng gọi là hoán

vị lựa chọn). Ví dụ, bít ở vị trí 33 của khoá dịch được chuyển tới vị trí 35 của đầu ra, và bít ở vị trí 18 của khoá dịch bị bỏ qua.

Bảng hoán vị nén:

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

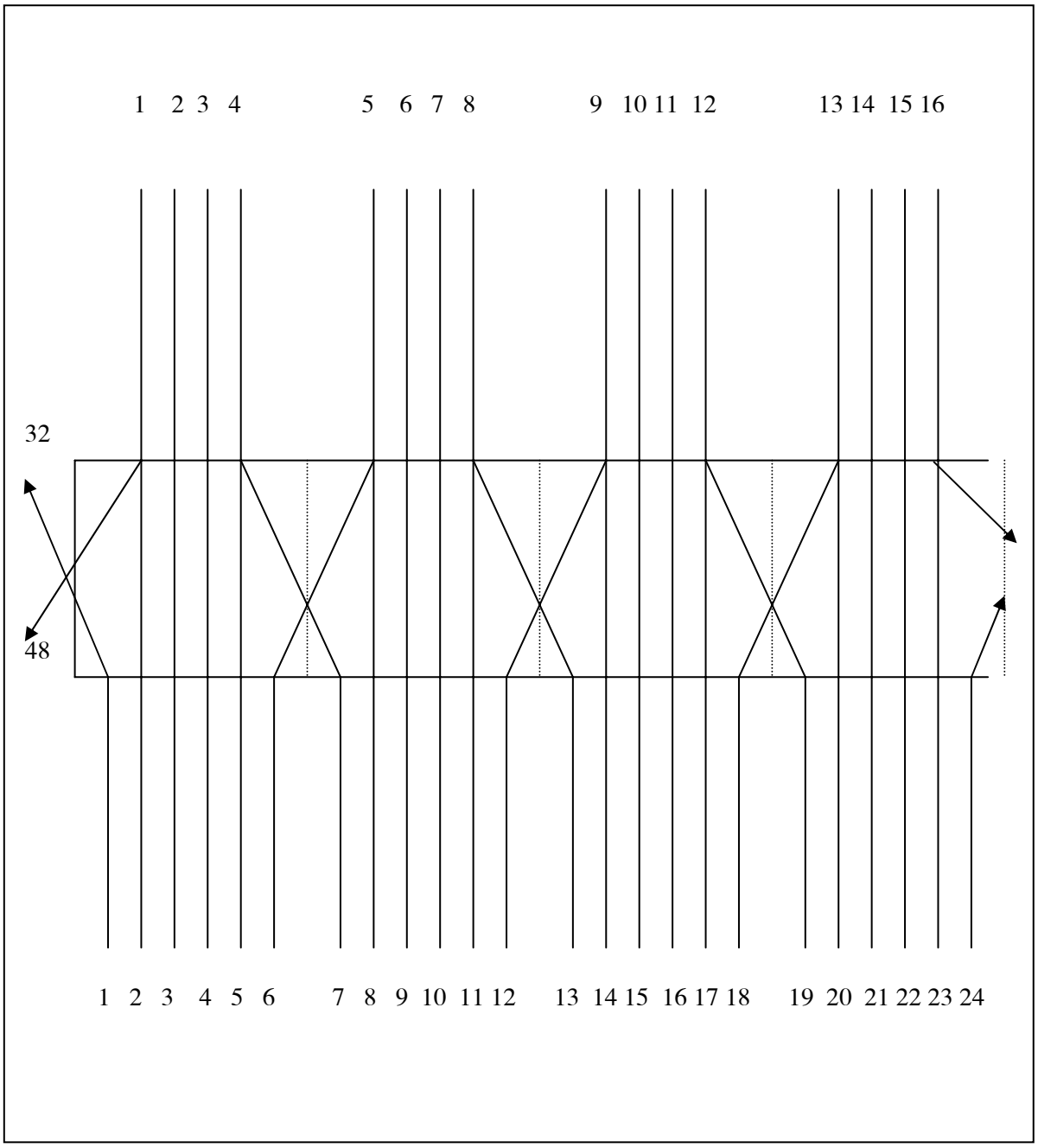
3.5. Hoán vị mở rộng

Ở thao tác này, nửa phải của dữ liệu, R_i, được mở rộng từ 32 bít thành 48 bít. Bởi vì sự thực hiện này thay đổi thứ tự của các bít bằng cách lặp lại một bít nào đó, nó được hiểu như là một sự hoán vị mở rộng. Sự thực hiện này nhằm mục đích tạo ra kết quả là dữ liệu cùng cỡ với khoá để thực hiện thao tác XOR.

Định nghĩa hoán vị mở rộng - hộp E. Với mỗi bộ 4 bít của khối dữ liệu vào, bít đầu tiên và bít thứ tư mỗi bít tương ứng với 2 bít của khối dữ liệu ra, trong khi bít thứ hai và bít thứ ba mỗi bít tương ứng với một bít của khối dữ liệu ra. Bảng dưới mô tả vị trí của các bít trong khối dữ liệu ra theo khối dữ liệu vào. Ví dụ, bít ở vị trí thứ 3 của khối dữ liệu vào được chuyển tới vị trí thứ 4 trong khối dữ liệu ra. Và bít ở vị trí 21 của khối dữ liệu vào được chuyển tới vị trí 30 và 32 trong khối dữ liệu ra.

Bảng hoán vị mở rộng E:

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	12	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



Hoán vị mở rộng

Mặc dù khối dữ liệu ra rộng hơn khối dữ liệu vào, nhưng một khối dữ liệu vào chỉ có duy nhất một khối dữ liệu ra.

3.6. Hộp thay thế S

Sau khi được nén, khoá được XOR với khối mở rộng, 48 bít kết quả được chuyển sang giai đoạn thay thế. Sự thay thế được thực hiện bởi 8 hộp thay thế (substitution boxes, S-boxes). Khối 48 bít được chia thành 8 khối 6 bít. Mỗi khối được thực hiện trên một hộp S riêng biệt (separate S-box): khối 1 được thực hiện trên hộp S_1, khối 2 được thực hiện trên hộp S_2,... , khối 8 được thực hiện trên hộp S_8.

Mỗi hộp S là một bảng gồm 4 hàng và 16 cột. Mỗi phần tử của hộp là một số 4 bít. Sáu bít vào hộp S sẽ xác định số hàng và số cột để tìm kết quả ra. Bảng 6 biểu diễn 8 hộp S.

Những bít vào xác định một phần tử trong hộp S một cách riêng biệt. Sáu bít vào của hộp được ký hiệu là b1, b2, b3, b4, b5 và b6. Bít b1 và b6 được kết hợp thành một số 2 bít, nhận giá trị từ 0 đến 3, tương ứng với một hàng trong bảng. Bốn bít ở giữa, từ b2 tới b5, được kết hợp thành một số 4 bít, nhận giá trị từ 0 đến 15, tương ứng với một cột trong bảng.

Ví dụ, giả sử ta đưa dữ liệu vào hộp S thứ 6 (bít 31 tới bít 36 của hàm XOR) là 110010. Bít đầu tiên và bít cuối cùng kết hợp thành 10, tương ứng với hàng thứ 3 của hộp S thứ 6. Bốn bít giữa kết hợp thành 1001, tương ứng với cột thứ 10 của hộp S thứ 6. Phần tử hàng 3 cột 9 của hộp S thứ 6 là 0. Giá trị 0000 được thay thế cho 110010.

Kết quả của sự thay thế là 8 khối 4 bít, và chúng được kết hợp lại thành một khối 32 bít. Khối này được chuyển tới bước tiếp theo: hộp hoán vị P (P-box permutation).

Hộp S thứ nhất

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

Hộp S thứ 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

Hộp S thứ 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

Hộp S thứ 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

Hộp S thứ 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

Hộp S thứ 6

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

Hộp S thứ 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

Hộp S thứ 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

3.7. Hộp hoán vị P

Khối dữ liệu 32 bít ra của hộp thay thế S được hoán vị tiếp trong hộp P. Sự hoán vị này ánh xạ mỗi bít dữ liệu vào tới một vị trí trong khối dữ liệu ra; không bít nào được sử dụng hai lần và cũng không bít nào bị bỏ qua. Nó được gọi là hoán vị trực tiếp (straight permutation). Bảng hoán vị cho ta vị trí của mỗi bí cần chuyển. Ví dụ, bít 4 chuyển tới bít 21, trong khi bít 32 chuyển tới bít 4.

Bảng hộp hoán vị P

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

Cuối cùng, kết quả của hộp hoá vị P được XOR với nửa trái của khối 64 bít khởi đầu. Sau đó, nửa trái và phải được chuyển đổi cho nhau và một vòng mới được tiếp tục.

3.8. Hoán vị cuối cùng

Hoán vị cuối cùng là nghịch đảo của hoán vị khởi đầu, và nó được mô tả trong bảng dưới. Chú ý rằng nửa trái và nửa phải không được tráo đổi sau vòng cuối cùng của DES; thay vào đó khối nối $R_{16}L_{16}$ được sử dụng như khối dữ liệu ra của hoán vị cuối cùng. Không có gì đưa ra ở đây; tráo đổi các nửa và dịch vòng hoán vị sẽ cho chính xác như kết quả trước; điều đó có nghĩa là thuật toán có thể được sử dụng cho cả mã hoá và giải mã.

Bảng hoán vị cuối cùng:

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

3.9. Giải mã DES

Sau khi thay đổi, hoán vị, XOR, và dịch vòng, chúng ta có thể nghĩ rằng thuật toán giải mã phức tạp, khó hiểu như thuật toán mã hoá và hoàn toàn khác thuật toán mã hoá. Trái lại, sự hoạt động được lựa chọn để đưa ra một đặc tính hữu ích: cùng thuật toán làm việc cho cả mã hoá và giải mã.

Với DES, có thể sử dụng cùng chức năng để giải mã hoặc mã hoá một khối. Chỉ có sự khác nhau đó là các khoá phải được sử dụng theo thứ tự ngược lại. Nghĩa là, nếu các khoá mã hoá cho mỗi vòng là k_1, k_2, k_3 ,... , k_{15}, k_{16} thì các khoá giải là k_{16}, k_{15},... , k_3, k_2, k_1. Giải thuật để tổng hợp khoá cho mỗi vòng cũng tương tự. Có khác là các khoá được dịch phải và số vị trí bit để dịch được lấy theo chiều ngược lại.

3.10. Phần cứng và phần mềm thực hiện DES

Việc mô tả DES khá dài dòng song việc thực hiện DES rất hữu hiệu bằng cả phần cứng lẫn phần mềm. Các phép tính số học duy nhất được thực hiện là phép XOR các xâu bít. Hàm mở rộng E, các hộp S, các hoán vị khởi đầu IP, hoán vị cuối cùng IP^{-1} và việc tính toán các khoá k_1, k_2,... , k_{16} đều có thể thực hiện được cùng lúc bằng tra bảng (trong phần mềm) hoặc bằng cách nối cứng chúng thành mạch.

Một phần mềm DES trên máy tính lớn IBM 3090 có thể thực hiện 32.000 phép tính mã hoá trong một giây. Với máy vi tính thì tốc độ thấp hơn. Bảng 9 đưa ra kết quả thực tế và sự đánh giá cho bộ xử lý của Intel và Motorola.

Bảng 9. Tốc độ của DES trên các bộ vi xử lý khác nhau

Bộ vi xử lý	Tốc độ (Mhz)	BUS (bít)	Khối DES (/giây)
8088	4.7	8	370
68000	7.6	16	900
80286	6.0	16	1.100

68020	16.0	32	3.500
68030	16.0	32	3.900
80286	25.0	16	5.000
68030	50.0	32	9.600
68040	25.0	32	16.000
68040	40.0	32	23.200
80486	33.0	32	40.600

(Chú ý : Phần mềm này được viết trên C và Assembler, và có thể mua được từ Utimaco-Belgium, Interleuvenlaan 62A, B-300 leuven, Belgium. Cỡ mã xấp xỉ 64K. ANSI C thực hiện chậm hơn khoảng 20%.)

Một ứng dụng rất quan trọng của DES là trong giao dịch ngân hàng Mỹ. DES được dùng để mã hoá các số định danh các nhân (PIN) và việc chuyển tài khoản được thực hiện bằng máy thủ quỹ tự động (ATM). DES còn được sử dụng rộng dãi trong các tổ chức chính phủ.

3.11. Sự an toàn của DES

Đã có rất nhiều sự nghiên cứu về độ dài của khoá, số vòng lặp, và thiết kế của hộp S (S-boxes). Hộp S có đặc điểm là khó hiểu, không có bất cứ sự rõ □ang nào như tại sao chúng phải như vậy. Mọi tính toán trong DES ngoại trừ các hộp S đều tuyến tính, tức việc tính XOR của hai đầu ra cũng giống như phép XOR hai đầu vào rồi tính toán đầu ra. Các hộp S chứa đựng thành phần phi tuyến của hệ là yếu tố quan trọng nhất đối với sự an toàn của hệ thống.

Tính bảo mật của một hệ mã hoá đối xứng là một hàm hai tham số: độ phức tạp của thuật toán và độ dài của khoá.

Giả sử rằng tính bảo mật chỉ phụ thuộc vào độ phức tạp của thuật toán. Có nghĩa rằng sẽ không có phương pháp nào để phá vỡ hệ thống mật mã hơn là cố gắng thử mọi khoá có thể, phương pháp đó được gọi là brute-force attack. Nếu khoá có độ dài 8 bít, suy ra sẽ có 2^8=256 khoá. Vì vậy, sẽ mất nhiều nhất 256 lần thử để tìm ra khoá đúng. Nếu khoá có độ dài 56 bít, thì sẽ có 2^{56} khoá có thể sử dụng. Giả sử một Suppercomputer có thể thử một triệu khoá trong một giây, thì nó sẽ cần 2000 năm để tìm ra khoá đúng. Nếu khoá

có độ dài 64 bít, thì với chiếc máy trên sẽ cần 600,000 năm để tìm ra khoá đúng trong số 2^{64} khoá. Nếu khoá có độ dài 128 bít, thì sẽ mất 10^{25} năm để tìm ra khoá đúng. Vũ trụ chỉ mới tồn tại 10^{10} năm, vì vậy 10^{25} thì một thời gian quá dài. Với một khoá 2048 bít, một máy tính song song thực hiện hàng tỉ tỉ phép thử trong một giây sẽ tiêu tốn một khoảng thời gian là 10^{597} năm để tìm ra khoá. Lúc đố vũ trụ có lẽ không còn tồn tại nữa.

Khi IBM đưa ra thiết kế đầu tiên của hệ mã hoá LUCIFER, nó có khoá dài 128 bít. Ngày nay, DES đã trở thành một chuẩn về mã hoá dữ liệu sử dụng khoá 56 bít, tức kích thước không gian khoá là 2^{56}. Rất nhiều nhà mã hoá hiện đang tranh luận về một khoá dài hơn của DES. Nhiều thiết bị chuyên dụng đã được đề xuất nhằm phục vụ cho việc tấn công DES với bản rõ đã biết. Sự tấn công này chủ yếu thực hiện tìm khoá theo phương pháp vét cạn. Tức với bản rõ X 64 bít và bản mã Y tương ứng, mỗi khoá có thể đều được kiểm tra cho tới khi tìm được một khoá k thoả mãn $E_k(X)=Y$ (có thể có nhiều hơn một khoá k như vậy).

Vào năm 1979, Diffie và Hellman tuyên bố rằng với một máy tính chuyên dụng bản mã hoá DES có thể được phá bằng cách thử mọi trường hợp của khoá trong vòng một ngày – giá của máy tính đó là 20 triệu đôla. Vào năm 1981, Diffie đã tăng lên là cần hai ngày để tìm kiếm và giá của chiếc máy tính đó là 50 triệu đôla.

3.12. Tranh 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 tóan 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 trong chương 1 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:

P_0 Mỗi hàng trong mỗi hộp S là một hoán vị của các số nguyên 0, 1, . . . , 15.

P_1 Không một hộp S nào là một hàm Affine hoặc tuyến tính các đầu vào của nó.

P_2 Việc thay đổi một bít vào của S phải tạo nên sự thay đổi ít nhất là hai bít ra.

P_3 Đối với hộp S bất kì và với đầu vào x bất kì S(x) và S(x $\oplus$ 001100) phải khác nhau tối thiểu là hai bít (trong đó x là xâu bít 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.

P_4 Với hộp S bất kì, đầu vào x bất kì và với e, f $\in$ {0,1}: S(x) $\neq$S(x $\oplus$ 11ef00).

P5 Với hộp S bất kì , nếu cố định một bít vào và xem xét giá trị của một bít đầu ra cố định thì các mẫu vào để bít ra này bằng 0 sẽ xấp xỉ bằng số mẫu ra để bít đó bằng 1.(Chú ý rằng, nếu cố định giá trị bít vào thứ nhất hoặc bít vào thứ 6 thì có 16 mẫu vào làm cho một bít ra cụ thể bằng 0 và có 16 mẫu vào làm cho bít này bằng 1. Với các bít vào từ bít thứ hai đến bít 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 kì, nếu ta cố định giá trị của một bít vào bất kì thì số mẫu vào làm cho một bít 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 bi chuyên dụng đã được đè 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 bít và bản mã y tương ứng, mỗi khoá đều có thể được kiểm tra cho tới

khi tìm được một khoá K thảo 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 chíp VLSI (mạch tích hợp mật độ lớn) có khả năng kiểm tra được 10^6khoá/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 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\$/khung. Giá của một khung máy chứa 5760 chíp 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 khung 10 khung máy như vậy có giá chừng 10^6 \$ sẽ giảm thời gian tìm kiếm khoá trng bình xuống còn 3,5 giờ.

3.13. DES trong thực tế.

Mặc dù việc mô tả DES khá dài dòng song người ta có thể thực hiện DES rất hữa hiệu bằng cả phần cứng lẫn phần mền. Các phép toán duy nhất cần được thực hiện là phép hoặc loại trừ các xâu bít. Hàm mở rộng E, các hộp S, các hoán vị IP và P và việc tính toán các giá tri K_1,.. . ,K_{16} đều có thể thực hiện được cùng lúc bằng tra bảng (trong phần mền) 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 CRUPTO'92 rằng họ sẽ chế tạo một xung có 50 ngàn xung có thể mã hoá với tốc độ 1 Gbít/s bằng cách xung nhịp có tốc độ 250MHz. Giá của xung 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.

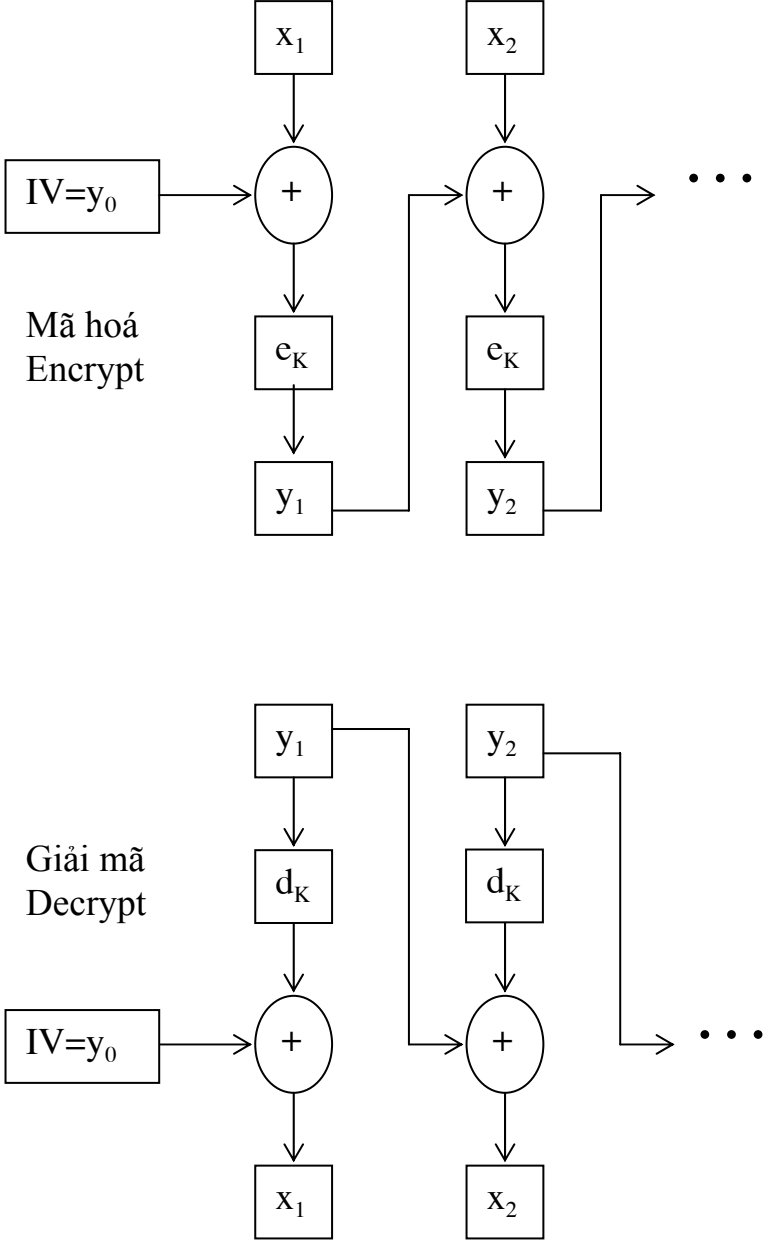
Một ứng dụng quan trọng của DES là trong giao dịch ngân hàng Mỹ - (ABA) DES được dùng để mã hoá các số định danh cá nhân (PIN) và việc chuyển tài khoản bằng máy thủ quỹ tự động (ATM). DES cũng được Hệ thống chi trả giữa các nhà băng của Ngân hàng hối đoái (CHIPS) dùng để xác

thực các giao dịch vào khoản 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.

3.14. Các chế độ hoạt động của DES.

Có 4 chế độ làm việc đã được phát triển cho DES: Chế độ chuyển 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 bít), mỗi x_i sẽ được mã hoá bằng cùng một khoá K để tạo thành một chuỗi các khối bản mã y_1y_2 ... theo quy tắc $y_i = e_K(y_{i-1}\oplus x_i)$ $i \geq 1$. Việc sử dụng chế độ CBC được mô tả trên hình 3.4.

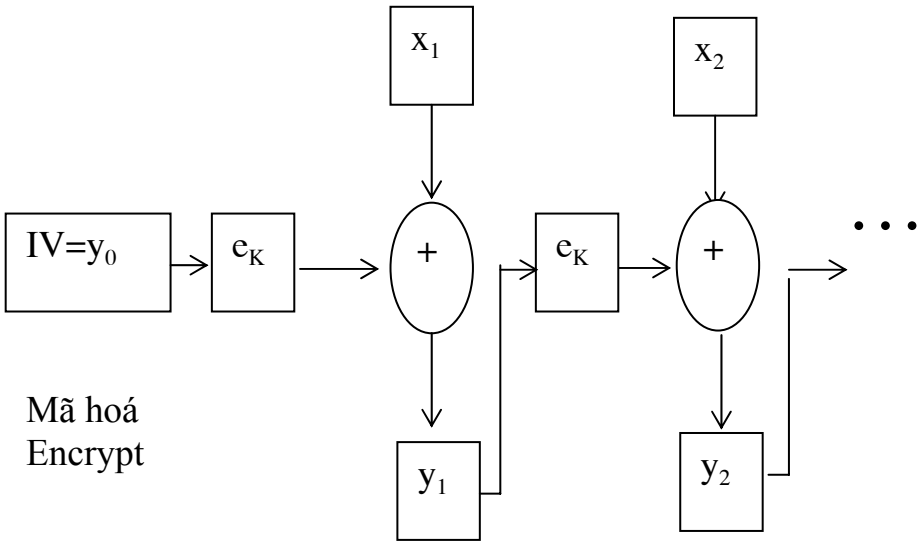
Hình 3.4. Chế độ CBC.

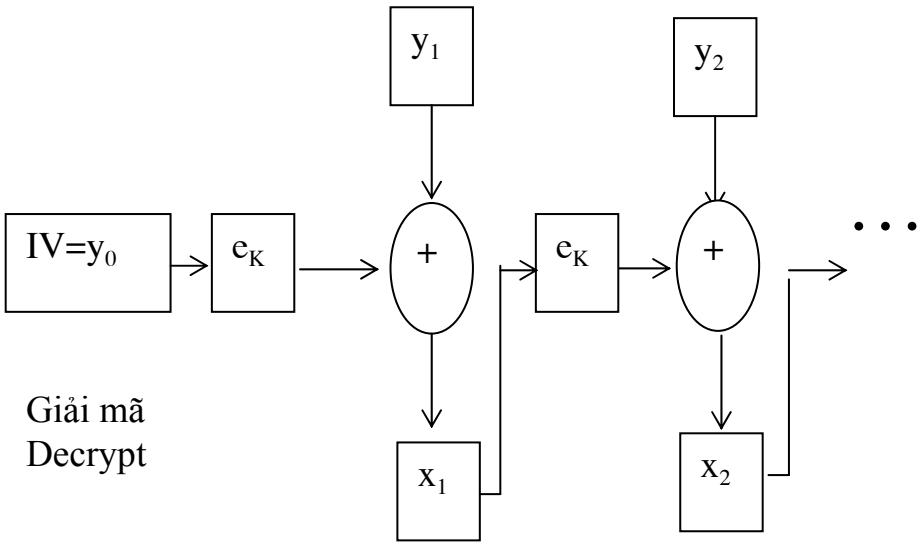


Trong các chế độ OFB và CFB dòng khoá được tạo ra sẽ được cộng mod 2 với bản rõ (tức là nó hoạt động như một hệ mã dòng, xem phần 1.1.7). OFB thực sự là một hệ mã dòng đồng bộ: dòng khoá được tạo bởi việc mã lặp véc tơ khởi tạo 64 bít (véc tơ IV). Ta xác định z_0 =IV và rồi tính dòng khoá z_1z_2 . . . theo quy tắc $z_i = e_K(z_{i-1})$, i≥1. Dãy bản rõ x_1x_2 . . . sau đó sẽ được mã hoá bằng cách tính $y_i = x_i \oplus z_i$,i ≥1.

Trong chế độ CFB, ta bắt đầu với y_0 = IV (là một véc tơ khởi tạo 64 bít) 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 ≥1. Cũng như trong chế độ OFB: $y_i = x_i \oplus z_i$,i ≥1. Việc sử dụng CFB được mô tả trên hình 3.5 (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).

Hình 3.5. Chế độ CFB





Cũng còn một số biến tấu của OFB và CFB được gọi là các chế độ phản hồi K bít (1 < K < 64). ở đây ta đã mô tả các chế độ phản hồi 64 bít. Các chế độ phản hồi 1 bít và 8 bít 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 bít 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.

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 bi 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 cáchb sử dụng chế độ BCB để tạo ra một MAC. Ta bắt đầu bằng véc tơ khởi tạ IV chứa toàn số 0. Sau đó dùng chế đô 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 $x_1,x_2,. . . ,x_n$ 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 x1. . .x_{n+1} bằng khoá thứ hai K_2 để tạo ra bản mã $y_1. . .y_{n+1}$. Khi Bob thu được y1. . .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 dung K_1 để giải mã $y_1. . .y_n$

Chương 4: Mật mã công khai

4.1. Giới thiệu về hệ mật mã khóa công khai.

4.1.1. Giới thiệu.

Trong mô hình mật mã cổ điển mà cho tới nay vẫn còn đang được nghiên cứu Alice (người gửi) và Bob (người nhận) bằng cách chọn một khoá bí mật K. Sau đó Alice dùng khoá K để mã hoá theo luật e_K và Bod dùng khoá K đó để giải mã theo luật giải d_K . Trong hệ mật này, d_K hoặc giống như e_K hoặc dễ dàng nhận được từ nó vì quá trình giải mã hoàn toàn tương tự như quá trình mã, nhưng thủ tục khoá thì ngược lại. Nhược điểm lớn của hệ mật này là nếu ta để lộ e_K thì làm cho hệ thống mất an toàn, chính vì vậy chúng ta phải tạo cho các hệ mật này một kênh an toàn mà kinh phí để tạo một kênh an toàn không phải là rẻ.

Ý tưởng xây dựng một hệ mật khoá công khai là tìm một hệ mật không có khả năng tính toán để xác định d_K nếu biết được 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 danh bạ, và khi Alice (người gửi) hoặc bất cứ một ai đó muốn gửi một bản tin cho Bob (người nhận) thì người đó không phải thông tin trước với Bob (người nhận) về khoá mật, mà người gửi sẽ mã hoá bản tin bằng cách dùng luật mã công khai e_K. Khi bản tin này được chuyển cho Bob (người nhận) thì chỉ có duy nhất Bob mới có thể giải được bản tin này bằng cách sử dụng luật giải mã bí mật d_K.

Ý tưởng về hệ mật khoá công khai đã được Diffie và Heliman đưa ra vào năm 1976. Còn việc thực hiện hệ mật khoá công khai thì lại được Rivest. Shamin và Adieman đưa ra đầu tiên vào năm 1977. Họ đã tạo nên hệ mật RSA nổi tiếng. Kể từ đó đã có một số hệ mật được công bố, độ mật của từng hệ dựa trên các bài toán tính toán khác nhau. Trong đó quan trọng nhất là các hệ mật sau:

- Hệ mật RSA

Độ bảo mật của hệ RSA dựa trên độ khó của việc phân tích ra thừa số nguyên tố các số nguyên tố lớn.

- Hệ mật xếp balô Merkle – Hellman.

Hệ này và các hệ có liên quan dựa trên tính khó giải của bài toán tổng các tập con.

- Hệ mật McEliece

Hệ mật nanỳ dựa trên lý thuyết mã đại số và vẫn được coi là an toàn. Hệ mật McEliece dựa trên bài toán giải mã cho các mã tuyến tính.

- Hệ mật ElGamal

Hệ ElGamal dựa trên tính khó giải của bài toán Logarit rời rạc trên các trường hữu hạn.

- Hệ mật Chor – Rivest

Hệ mật Chor – Rivest cũng được xem như một loại hệ mật xếp balô. Tuy nhiên hệ mật này vẫn còn được coi là hệ mật an toàn.

- Hệ mật trên các đường cong Elliptic.

Các hệ này là biến tướng của hệ mật khác, chúng làm việc trên các đường cong Elliptic chứ không phải trên các trường hữu hạn. Hệ mật này đảm bảo độ mật vơi khoá số nhỏ hơn các hệ mật khoá công khai khác.

Một chú ý quan trọng là một hệ mật khoá công khai không bao giờ có thể bảo đảm được độ mật tuyệt đối (an toàn vô điền kiện). Sở dĩ vậy vì đối phương nghiên cứu một bản mã C có thể mã lần lượt các bản rõ có thể bằng luật mã công khai e_K cho tới khi anh ta tìm được một bản rõ duy nhất P bảo đảm $C = e_K(P)$. Bản rõ P này chính là kết quả giải mã của C. Bởi vậy ta chỉ nghiên cứu độ mật về mặt tính toán của hệ này.

Một chú ý quan trọng và có ý ích khi nghiên cứu nữa là khái niệm về hàm cửa sập một chiều. Ta định nghĩa khái niệm này một cách không hình thức.

Định nghĩa: Hàm f: $X \rightarrow Y$ đực gọi là hàm một chiều nếu tính $y=f(x)$ với mọi $x \in X$ là dễ nhưng việc tìm x khi biết y lại là vấn đề khó.

Thực ra phát biểu trên chỉ là định nghĩa phi hình thức (do thuật ngữ "khó" được dùng đến là không định lượng và thậm chí sau này chúng ta đã biết là ngay cả khi đã định lượng bằng sự không tồn tại thuật toán giải bài

toán ngược trong phạm vi đa thức thì khái niệm "khó" nêu trên có tồn tại hay không cũng chưa được ai khẳng định rõ ràng) và điều đáng tiếc hơn nữa là tất cả các hàm ứng cử viên cho khái niệm này cho đến nay chỉ mới "được coi là một chiều.

Chúng ta dễ dàng thống nhất được với nhau là chỉ riêng hàm một chiều là không đủ để xây dựng thành một luật mã theo kiểu công khai hàm mã hoá do vì chính bản thân chủ nhân của bức điện mật cũng gặp phải hoàn cảnh tương tự như người khác. Như vậy để có thể giải mã một cách hữu hiệu thì người giải mã phải có một "hiểu biết tuyệt mật" nào đó về khoá giải (một hiểu biết theo kiểu nếu biết nó thì cách giải dễ dàng) "hiểu biết tuyệt mật" này được gọi là cửa sập. Hàm một chiều như trên được gọi là hàm một chiều có cửa sập.

Dĩ nhiên dù không biết cửa sập thì người thám mã vẫn có thể sử dụng hiểu biết về hàm f để lần lượt tính tất cả các giá trị f(x) cho mọi bản rõ x cho tới khi tìm được bản rõ thoả mãn y=f(x). Bản rõ tìm được trên chính là kết quả giải mã của y. Ngoài ra người thám mã còn có thể sử dụng nhiều phương pháp tấn công khác nhằm vào đặc thù riêng của từng hàm f để tìm ra bản rõ trong các trường hợp riêng rẽ khác chứ không nhất thiết phải giải bài toán ngược.

Tóm lại đọc an toàn của hệ mật khoá công khai không chỉ phụ thuộc vào độ khó của việc giải bài toán ngược mà tính bền của sự an toàn này còn phụ thuộc vào các phương pháp tấn công của các thám mã, vả lại như đã trình bày ở trên thì toàn bộ các hê khoá mật công khai đang được sử dụng đều chưa đực sự khẳng định về tính "khó" mà ngay cả khi đã có sự đảm bảo này thì có sự tiến bộ không ngừng của công nghệ tính toán tghì hiển nhiên nhiều vấn đề chưa thể chứp nhận được trong hiện tại sẽ được chấp nhận trong tương lai. Thực tế không chỉ đối với các hệ mât khoá công khai do vậy quan niêm mới về tính an toàn tương đối mà với nó đã nẩy sinh ra các hệ mật khoá công khai đồng thời cũng đặt cho chúng ta nhiều bài toán nghiêm túc phải giải quyết khi sử dụng hệ mật này. Chương này giới thiệu cụ thể một số hệ mật công khai

mà với nó sự an toàn cũng như khả năng ứng dụng của nó đã được các bộ óc vĩ trên thế giới thừa nhận là hệ mật khoá công khai sáng giá nhất, đó là hệ mật khoá công khai RSA.

Hàm mã công khai e_k của Bob phải là một hàm dễ tính toán. Song việc tính hàm ngược (tức là hàm giải mã) phải rất khó khăn (đối với bất kỳ ai không phải là Bob). Đặc tính dễ tính toán nhưng khó tính ngược thường được gọi là đặc tính một chiều. Bởi vậy điều cần thiết là e_k phải là một hàm một chiều.

Các hàm một chiều đóng một vai trò trọng yếu trong mật mã học: Chúng rất quan trọng trong việc xây dựng 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 tới nay vẫn không tồn tại được một hàm nào có thể chứng minh được là một 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 p và q, giả sử b là một số nguyên dương. Khi đó ta xác định ánh xạ $f:Z_n \rightarrow Z_n$ là

$f(x)=x^b \bmod n$.

(với b và n được chọn thích hợp thì đây chính là hàm mã RSA).

Để xây dựng một hệ mật khoá công khai thì việc tìm một hàm một chiều vẫn chưa đủ. Ta không muốn e_k là một hàm một chiều đối với Bob vì anh ta phải có khả năng giải mã các bản tin nhận được có hiệu quả. Điều cần thiết là Bob phải có một cửa sập chứa thông tin bí mật cho phép dễ dàng tìm ngược 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 hàm một chiều và nó sẽ trở nên dễ tính ngược nếu biết một cửa sập nhất định.

4.1.2. Nhắc lại một số kiến thức số học liên quan

Định nghĩa:

Hàm Phi Euler của số nguyên dương n là số các số nguyên tố cùng nhau với n nhỏ hơn n.Kí hiệu $\theta(n)$

Ví dụ: θ(6)=2, θ(26)=12

Tính chất của hàm Phi euler:

1. Nếu n là số nguyên tố thì θ(n) = n-1

Ví dụ: θ(7)=6

2. Nếu p, q là 2 số nguyên tố cùng nhau thi:

θ(p*q)=θ(p)*θ(q)

ví dụ θ(26)=θ(2*13)=θ(2)*θ(13)=1*12=12

3. Nếu p là số nguyên tố thi: $\theta(p^r)=(p-1)*p^{r-1}$

Định lý:

Nếu a, n là nguyên tố cùng nhau thi $a^{\theta(n)}=1 \bmod n$

4.2. Hệ mật RSA

4.2.1. Thuật toán RSA

RSA là tên viết tắt của ba tác giả Rivest, Sharmir, Adleman của trường MIT đã đề ra hệ mật mã công khai. Hệ mật này được đề xuất năm 1977, dựa trên cơ sở tính các luỹ thừa trong số học. Độ an toàn của hệ mật dựa trên độ khó của việc phân tích thành thừa số nguyên tố của các số nguyên lớn. Nhiều hệ mật khoá công khai sau này đã được phát triển nhưng đều thua kém hệ RSA. Các hệ balo cửa sập đã bị phá vỡ và cho đến nay, ngoài hệ RSA, chưa có một hệ nào khác cung cấp được cả độ an toàn và chữ ký số.

a. Thuật toán tạo khoá

Bước 1: B (người nhận) tạo hai số nguyên tố lớn ngẫu nhiên p và q (p<>q)

Bước 2: B tính n=p*q và $\Phi(n) = (p-1)(q-1)$

Bước 3: B chọn một số ngẫu nhiên e ($0 < e < \Phi(n)$) sao cho ƯCLN$(b, \Phi(n))=1$

Bước 4: B tính $d=e^{-1}$ bằng cách dùng thuật toán Euclide

Bước 5: B công bố n và e trong danh bạ làm khoá công khai (public key), còn d làm khoá bí mật (private key).

b. Thuật toán mã hoá và giải mã

+ *Mã hoá:*

Bước 1: A nhận khoá công khai của B.

Bước 2: A biểu diễn thông tin cần gửi thành số m (0 <= m <= n-1)

Bước 3: Tính $c= m^e \bmod n$

Bước 4: Gửi c cho B.

+ *Giải mã:* B giải mã bằng cách tính $m=c^d \bmod n$

**** Chứng minh hệ mật RSA***

+ Cần chứng minh: $m = (m^e \bmod n)^d \bmod n$

Thật vậy

p, q là số nguyên tố, n=pq, $\Phi(n) = (p-1)(q-1)$ nên ta có

$m^{\Phi(n)} = 1 \bmod n$

Mặt khác, do $ed = 1 \bmod n$ nên $ed = k\Phi(n) + 1$

Theo định lý Fermat ta có

$x^{p-1} = 1 \bmod p \rightarrow x^{(p-1)(q-1)} = 1 \bmod p$

$x^{q-1} = 1 \bmod q \rightarrow x^{(p-1)(q-1)} = 1 \bmod q$

$\rightarrow x^{\Phi(n)} = 1 \bmod n$

$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$

$= m^{k.\Phi(n)+1} \bmod n$

$= m^1 \bmod n$

$= m$ (dpcm)

**** Ví dụ:***

B chọn p=5, q=7. Khi đó n=35, Φ=24

Chọn e = 5 (e và Φ nguyên tố cùng nhau).

	Letter	m	m^e	$c=m^e \bmod n$
Encrypt	I	12	1524832	17

	c	c^d	$m=c^d \bmod n$	letter
Decrypt	17	48196857210675091509141182522307200		

123.3

4.2.2. Một số thuật toán triển khai trong RSA I

****Thuật toán "bình phương và nhân" như sau:***

Tính x^b mod n

Trước hết biểu diễn b=$\sum_{i=0}^{l-1} b_i 2^i 2$ trong đó b_i = 0 hoặc 1, $0 \le i \le l-1$.

i) z=1

ii) cho i chạy từ giá trị l-1 về 0

z=z^2 mod n

Nếu b_i = 1 thì z=z*x mod n

iii) giá trị cần tìm chính là giá trị z cuối cùng.

Như vậy sử dụng thuật toán "bình phương và nhân" sẽ làm giảm số phép nhân modulo cần thiết, để tính x mod n nhiều nhất là 2, trong l là số bít trong biểu diễn nhị phân của b. Vì $l \le k$ nên có thể coi x^b mod n được thực hiện trong thời gian đa thức $0(k^3)$.

*** Thuật toán Ơclít mở rộng.**

Begin

g_0:= $\Phi(n)$; g_1:=e;

u_0:=1; u_1:=0;

v_0:=0; v_1:=1;

While $g_i \neq 0$ do

Begin

y:=g_{i-1} div g_i ;

g_{i+1}:= $g_{i-1} - y.g_i$;

u_{i+1}:= $u_{i-1} - y.u_i$;

v_{i+1}:= $v_{i-1} - y.v_i$;

i:= i+1 ;

End;

x:= v_{i-1};

If x>0 then d:=x else d:=x+$\Phi(n)$;

END.

Vì vậy muốn xây dựng hệ RSA an toàn thì n=pq phải là một số đủ lớn, để không có khả năng phân tích nó về mặt tính toán. Để đảm bảo an toàn nên chọn các số nguyên tố p và q từ 100 chữ số trở lên.

Tuy nhiên máy tính thông thường khó có thể tính toán với số nguyên lớn đến mức như vậy. Do đó cần phải có thư viện các thuật toán làm việc với các số nguyên lớn. Ta có thể lưu trữ số lớn như sau:

- Phân tích số lớn thành số nhị phân.
- Chia số nhị phân thành các khối 32 bít, lưu vào mảng, mỗi phần tử của mảng lưu 32 bít.

Ví dụ: giả sử a là số lớn được phân tích thành số nhị phân $a = a_0a_1…a_n$

32 bít	32 bít	……	32 bít
a_0	a_1	……	a_n

** Cộng hai số lớn:*

Số a	a_0	a_1	……	a_n	
Số b	b_0	b_1	……	b_n	
Số c	c_0	c_1	……	c_n	c_{n+1}

Có một ô nhớ 32 bít để ghi số nhớ khi cộng 2 số, ban đầu ô nhớ này bằng 0.

Khi cộng thì các phần tử tương ứng cộng với nhau

nhớ + $a_0 + b_0 = c_0$

nhớ + $a_1 + b_1 = c_1$

nhớ + $a_i + b_i = c_i$

Để xem kết quả có nhớ hay không khi tổng $c_i < a_i$ thì nhớ = 1

Mảng lưu trữ tổng bao giờ cũng lớn hơn mảng của các số hạng tổng một phần tử, phần tử mảng cuối cùng này (c_{n+1}) lưu số nhớ.

** Nhân số lớn*

Khi nhân 2 số 32 bit sẽ tạo ra số 64 bít nhưng hiện nay máy tính không lưu được số 64 bít, nên nó chia số 64 bít thành 2 số 32 bít (32 bít thấp và 32 bít cao). Ban đầu nhớ = 0.

32 bít low	32 bít high

Như vậy khi nhân a_0 x b_0 + nhớ = c_0 (c_0 là số 64 bít), số c_0 sẽ chia thành 2 số 32 bít và ghi vào mảng c phần tử c_0 là số 32 bít thấp và số nhớ là 32 bít cao.

Phần tử tiếp theo $c_1 = a_0$ x $b_1 + a_1$ x b_0 + nhớ.

c_1 cũng chia làm 2 số 32 bít và ghi lại vào mảng c phần tử c_1 số 32 bít thấp và số nhớ là 32 bít cao. Tương tự như vậy ta có tổng quát sau:

$$c_i = nho + \sum_{k=0}^{i} a_k b_{i-k}$$

Điều cốt yếu trong việc thiết lập hệ RSA là tạo ra các số nguyên tố lớn (khoảng 100 chữ số). Quá trình thực hiện trong thực tế là : trước hết tạo ra các số ngẫu nhiên lớn, sau đó kiểm tra tính nguyên tố của nó bằng cách dùng thuật toán xác suất Monte – Carlo thời gian đa thức (như thuật toán Miller – Rabin hoặc thuật toán Solovay – Strasen). Đây là các thuật toán kiểm tra tính nguyên tố nhanh của số n trong thời gian đa thức theo $\log_2 n$, là số các bít trong biểu diễn nhị phân của n). Tuy nhiên vẫn có khả năng thuật toán kiểm tra n là số nguyên tố nhưng thực tế n vẫn là hợp số. Bởi vậy, bằng cách thay đổi thuật toán nhiều lần , có thể giảm xác suất sai số dưới một ngưỡng cho phép.

Thuật toán kiểm tra số nguyên tố: thuật toán Miller – Rabin

- Phân tích $n - 1 = 2^k . m$, với m lẻ
- Chọn ngẫu nhiên một số a sao cho $1 \le a \le n-1$
- Tính $b \equiv a^m \bmod n$.
- Nếu b = 1 thì n là số nguyên tố và thoát.
- For i:=1 to k-1 do
- Nếu b = -1 thì n là số nguyên tố, nếu không $b = b^2 \bmod n$.
- Trả lời n là hợp số.

Xác suất sai lầm của thuật toán này là < 1/4.

Trong thực tế thì chưa được biết có một thusật toán kiểm tra chắc chắn số sinh ra có phải nguyên tố hay không.

Một vấn đề quan trọng khác: là cần phải kiểm tra bao nhiêu số nguyên tố ngẫu nhiên (với kích thước xác định) cho tới khi tìm được một số nguyên tố. Một kết quả nổi tiếng trong lý thuyết số (gọi là định lý số nguyên tố) phát biểu rằng: số các số nguyên tố không lớn hơn N xấp xỉ bằng N/lnN. Bởi vậy, nếu p được chọn ngẫu nhiên thì xác suất p là một số nguyên tố sẽ vào khoảng 1/lnp.

4.2.3. Độ an toàn của hệ mật RSA.

a. Bài toán phân tích số và việc phá hệ mật RSA.

Cách tấn công dễ thấy nhất đối với hệ mật RSA là người thám mã sẽ cống gắng phân tích n rathừa số nguyên tố n=p*q và khi đó anh ta dễ dàng tính được φ(n)=(p-1)(q-1) và do đó tìm được thông tin cửa sập d tương ứng với thông tin mã hoá E bằng thuật toán Euclide. Như vậy chúng ta thấy ngay rằng việc phá hệ mật RSA là "dễ hơn" bài toán phân tích số nguyên ra thừa số nguyên tố tuy nhiên cũng chưa có một kết quả nào chỉ ra rằng bài toán phân tích số là thực sự khó hơn cho nên người ta thườn thừa nhận rằng bài toán phá hệ RSA là tương đương với bài toán phân tích số nguyên thành thừa số người.

Để đảm bảo tính khó phân tích ra thừa số của n=p*q thì yêu cầu đầu tiên là p,q là các số nguyên tố lớn xấp xỉ bằng nhau và là số nguyên tố "mạnh ". Khái niệm "mạnh" ở đây chỉ bắt nguồn từ ý nghĩa khó phân tích do vậy nó sẽ được bổ xung cùng với kết quả có được của khả năng phân tích số. Nói một cách khác là khái niệm "mạnh" bao gồm sự loại trừ các lớp số nguyên tố mà với chúng tồn tại thuật toán phân tích hiệu quả, chúng ta có thể biết đến một khái niệm sơ khai của tính "mạnh" đó là các số nguyên tố p mà p-1 và p+1 có chứa thừa số nguyên tố lớn.

b. Việc tấn công hệ mật RSA khác phương pháp phân tích số.

Một kết quả thú vị là một thuật toán bất kỳ để tính số mũ giải mã d đều có thể được dùng như một chương trình con trong thuật toán xác suất kiểu Las Vegas để phân tích n.

Như vậy mặc dù rằng nếu d bị lộ thì việc phân tích n cũng không còn ý nghĩa theo quan điểm phá hệ mật tuy nhiên kết quả trên dù sao cũng cho ta một thuật toán phân tích số n khi biết d với xác suất thành công không quá ½ của mỗi lần chọn số ngẫu nhiên làm đầu vào cho thuật toán.

4.2.4. Các thuật toán phân tích số.

Trong phần này giới thiệu một số thuật toán phân tích số nguyên được coi là "mạnh nhất" theo nghĩa thời gian tính tốt nhất hiện nay. Việc trình bày của chúng tôi dựa trên quan điểm không phải là đưa ra thuật toán chi tiết nhằm mục đích phân tích số nguyên mà chủ yếu nêu ra ý tưởng của thuật toán và quan trọng nhất là đưa ra thông số về thời gian tính của chúng nhằm chứng minh cho kích thước tối thiểu của các modulo được sử dụng trong mật mã theo dạng tích hai số nguyên tố lớn. Các thuật toán được kể đến bao gồm thuật toán sàng bậc hai, thuật toán phân tích trên đường cong Elliptic, thuật toán sàng trường số.... nhưng do hai thuật toán sau đều cần phải có kiến thức bổ trợ khá cồng kềnh về đại số hiện đại vả lại điều kiện về tài liệu lại không đủ chi tiết nên bài giảng này chỉ trình bày thuật toán sàng bậc hai và cũng dừng ở những nét chính yếu nhất.

Các thuật toán phân tích số:

**** Thuật toán sàng Eratosthenes***

Đây là thuật toán có tính phổ thông, với n có ước nhỏ thì việc áp dụng thuật toán này là hiệu quả. Thời gian tính của nó là $0(\sqrt{n})$. Thuật toán được mô tả như sau:

i) p=1

ii) p=p+1

iii) Tính r = n mod p. Nếu r > 0 quay về bước 2.

Ngược lại p là ước của N, dừng chương trình.

**** Thuật toán sàng đồng dư***

Thuật toán được mô tả như sau:

i) Lấy ngẫu nhiên hai số a và b, với a,b $\in Z_n^*$

ii) Kiểm tra gcd((a-b) mod n,n) >1 hoặc gcd((a+b) mod n,n)>1

- Nếu đúng thì gcd((a-b) mod n,n) >1 hoặc gcd((a+b) mod n,n)>1 là ước của n dừng chương trình.

- Ngược lại quay về i)

Phân tích thuật toán này dưới góc độ xác suất: Cho p là ước nguyên tố nhỏ nhất của n, thế thì cần có tối thiểu bao nhiêu cặp a,b được xét đến để xác suất có ít nhất một cặp trong số đó thoả mãn $((a \pm b) \bmod p) \equiv 0 \geq 0.5$?

Bài toán trên được gọi là bài toán "trùng ngày sinh" và số m tối thiểu cần tìm trong bài toán sẽ là $m \approx c.p$, với c là một hằng số tính được nào đó. Thuật toán có thể thành công với xác suất >0.5, sau không quá m bước.

Bằng cách duyệt dần thì thời gian của thuật toán không khác gì thời gian của phép sàng. Tác giả J.M.Pollard đã sử dụng một phương pháp còn gọi là "phương pháp δ". Chỉ cần thông qua $\sqrt{m}$ bước có thể duyệt được m cặp khác nhau như đã nêu trên trong thuật toán.

**** Thuật toán Pollard***

Thuật toán hiệu quả trong việc tìm các ước nhỏ là thuật toán dựa vào phương pháp δ và được gọi là thuật toán Pollard. Thời gian tính của thuật toán này chỉ còn là $0(\sqrt{n})$. . Với p là ước nguyên tố nhỏ nhất của n. Trong trường hợp tồi nhất ($p \approx \sqrt{n}$) thì thời gian tính của thuật toán cũng chỉ là $\sqrt[4]{n}$

Phương pháp δ của Pollard:

Tìm hai phần tử đồng dư modilo p ($a \equiv \pm b \bmod p$) nhưng không đồng dư modulo n. Lúc này p sẽ là ước của gcd(n, (a mb) mod n). Có thể mô tả thuật toán như sau:

Chọn dãy giả ngẫu nhiên {x_i mod n, i=1,2,…} được xác định như sau: $x_{i+1} \equiv (x_i^2 + a) \bmod n$ với $a \neq 0$ và $a \neq -2$ còn giá trị đầu x_0 tuỳ ý.

Thuật toán:

i) i=0

ii) i:=i+1

iii) Xét gcd(($x_{2i} - x_i$) mod n,n) > 1

- Nếu đúng ta có p = gcd(($x_{2i} - x_i$) mod n,n). Dừng chương trình

- Ngược quay về bước ii)

Chúng ta đi phân tích thời gian của thuật toán:

$x_{2i} - x_i \equiv (x_{2i-1}^2 + a) - (x^2_{i-1} + a) \equiv x^2_{2i-1} - x^2_{i-1})$

$\equiv (x_{2i-1} - x_{i-1})(x_{2i-1} + x_{i-1}) \equiv$

$\equiv (x_{2i-1} + x_{i-1})(x_{2i-2} + x_{i-2}) \ldots (x_i + x_0)(x_i - x_0)$

Tại bước thứ i chúng ta xét đến i+1 cặp khác nhau và cũng dễ dàng nhận ra rằng các cặp được xét trong mọi bước là không giống nhau, do đó hiển nhiên với $\sqrt{p}$ bước chúng ra đã có p cặp khác nhau được xét đến và như đã phân tích ở trên. Thuật toán thành công với xác suất > 0.5 hay thuật toán của Pollard được thực hiện trong $0(\sqrt{n})$ bước.

**** Thuật toán p-1***

Thuật toán p – 1 của Pollard là thuật toán phân tích số nguyên n dựa vào phân tích của p – 1với p là một ước nguyên tố của n. Đây là một thuật toán có tác dụng nếu ta biết được các ước nguyên tố của một thừa số p của n nói chung và đặc biệt nếu n có một thừa số nguyên tố p mà p – 1 chỉ gồm những ước nguyên tố nhỏ nhất thì thuật toán có hiệu quả. Thuật toán này chỉ có hai đầu vào là n số nguyên lẻ cần được phân tích và một số b.

Các bước của thuật toán

i) Đầu vào là hai số n và b

ii) a:=2

iii) for j:=2 to b do a: = a^j mod n

iv) d = gcd(a-1,n)

v) if 1 < d < n then d là một thừa số của n

else không tìm được thừa số của n.

Ví dụ:

Giả sử n = 15770708441 và b=180. áp dụng thuật toán p – 1 ta có:

+ a = 1160221425

+ d = 135979

Thực tế phân tích đầy đủ n thành các ước nguyên tố là:

N = 15770708441 =135979 x 115979

Phép phân tích sẽ thành công do 135978 chỉ gồm các thừa số nguyên tố nhỏ: 135978 = 2 x 3 x 131 x 173

Trong thuật toán có (b-1) luỹ thừa theo modulo, mỗi luỹ thừa cần nhiều nhất là $2\log_2 b$ phép nhân modulo dùng thuật toán bình phương và nhân. Việc tìm ước chung lớn nhất có thể được thực hiện trong thời gian $0((\log n)^3)$ bằng thuật toán Ơclít. Bởi vậy, độ phức tạp của thuật toán là

$0(b \log b (\log n)^2 + (\log n)^3)$

Nếu b là $0((\log n)^i$ với một số nguyên i xác định nào đó thì thuật toán thực sự là thuật toán thời gian đa thức, tuy nhiên với phép chọn b như vậy, xác suất thành công sẽ rất nhỏ. Mặt khác, nếu tăng kích thước của b lên thật lớn thì thuật toán sẽ thành công nhưng nó sẽ không nhanh hơn phép chia thử.

Điểm bất lợi của thuật toán này là nó yêu cầu n phải có ước nguyên tố p sao cho p - 1 chỉ có các thừa số nguyên tố bé. Ta có thể xây dựng được hệ mật RSA với modulo n = p.q hạn chế được việc phân tích theo phương pháp này. Trước tiên tìm một số nguyên tố lớn p_1 sao cho $p = 2p_1 + 1$ cũng là một số nguyên tố và một số nguyên tố lớn q_1 sao cho $q = 2q_1 + 1$ cũng là một số nguyên tố. Khi đó modulo của RSA n = p.q sẽ chống được cách phân tích theo phương pháp p – 1.

**** Thuật toán p ± 1***

Thuật toán p ± 1 của Williams cũng dựa vào kết quả phân tích của p ± 1 với p là một ước nguyên tố của n. Để tiện nghiên cứu phương pháp p ± 1, trước hết điểm lại một số kết quả của chính liên quan đến dãy Lucas

Định nghĩa 1: (dãy Lucas)

Cho a, b là hai nghiệm của phương trình $x^2 - px + q = 0$ (1)

Ký hiệu $u_m = \frac{a^m - b^m}{a - b}$ và $v_m = a^m + b^m$ (2)

Các dãy $\{u_m\}$, $\{v_m\}$, m = 0, 1, 2,… gọi là dãy Lucas của phương trình (1)

Ngược lại phương trình (1) gọi là phương trình đặc trưng của dãy (2)

Tính chất 1: Nếu i là ước của j thì u_i ước của u_j

Tính chất 2: Ta có $u_0 = 0$, $u_1 = 1$, $v_0 = 2$, $v_1 = p$ và $\forall m > 1$ thì u_m và v_m được tính theo công thức sau:

$$\begin{bmatrix} u_{m+1} & v_{m+1} \\ u_m & v_m \end{bmatrix} = \begin{bmatrix} p & -Q \\ 1 & 0 \end{bmatrix}^m \begin{bmatrix} u_1 & v_1 \\ u_0 & v_0 \end{bmatrix}$$

Định lý: $\{u_m\}$ là dãy Lucas của phương trình (1) với $p^2 - 4Q = d^2\Delta$ có Δ không có ước chính phương (hay bình phương tự do). Nếu p không là ước của 4Q thì $u_p - \left[\frac{\Delta}{p}\right] \equiv 0 \bmod p$ ở đây $\left[\frac{\Delta}{p}\right]$ là ký hiệu Legendre

Thuật toán $p \pm 1$

i) $Q = 2^{\log_2 n} \ldots q^{\log_{q_k} k}$, i = 1, j = 0

ii) Lấy Δ không có ước chính phương ngẫu nhiên trong Z_n^*. Tìm R, S nguyên sao cho $R^2 - 4S = \Delta d^2$ với $d \neq 0$ nào đó.

Xét gcd(ΔQ, n) > 1

- Nếu đúng ta có ước của n là gcd(ΔQ, n). Dừng chương trình
- Ngược lại tính $b \equiv u_0 \bmod n$ (phần tử thứ Q trong dãy Lucas của phương trình $x^2 - Rx + S = 0$)

iii) Xét đẳng thức b = 0

- Nếu đúng chuyển sang (iv)
- Ngược lại chuyển sang (vi)

iv) Xét $j < \log_q n$

- Nếu đúng j = j + 1, Q = Q/q quay về (iii)
- Ngược lại chuyển sang (v)

v) Xét i < k

- Nếu đúng thì : i = i+1, j = 0
- Nếu $b \neq 1$ thì $Q = Q.q_i$ quay về (iv)
- Ngược lai quay về (i)

vi) Xét gcd(b,n) > 1

- Nếu đúng có ước của n là gcd(b,n). Dừng chương trình

- Ngược lại quay về (iv)

Ta thấy rằng để vét hết các khả năng p + 1 (trong trường hợp $\left[\frac{\Delta}{p}\right] = -1$ và p -1 (trong trường hợp $\left[\frac{\Delta}{p}\right] = 1$)) là ước của Q. Việc xét đẳng thức b = 0 trong mỗi bước, nếu sai nhằm đảm bảo cho ta b không là bội của n và nếu p + 1 hoặc p – 1 là ước của Q thì theo các kết quả ở tính chất và định lý trên cho ta b là bội của p và như vậy gcd(b,n) là ước thực sự của n.

Tóm lại, thuật toán trên rõ ràng hiệu quả trong cả hai trường hợp p + 1 hoặc p – 1 chỉ gồm các ước nguyên tố nhỏ, tuy nhiên căn cứ vào công thức tính các giá trị của dãy Lucas, ta thấy ngay rằng hệ số nhân của thuật toán này là lớn hơn nhiều so với thuật toán của Pollard trong trường hợp cùng phân tích được n với ước p của nó có p – 1 chỉ gồm các ước nhỏ bởi vì thay cho việc tính một luỹ thừa thông thường thì thuật toán của Lucas phải tính một luỹ thừa của một ma trận

Từ thuật toán trên, ta có thể kết luận:

- p phải là một số lớn
- Các ước phải có kích thước xấp xỉ nhau
- Các ước không được xấp xỉ nhau về giá trị
- Ước nguyên tố p của modulo n không được có p + 1 hoặc p – 1 phân tích hoàn toàn ra các thừa số nguyên tố nhỏ
- Không có số Lucas $u_i = 0 \bmod p$ với i bé đối với các phương trình đặc trưng có biểu thức Δ nhỏ
- P phải có khoảng cách luỹ thừa 2 đủ lớn.

**** Phương pháp Ơ le:***

Phương pháp Ơ le chỉ có tác dụng đối với một lớp số nguyên đặc biệt cụ thể là chỉ dùng phân tích cho các số nguyên là tích của các số nguyên tố cùng dạng $r^2 + DS^2$. Thuật toán dựa trên cơ sở là đẳng thức của Legendre (còn gọi là đẳng thức Diophantus)

Đẳng thức Diophantus:

$(x^2 + Ly^2)(a^2 + Lb^2) = (x \pm Lyb)^2 + L(xb \, mya)^2$

Chứng minh: Biến đổi vế phải đẳng thức trên:

$(xa \pm Ly^2) + L(xb \, mya)^2 = x^2a^2 \pm 2Labxy + L^2y^2b^2 + Lx^2b^2$ m2Labxy + $Ly^2a^2 = a^2(x^2 + Ly^2) + Lb^2(Ly^2 + x^{2)}) = (a^2 + Ly^2)(x^2 + Ly^2)$

Sau đó Ơ le đã chứng minh được rằng:

Định lý: Nếu n có hai biểu diễn khác nhau $n = r^2 + Ls^2 = u^2 + Lv^2$ với gcd() = 1 thì n phân tích được thành tích của hai thừa số n=p.q cùng dạng $p = x^2 + Ly^2$ và $q= a^2 + Lb^2$

Như vậy điều kiện nhận biết số nguyên n là tích của hai ước số đều có dạng $r^2 + Ls^2$ là n cũng có dạng đó và có hai biểu diễn khác nhau theo dạng trên.

Thứ nhất, ta thấy rằng từ $n = r^2 + Ls^2$ nên để tìm biểu diễn theo dạng đã nêu trên của n ta có thể tiến hành bằng cách duyệt theo s cới nhận biết $n – Ls^2$ là số chính phương. Với phương pháp dò tìm trên thì giá trị s tối đa cần xét đến là $\left[\sqrt{\frac{n}{b}}\right]$ và đây cũng là cận tính toán của thuật toán Ơle.

Giả sử đã tìm được hai biểu diễn khác nhau của n là: $n = r^2 + Ls^2 = u^2 + Lv^2$. Không mất tính tổng quát ta coi r, s, u, v không âm và r > u. Khi đó giải hệ phương trình sau đây ta tìm được x, y, a, b

$$\begin{cases} xa + Lyb = rv \\ xa - Lyb = \pm u \\ xb - ya = \pm s \\ xb + ya = v \end{cases}$$

Dấu trừ của phương trình (2) và 93) được lấy khi vế trái tương ứng âm.

Một điều khó khăn khi thực hiện thuật toán phân tích Ơle là vấn đề xác định tham số L. Nhìn chung việc thực hiện thuật toán Ơlư chỉ áp dụng cho những số n mà bản thân nó đã biết một biểu diễn. Tuy nhiên lại có thể bằng cách dò tìm L chúng ta có thể thành công trong việc phân tích.

Như vậy thuật toán nay chỉ dùng cho một lớp số đặc biệt nên khó được dùng để tạo nên một tiêu chuẩn thích hợp cho các modulo hợp số.

**** Phương pháp sàng Dyxon và sàng bậc hai***

Trong phần này giơi thiệu thuật toán phân tích hai số nguyên được coi là mạnh nhất theo nghĩa thời gian tính tốt nhất hiện nay. ý tưởng của một loạt khá lớn các thuật toán phân tích số như phương pháp phân tích các dạng chính phương Danien Shaks, phương pháp đặc biệt của Ơle, phương pháp khai triển liên phân số của Morrison và Brillhart, phương pháp sàng bậc hai của Pomerance, Dixon… là cố tìm được $x \neq \pm y \bmod n$ sao cho $x^2 \equiv y^2 \bmod n$, còn kỹ thuật tìm cụ thể như thế nào thì chính là nội dung riêng của từng thuật toán

Thuật toán Dixon được thực hiện như sau:

- Sử dụng một tập B chứa các số nguyên tố bé và gọi là cơ sở phân tích
- Chọn một vài số nguyên x sao cho tất cả các thừa số nguyên tố của $x^2 \bmod n$ nằm trong cơ sở B,
- Lấy tích của một vài giá trị x sao cho mỗi nguyên tố trong cơ sở được sử dụng một số chẵn lần. Chính điều này dẫn đến một đồng dư thức dạng mong muốn $x^2 \equiv y^2 \bmod n$ mà ta hy vọng sẽ đưa tới việc phân tích n và suy ra gcd(x-y,n) là một ước của n.

Ví dụ:

Giả sử chọn: n = 15770708441, B = {2, 3, 5, 7, 11, 13}

Và chọn ba giá trị x là : 8340934156, 12044942944, 2773700011

Xét ba đồng dư thức:

$8340934156^2 \equiv 3x7 \pmod n$

$12044942944^2 \equiv 2x7x13 \pmod n$

$2773700011^2 \equiv 2x3x13 \pmod n$

Lấy tích của ba đồng dư thức trên:

$(8340934156 \times 12044942944 \times 2773700011)^2 \equiv (2 \times 3 \times 7 \times 13)^2 \bmod n$

Rút gọn biểu thức bên trong dấu ngoặc trong modulo đó ta có:

$9503435785^2 \equiv 546^2 \pmod n$

Suy ra

$$\begin{cases} x = 9503435785 \\ y = 546 \end{cases}$$

Tính gcd(x-y,n) = gcd(9503435785 – 546, 15770708441) = 1157759

Ta nhận thấy 115759 là một thừa số của n

Giả sử:

- **B** = {p_1,…, p_B} là một cơ sở phân tích
- C lớn hơn B một chút (chẳng hạn C = B + 10)
- Có đồng dư thức: $x_j^2 \equiv p_1^{\alpha_{1j}} p_2^{\alpha_{2j}} \cdots p_B^{\alpha_{Bj}} \pmod n$

Với $1 \le j \le C$, mỗi j, xét véc tơ:

$a_j = (\alpha_{1j} \bmod 2, \alpha_{2j} \bmod 2, ..., \alpha_{Bj} \bmod 2) \in (Z_2)^B$

Nếu có thể tìm được một tập con các a_j sao cho tổng theo modulo 2 là vectơ (0, 0,…,0) thì tích của các x_j tương ứng sẽ được sử dụng mỗi nhân tử trong **B** một số chẵn lần.

Ví dụ:

Xét lại ví dụ trên n = 15770708441, **B** = {2, 3. 5, 11, 13Ư

Cho ba vectơ a_1, a_2, a_3 :

A_1 = (0, 1, 0, 1, 0, 0)

A_2 = (1, 0, 0, 1, 0, 1)

A_3 = (1, 1, 0, 0, 0, 1)

Suy ra $a_1 + a_2 + a_3$ = (0, 0, 0, 0, 0, 0) mod 2

Trong trường hợp này nếu C<B, vẫn tìm được phụ thuộc tuyến tính. Đây là lý do cho thấy đồng dư thức (thiết lập theo tích) sẽ phân tích thành công được n.

Bài toán tìm một tập con C véc tơ a_1, a_2, …, a_c sao cho tổng theo modulo 2 là một véctơ toàn chứa số 0 chính là bài toán tìm sự phụ thuộc tuyến tính (trên Z_2) của vectơ này. Với C > B, sự phụ thuộc tuyến tính này nhất định phải tồn tại và ta có thể dễ dàng tìm được bằng phương pháp loại trừ Gaux. Lý do giải thích tại sao lấy C > B + 1 là do không có gì đảm bảo để một đồng dư thức cho trước bất kỳ sẽ tạo được phân tích n. Người ta chỉ ra rằng khoảng 50% thời gian thuật toán cho ra $x \equiv \pm y \pmod n$. Tuy nhiên nếu C > B + 1 thì

có thể nhận được một vài đồng dư thức như vậy. Hy vọng là ít nhất một trong các đồng dư thức kết quả sẽ dẫn đến việc phân tích n.

Vấn đề cần đặt ra là phải làm như thế nào để nhận được các số nguyên x_j mà các giá trị x_j^2 mod n có thể phân tích hoàn toàn trên cơ sở **B**. Một số phương pháp có thể thực hiện được điều đó. Biện pháp sàng bậc hai do Pomerance đưa ra dùng các số nguyên dạng $x_j = j + \lfloor \sqrt{n} \rfloor$, j = 1, 2, … dùng để xác định các x_j phân tích được trên **B**.

Nếu B là một số lớn thì thích hợp hơn cả là nên phân tích số nguyên x_j trên **B**. Khi B càng lớn thì càng phải gom nhiều đồng dư thức hơn trước khi có thể tìm ra một số quan hệ phụ thuộc và điều này dẫn đến thời gian thực hiện cỡ

$$0(e^{(1+0(1)\sqrt{\ln n \ln\ln n})})$$

Với 0(1) là một hàm tiến tới 0 khi n tiến tới ∞

Thuật toán sàng trường số là thuật toán cũng phân tích n bằng cách xây dựng một đồng dư thức $x^2 \equiv y^2$ mod n, song nó lại được thực hiện bằng cách tính toán trên vành các số đại số.

**** Thời gian tính các thuật toán trên thực tế***

Thuật toán đường cong Elliptic hiệu quả hơn nếu các thừa số nguyên tố của n có kích thước khác nhau. Một số rất lớn đã được phân tích bằng thuật toán đường cong Elliptic là số Fermat $(2^{2^n} - 1)$ (được Brent thực hiện năm 1988). Thời gian tính của thuật toán này được tính là

$$0(e^{(1+0(1)\sqrt{2\ln p \ln\ln p})})$$

p là thừa số nguyên tố nhỏ nhất của n

Trong trường hợp nếu hai ước của n chênh lệch nhau nhiều thì thuật toán đường cong Elliptic tỏ ra hơn hẳn thuật toán sàng bậc hai. Tuy nhiên nếu hai ước của n xấp xỉ nhau thì thuật toán sàng bậc hai nói chung trội hơn thuật toán đường cong Elliptic.

Sàng bậc hai là một thuật toán thành công nhất khi phân tích các modulo RSA với n = p.q và p, q là các số nguyên tố có cùng kích thước. Năm 1983, thuật toán sàng bậc 2 đã phân tích thành công số có 69 chữ số, số này là một thừa số của $2^{251} - 1$ (do Davis, Holdredye và Simmons thực hiện). Đến năm 1989 đã có thể phân tích được các số có tới 106 chữ số theo thuật toán này (do Lenstra và Manasse thực hiện), nhờ phân bố các phép tính cho hàng trăm trạm làm việc tách biệt (người ta gọi phương pháp này là "Phân tích thừa số bằng thư tín điện tử").

Các số RSA – d với d là chữ số thập phân của số RSA (d = 100 $\div$ 500) được công bố trên Internet như là sự thách đố cho các thuật toán phân tích số. Vào 4/1994 Atkins, Lenstra và Leyland đã phân tích được một số 129 chữ số, nhờ sử dụng sàng bậc hai. Việc phân tích số RSA – 129 trong vòng một năm tính toán với máy tính có tốc độ 5 tỷ lệnh trên 1 giây, với công sức của hơn 600 nhà nghiên cứu trên thế giới.

Thuật toán sàng trường số là một thuật toán mới nhất trong ba thuật toán. Thuật toán sàng trường số cũng phân tích số nguyên n bằng việc xây dựng đồng dư thức $x^2 \equiv y^2 \bmod n$. Nhưng việc thực hiện bằng cách tính toán trên các vành đại số… Sàng trường số vẫn còn trong thời kỳ nghiên cứu. Tuy nhiên theo dự đoán thì phải chứng tỏ nhanh hơn với các số có trên 125 chữ số thập phân. Thời gian tính của thuật toán sàng trường số là

$$0(e^{(1.92-0(1))\sqrt[3]{\ln n}\sqrt[3]{(\ln \ln n)^2}})$$

Việc trình bày các thuật toán phân tích trên để hiểu rõ một phần nào các biện pháp tấn công vào RSA để có thể xây dựng một hệ mật an toàn hơn. Từ các thuật toán trên yêu cầu đối với p và q nên thoả mãn:

- Các số nguyên p và q phải xấp xỉ nhau về độ dài nhưng không được xấp xỉ nhau về độ lớn.

- Các số $p \pm 1$ và $q \pm 1$ phải có ít nhất một thừa số nguyên tố lớn

- Phải có khoảng luỹ thừa 2 đủ lớn

- Giá trị F = gcd($p \pm 1$, $q \pm 1$) không được lớn hơn $\sqrt[3]{n}$

- Các số p và q phải là các số có ít nhất 100 chữ số thập phân

Nhận xét đầu để ngăn chặn khả năng tấn công bởi thuật toán sơ đẳng nhất, đó là thuật toán sàng, đồng thời như các phân tích trên thì đã đưa bài toán phân tích về trường hợp khó giải nhất, của ngay thuật toán được đánh giá là có triển vọng nhất đó là thuật toán dựa vào phương pháp trường số.

Nhận xét thứ hai dựa vào khả năng của thuật toán Pollard và thuật toán Williams mà khả năng đó phụ thuộc chủ yếu vào việc các số $p \pm 1$ và $q \pm 1$ phân tích được hoàn toàn qua các số nguyên tố trong tập **B**. Trong tập **B** có thể là tập các số nguyên tố nhỏ hơn 32 bits. Ngược lại cũng có thể sử dụng tập **B** lớn hơn. Do đó nhận xét này cũng hợp lý.

Việc có một tham số công khai như số mũ lập mã e chắc chắn phải cung cấp thêm thông tin cho bài toán phân tích số. Do đó cần tìm hiểu mức độ ảnh hưởng của thông tin này để xây dựng nên một yêu cầu với số mũ e này và phần nào đó có tính đối ngẫu liên quan cả số mũ giải mã d.

Để cho một số nguyên tố đáp ứng tiêu chuẩn về độ dài thì đối với hệ mật sử dụng bài toán logarit cần các số nguyên tố có độ dài khoảng gấp rưỡi so với các số nguyên tố dùng cho loại hệ mật dựa trên bài toán phân tích số. Nếu có được một thuật toán nhanh (thuật toán xác suất như Rabin – Miller) thì thời gian tính cũng phải cỡ $0(n^3)$ (với n là độ dài khoảng gấp rưỡi so với các số nguyên tố trong các số nhỏ hơn n theo Direcle là $\Pi(n) \approx \frac{\ln n}{n}$, do vậy khả năng tìm được số nguyên tố 521 bít so với một số nguyên tố 350 bit lâu hơn gấp nhiều lần.

Thiết kế một hệ mật sử dụng bài toán logarit rời rạc chỉ cần đúng một số nguyên tố trong khi để có một tính năng tương đương, thì hệ mật dựa trên bài toán phân tích số nguyên ra thừa số nguyên tố cần đến 2k số nguyên tố cho hệ thống có k người sử dụng. Các số nguyên tố cần dùng cho hệ mật thứ hai đòi hỏi phải có các ước nguyên tố lớn, dẫn đến khả năng tìm kiếm số nguyên tố cũng sẽ khó khăn hơn nhiều so với hệ mật thứ nhất.

4.3. Một số hệ mật mã công khai khác

Trong chương này ta sẽ xem xét một số hệ mật khoá công khai khác. Hệ mật Elgamal dựa trên bài toán logarithm rời rạc là bài toán được dùng nhiều trong nhiều thủ tục mật mã. Bởi vậy ta sẽ dành nhiều thời gian để thảo luận về bài toán quan trọng này. ở các phần sau sẽ xem xét sơ lược một số hệ mật khoá công khai quan trọng khác bao gồm các hệ thoóng loại Elgamal dựa trên các trường hữu hạn và các đường cong elliptic, hệ mật xếp ba lô Merkle-Helman và hệ mật McElice.

4.3.1.Hệ mật Elgamal và các logarithm rời rạc.

Hệ mật Elgamal được xây dựng trên bài toán logảithm rời rạc . Chúng ta sẽ bắt đầu băng việc mô tả bài toán bài khi thiết lập môi trường hữu hạn Z_p, p là số nguyên tố (Nhớ lại rằng nhóm nhân Z_p^* là nhóm cyclic và phần tử sinh của Z_p^* được gọi là phần tử nguyên thuỷ).

Bài toán logarithm rời rạc trong Zp là đối tượng trong nhiều công trình nghiên cứu và được xem là bài toán khó nếu p được chọn cẩn thận. Cụ thể không có một thuật toán thời gian đa thức nào cho bài toán logarithm rời rạc. Để gây khó khăn cho các phương pháp tấn công đã biết p phải có ít nhất 150 chữ số và (p-1) phải có ít nhất một thừa số nguyên tố lớn. Lợi thế của bài toán logarithm rời rạc trong xây dượng hệ mật là khó tìm được các logarithm rời rạc, song bài toán ngược lấy luỹ thừa lại có thể tính toán hiệu quả theo thuật toán "bình phương và nhân". Nói cách khác, luỹ thừa theo modulo p là hàm một chiều với các số nguyên tố p thích hợp.

Elgamal đã phát triển một hệ mật khoá công khai dựa trên bài toán logarithm rời rạc. Hệ thống này được trình bày sau.

Hệ mật này là một hệ không tất định vì bản mã phụ thuộc vào cả bản rõ x lẫn giá trị ngẫu nhiên k do Alice chọn. Bởi vậy, sẽ có nhiều bản mã được mã từ cùng bản rõ.

Bài toán logarithm rời rạc trong Zp

Đặc trương của bài toán: I = (p,α,β) trong đó p là số nguyên tố,
$\alpha \in Zp$ là phần tử nguyên thuỷ , $\beta \in Zp^*$
Mục tiêu:Hãy tìm một số nguyên duy nhất a, $0 \le a \le p-2$ sao cho:

$$\alpha^a \equiv \beta \pmod p$$

Ta sẽ xác định số nguyên a bằng $\log_\alpha \beta$

Hệ mật khoá công khai Elgamal trong Zp^*

Cho p là số nguyên tố sao cho bài toán logarithm rời rạc trong Zp là khó giải. Cho $\alpha \in Zp^*$ là phần tử nguyên thuỷ.Giả sử $P = Zp^*$, $C = Zp^* \times Zp^*$. Ta định nghĩa:

$$K = \{(p, \alpha, a, \beta): \beta \equiv \alpha^a \pmod p\}$$

Các giá trị p, α,β được công khai, còn a giữ kín

Với K = (p, α,a,β) và một số ngẫu nhiên bí mật $k \in Zp-1$, ta xác định:

$$e_k(x,k) = (y_1, y_2)$$

trong đó

$$y_1 = \alpha^k \bmod p$$

$$y_2 = x\beta^k \bmod p$$

với $y_1, y_2 \in Zp^*$ ta xác định:

$$d_k(y_1, y_2) = y_2 (y_1^a)^{-1} \bmod p$$

Sau đây sẽ nmô tả sơ lược cách làm việc của hệ mật Elgamal .Bản rõ x được "che dấu" bằng cách nhân nó với β^k để tạo y_2. Giá trị α^k cũng được gửi đi như một phần của bản mã. Bob – người biết số mũ bí mật a có thể tính được β^k từ α^k. Sau đó anh ta sẽ "tháo mặt nạ" bằng cách chia y_2 cho β^k để thu được x.

Ví dụ:

Cho p = 2579, α = 2, a = 765. Khi đó

$$\beta = 2^{765} \bmod 2579 = 949$$

Bây giờ ta giả sử Alice muốn gửi thông báo x = 1299 tới Bob. Giả sử số ngẫu nhiên k mà cô chọn là k = 853. Sau đó cô ta tính

$y_1 = 2^{853} \bmod 2579$

$= 435$

$y2 = 1299 \times 949853 \bmod 2579$

$= 2396$

Khi đó Bob thu được bản mã y = (435,2396), anh ta tính

$x = 2396 \times (435^{765})^{-1} \bmod 2579$

$= 1299$

Đó chính là bản rõ mà Alice đã mã hoá.

4.3.2 Mật mã Balô.

4.3.2.1. Cơ sở của mật mã balô

Mật mã balô xuất phát từ bài toán tổng tập con tổng quát (bài toán □al ô). Bài toán được phát biểu như sau:

Cho dãy các số dương S={s1, s2,…., sn} và một số dương C. Hỏi có tồn tại một tập con nằm trong S sao cho tổng tập con đó bằng C. (Hỏi có tồn tại một véc tơ nhị phân x=(x1, x2,…, xn) sao cho C=∑xi.si (i=1..n))

Đây là bài toán khó có thời gian là hàm mũ $O(2^n)$.

Nếu S là dãy siêu tăng thì bài toán trên giải được với thời gian tuyến tính O(n).

Định nghĩa: *Dãy S gọi là siêu tăng nếu mọi si>∑sj (j=1,..i-1) (tức là phần tử đứng sau lớn hơn tổng các phần tử đứng trước nó)*

Khi đó bài toán tổng tập con được phát biểu như sau:

Cho dãy siêu tăng S={s1, s2,…., sn} và một số dương C. Hỏi có tồn tại một tập con nằm trong S sao cho tổng tập con đó bằng C. (Hỏi có tồn tại một véc tơ nhị phân x=(x1, x2,…, xn) sao cho C=∑xi.si (i=1..n))

Khi đó bài toán được giải như sau:

```
For i:=n downto 1 do
Begin
   If C>=si then
         xi=1
```

Else xi:=0;

C:=C-xi.si;

End;

If C=0 then "bài toán có đáp án là véc tơ x"

Else "bài toán không có đáp án";

Áp dụng bài toán này ta sử dụng dãy S siêu tăng làm khóa bí mật. Sau đó tác động lên dãy S để biến đổi thành một dãy bất kỳ, và công khai dãy này là khóa công khai. Ta có hệ mật mã □al ô như sau:

4.3.2.2. Thuật toán:

* Tạo khóa:

- Chọn dãy siêu tăng S={s1, s2, …, s3}

- Chọn p sao cho p>∑si (i=1..n)

- Chọn a sao cho 1<a<p-1 và (a,p)=1;

- tính t=a.s mod p

=> khóa công khai là t, khóa bí mật là: a, p, S

* Mã:

Chọn bản rõ là dãy nhị phân x=(x1, x2,…, xn)

Tính bản mã y=∑xi.ti (i=1..n)

Gửi bản mã y

* Giải mã:

- Tính $C=a^{-1}.y \bmod p$

- Giải bài toán ba lô với S là dãy siêu tăng và số dương C để tìm bản rõ x

* Chứng minh tính đúng của hệ mật mã ba lô (Bạn đọc tự chứng minh)

Ví dụ:

(Như một bài tập).

Chương 5

Các sơ đồ chữ kí số

5.1. Giới thiệu.

Trong chương này, chúng ta xem xét các sơ đồ chữ kí số (còn được gọi là chữ kí số). Chữ kí viết tay thông thường trên tài liệu thường được dùng để xác người kí nó. Chữ kí được dùng hàng ngày chẳng hạn như trên một bức thư nhận tiền từ nhà băng, kí hợp đồng…

Sơ đồ chữ kí là phương pháp kí một bức điện lưu dưới dạng điện tử. Chẳng hạn một bức điện có ký hiệu được truyền trên mạng máy tinh. Chương này trình bày một vài sơ đồ chữ kí số. Ta sẽ thảo luận trên một vài khác biệt cơ bản giữa các chữ kí thông thường và chữ kí số.

Đầu tiên là một vấn đề kí một tài liệu. Với chữ kí thông thường, nó là một phần vật lý của tài liệu. Tuy nhiên, một chữ kí số không gắn theo kiểu vật lý vào bức điện nên thuật toán được dùng phải "không nhìn thấy" theo cách nào đó trên bức điện.

Thứ hai là vấn đề về kiểm tra. Chữ kí thông thường được kiểm tra bằng cách so sánh nó với các chữ kí xác thực khác. Ví dụ, ai đó kí một tấm séc để mua hàng, người bán phải so sánh chữ kí trên mảnh giấy với chữ kí nằm ở mặt sau của thẻ tín dụng để kiểm tra. Dĩ nhiên, đây không phải là phương pháp an toàn vì nó dể dàng giả mạo. Mắt khác, các chữ kí số có thể được kiểm tra nhờ dùng một thuật toán kiểm tra công khai. Như vậy, bất kỳ ai cũng có thể kiểm tra dược chữ kí số. Việc dùng một sơ đồ chữ kí an toàn có thể sẽ ngăn chặn được khả năng giả mạo.

Sự khác biệt cơ bản khác giữa chữ kí số và chữ kí thông thường bản copy tài liệu được kí băng chữ kí số đồng nhất với bản gốc, còn copy tài liệu có chữ kí trên giấy thường có thể khác với bản gốc. Điều này có nghĩa là phải cẩn thận ngăn chăn một bức kí số khỏi bị dung lại. Vì thế, bản thân bức điện cần chứa thông tin (chẳng hạn như ngày tháng) để ngăn nó khỏi bị dùng lại.

Một sơ đồ chữ kí số thường chứa hai thành phần: thuật toán kí và thuật toán xác minh. Bob có thể kí bức điện x dùng thuật toán kí an toàn. Chữ kí

y=sig(x) nhận được có thể kiểm tra bằng thuật toán xác minh công khai ver(x,y). Khi cho trước cặp (x,y), thuật toán xác minh có giá trị TRUE hay FALSE tuỳ thuộc vào chữ kí được thực như thế nào. Dưới đây là định nghĩa hình thức của chữ kí:

Định nghĩa:

Một sơ đồ chữ kí số là bộ 5(P, A, K, S, V) thoả mãn các điều kiện dưới đây:

1. P là tập hữu hạn các bức điện có thể.
2. A là tập hữu hạn các chữ kí có thể.
3. K không gian khoá là tập hữu hạn các khoá có thể.
4. Với mỗi k thuộc K tồn tại một thuật toán kí $sig_k \in S$ và là một thuật toán xác minh $ver_k \in V$. Mỗi $sig_k : P \rightarrow A$ và ver_k: P×a →{true,false} là những hàm sao cho mỗi bức điện x∈ P và mối chữ kí y∈ A thoả mãn phương trình dưới đây.

$$ver_k \quad \begin{cases} \text{True nếu } y=sig(x) \\ \text{False nếu } y \# sig(x) \end{cases}$$

Với mỗi k thuộc K hàm sig_k và ver_k là các hàm thời than đa thức. Ver_k sẽ là hàm công khai sig_k là mật. Không thể dễ dàng tính toán để giả mạo chữ kí của Bob trên bức điện x. Nghĩa là x cho trước, chỉ có Bob mới có thể tính được y để ver_k = True. Một sơ đồ chữ kí không thể an toàn vô điều kiện vì Oscar có thể kiểm tra tất cả các chữ số y có thể có trên bức điện x nhờ □ung thuật toán ver công khai cho đến khi anh ta tìm thấy một chữ kí đúng. Vi thế, nếu có đủ thời gian. Oscar luôn luôn có thể giả mạo chữ kí của Bob. Như vậy, giống như trường hợp hệ thống mã khoá công khai, mục đích của chúng ta là tìm các sơ đồ chữ kí số an toan về mặt tính toán.

Xem thấy rằng, hệ thống mã khoá công khai RSA có thể □ung làm sơ đồ chữ kí số.

Như vậy, Bob kí bức điện x dùng qui tắc giải mã RSA là d_k. Bob là người tạo ra chữ kí vì $d_k = sig_k$ là mật. Thuật toán xác minh dùng qui tắc mã RSA e_k. Bất kì ai cũng có thể xác minh chữ kí vi e_k được công khai.

Chú ý rằng, ai đó có thể giả mạo chữ kí của Bob trên một bức điện " ngẫu nhiên" x bằng cách tìm $x=e_k(y)$ với y nào đó, khi đó $y= sig_k(x)$. Một giải pháp xung quanh vấn đề khó khăn này là yêu cầu bức điện chưa đủ phần dư để chữ kí giả mạo kiểu này không tương ứng với bức điện. Nghĩa là x trừ một xác suất rất bé. Có thể dùng các hàm hash trong việc kết nối với các sơ đồ chữ kí số sẽ loại trừ được phương pháp giả mạo này.

Sơ đồ chữ kí RSA

Cho n= p.q, p và q là các số nguyên tố. Cho $P =A= Z_n$
$ab \equiv 1(\mathrm{mod}(\phi(n)))$. Các giá trị n và b là công khai, a giữ bí mật.
Hàm kí:
$sig_k(x)= x^a \bmod n$
và kiểm tra chữ kí:
$ver_k (x,y)= \text{true} \Leftrightarrow x \equiv y^b \pmod n$
$(x,y \in Z_n)$

Ta xét tóm tắt cách kết hợp chữ kí và mã khoá công khai. Giả sử rằng, Alice tính toán chữ kí $y= sig_{Alice}(x)$ và sau đó mã cả x và y bằng hàm mã khoá công khai e_{Bob} của Bob, khi đó cô ta nhận được $z = e_{Bob}(x,y)$. Bản mã z sẽ được truyền tới Bob. Khi Bob nhận được z, anh ta sẽ trước hết sẽ giải mã hàm d_{Bob} để nhận được (x,y). Sau đó anh ta ung hàm xác minh công khai của Alice để kiểm tra xem $ver_{Alice}(x,y)$ có bằng True hay không.

Song nếu đầu tiên Alice mã x rồi sau đó mới kí tên bản mã nhận được thì khi đó cô tính :

$$y= sig_{Alice}(e_{Bob}(x)).$$

Alice sẽ truyền cặp (z,y) tới Bob. Bob sẽ giải mã z, nhận x và sau đó xác minh chữ kí y trên x nhờ dùng ver_{Alice}. Một vấn đề tiềm ẩn trong biện pháp này là nếu Oscar nhận được cặp (x,y) kiểu này, được ta có thay chữ kí y của Alice bằng chữ kí của mình.

$$Y' = sig_{Oscar}(e_{Bob}(x)).$$

(Chú ý, Oscar có thể kí bản mã $e_{Bob}(x)$ ngay cả khi anh ta không biết bản rõ x). Khi đó nếu Oscar truyền (x, y') đến Bob thì chữ kí Oscar được Bob xác minh bằng ver_{Oscar} và Bob có thể suy ra rằng, bản rõ x xuất phát từ Oscar. Do khó khăn này, hầu hết người sử dụng được khuyến nghị nếu kí trước khi mã.

5.2. Sơ đồ chữ kí ELGAMAL

Sau đây ta sẽ mô tả sơ đồ chữ kí Elgamal đã từng dưới thiệu trong bài báo năm 1985. Bản cả tiến của sơ đồ này đã được Viện Tiêu chuẩn và Công Nghệ Quốc Gia Mỹ (NIST) chấp nhận làm chữ kí số. Sơ đồ Elgamal (E.) được thiết kế với mục đích dành riêng cho chữ kí số, khác sơ đồ RSA dùng cho cả hệ thống mã khoá công khai lẫn chữ kí số.

Sơ đồ E, là không tất định giống như hệ thống mã khoá công khai Elgamal. Điều này có nghĩa là có nhiều chữ kí hợp lệ trên bức điện cho trước bất kỳ. Thuật toán xác minh phải có khả năng chấp nhận bất kì chữ kí hợp lệ khi xác thực.

Nếu chữ kí được thiết lập đúng khi xác minh sẽ thành công vì :

$$\beta^{\gamma}\gamma^{\delta} \equiv \alpha^{a\gamma}\alpha^{k\gamma} (\text{mod } p)$$

$$\equiv \alpha^{x} (\text{mod } p)$$

là ở đây ta dùng hệ thức :

$$a\gamma + k\delta \equiv x \ (\text{mod } p-1)$$

Sơ đồ chữ kí số Elgamal.

Cho p là số nguyên tố sao cho bài toán logarit rời rạc trên Z_p là khó và giả sử $\alpha \in Z_n$ là phần tử nguyên thuỷ $p = Z_p^*$, $a = Z_p^* \times Z_{p-1}$ và định nghĩa:

$K = \{(p,\alpha,a,\beta) : \beta \equiv \alpha^a (\text{mod } p)\}$.

Giá trị p,α,β là công khai, còn a là mật.

Với $K = (p, \alpha, a, \beta)$ và một số ngẫu nhiên (mật) $k \in Z_{p-1}$. định nghĩa :

$$Sig_k(x,y) = (\gamma,\delta),$$

trong đó $\gamma = \alpha^k \bmod p$

và $\delta = (x-a)\, k^{-1} \bmod (p-1)$.

Với $x, \gamma \in Z_p$ và $\delta \in Z_{p-1}$, ta định nghĩa :

$$Ver(x, \gamma, \delta) = \text{true} \Leftrightarrow \beta^{\gamma}\gamma^{\delta} \equiv \alpha^{x} (\text{mod } p).$$

Bob tính chữ kí bằng cách dùng cả gía trị mật a (là một phần của khoá) lẫn số ngẫu nhiên mật k (dùng để kí lên bức điện x). Việc xác minh có thực hiện duy nhất bằng thông báo tin công khai.

Chúng ta hãy xét một ví dụ nhỏ minh hoạ.

Giả sử cho p = 467, α =2, a = 127, khi đó:

$$\beta = \alpha^a \bmod p$$
$$= 2^{127} \bmod 467$$
$$= 132$$

Nếu Bob muốn kí lên bức điện x = 100 và chọn số ngẫu nhiên k =213 (chú ý là UCLN(213,466) =1 và $213^{-1} \bmod 466 = 431$. Khi đó

$$\gamma = 2^{213} \bmod 467 = 29$$

và $\delta = (100 - 127 \times 29)\ 431 \bmod 466 = 51.$

Bất kỳ ai củng có thể xác minh chữ kí bằng các kiểm tra :

$$132^{29}\ 29^{51} \equiv 189 \pmod{467}$$

và $$2^{100} \equiv 189 \pmod{467}$$

Vì thế chữ kí là hợp lệ.

Xét độ mật của sơ đồ chữ kí E. Giả sử, Oscar thử giả mạo chữ kí trên bức điện x cho trước không biết a. Nếu Oscar chọn γ và sau đó thử tìm giá trị δ tương ứng, anh ta phải tính logarithm rời rạc $\log_\gamma \alpha^x \beta^{-\gamma}$. Mặt khác, nếu đầu tiên ta chọn δ và sau đó thử tim γ và thử giải phương trình:

$$\beta^\gamma\ \gamma^\delta \equiv \alpha^x \pmod p.$$

để tìm γ. Đây là bài toán chưa có lời giải nào. Tuy nhiên, dường như nó chưa được gắn với đến bài toán đã nghiên cứu kĩ nào nên vẫn có khả năng có cách nào đó để tính δ và γ đồng thời để (δ, γ) là một chữ kí. Hiện thời không ai tìm được cách giải song cũng ai không khẳng định được rằng nó không thể giải được.

Nếu Oscar chọn δ và γ và sau đó tự giải tìm x, anh ta sẽ phải đối mặt với bài toán logarithm rời rạc, tức bài toán tính $\log_\alpha$ Vì thế Oscar không thể kí một bức điện ngẫu nhiên bằng biện pháp này. Tuy nhiên, có một cách để Oscar có thể kí lên bức điện ngẫu nhiên bằng việc chọn γ, δ và x đồng thời: giả thiết i và j là các số nguyên $0 \le i \le p-2$, $0 \le j \le p-2$ và UCLN(j,p-2) = 1. Khi đó thực hiện các tính toán sau:

$$\gamma = \alpha^i \beta^j \bmod p$$

$$\delta = -\gamma\, j^{-1} \bmod (p-1)$$

$$x = -\gamma\, i\, j^{-1} \bmod (p-1)$$

Trong đó j^{-1} được tính theo modulo (p-1) (ở đây đòi hỏi j nguyên tố cùng nhau với p-1).

Ta nói rằng (γ, δ) là chữ kí hợp lệ của x. Điều này được chứng minh qua việc kiểm tra xác minh :

Ta sẽ minh hoạ bằng một ví dụ :

Giống như ví dụ trước cho p = 467, α = 2, β =132. Giả sữ Oscar chọn i = 99,j = 179; khi đó $j^{-1} \bmod (p-1) = 151$. Anh ta tính toán như sau:

$$\gamma = 2^{99}132^{197} \bmod 467 = 117$$

$$\delta = -117 \times 151 \bmod 466 = 51.$$

$$x = 99 \times 41 \bmod 466 = 331$$

Khi đó (117, 41) là chữ kí hợp lệ trên bức điện 331 như thế đã xác minh qua phép kiểm tra sau:

$132^{117}\, 117^{41} \equiv 303 \pmod{467}$

và $2^{331} \equiv 303 \pmod{467}$

Vì thế chữ kí là hợp lệ.

Sau đây là kiểu giả mạo thứ hai trong đó Oscar bắt đầu bằng bức điện được Bob kí trước đây. Giả sử (γ, δ) là chữ kí hợp lệ trên x. Khi đó Oscar có khả năng kí lên nhiều bức điện khác nhau. Giả sử i, j, h là các số nguyên, $0 \le h, i, j \le p-2$ và UCLN (h γ - j δ, p-1) = 1. Ta thực hiện tính toán sau:

$\lambda = \gamma^h \alpha^i \beta^j \bmod p$

$\mu = \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1)$

$x' = \lambda(hx+i\delta)^{-1} \bmod (p-1)$,

Trong đó $(h\gamma - j\delta)^{-1}$ được tính theo modulo (p-1). Khi đó dễ dàng kiểm tra điệu kiện xác minh :

$$\beta^{\lambda} \lambda^{\mu} \equiv \alpha^{x'} \pmod p$$

vì thế (λ, μ)là chữ kí hợp lệ của x'.

Cả hai phương pháp trên đều tạo các chữ kí giả mạo hợp lệ song không xuất hiện khả năng đối phương giả mạo chữ kí trên bức điện có sự lựu chọn của chính họ mà không phải giải bài toán logarithm rời rạc, vì thế không có gì nguy hiểm về độ an toàn của sơ đồ chữ kí Elgamal.

Cuối cùng, ta sẽ nêu vài cách có thể phải được sơ đồ này nếu không áp dụng nó một cách cẩn thận (có một số ví dụ nữa về khiếm khuyết của giao thức, một số trong đó là xét trong chương 4). Trước hết, giá trị k ngẫu nhiên được dùng để tính chữ kí phải giữ kín không để lộ. vì nếu k bị lộ, khá đơn giản để tính :

$$A = (x - k\gamma)\delta^{-1} \bmod (p-1).$$

Dĩ nhiên, một khi a bị lộ thì hệ thống bị phá và Oscar có thể dễ dang giả mạo chữ kí.

Một kiểu dung sai sơ đồ nữa là dùng cùng giá trị k để kí hai bức điện khác nhau. điều này cùng tạo thuận lợi cho Oscar tinh a và phá hệ thống. Sau đây là cách thực hiện. Giả sử (γ, δ_1) là chữ kí trên x_1 và (γ, δ_2) là chữ kí trên x_2. Khi đó ta có:

$$\beta^{\gamma} \gamma^{\delta 1} \equiv \alpha^{x1} \pmod p$$

và $$\beta^{\gamma} \gamma^{\delta 2} \equiv \alpha^{x2} (\text{mod} p).$$

Như vậy

$$\alpha^{x1-x2} \equiv \alpha^{\delta 1-\delta 2} \pmod p.$$

Nếu viết $\gamma = \alpha^k$, ta nhận được phương trình tìm k chưa biết sau.

$$\alpha^{x1-x2} \equiv \alpha^{k(\delta 1 - \delta 2)} \pmod{p}$$

tương đương với phương trình

$$x_1 - x_2 \equiv k(\delta_1 - \delta_2) \pmod{p-1}.$$

Bây giờ giả sử d =UCLN($\delta_1 - \delta_2$, p-1). Vì d | (p-1) và d | ($\delta_1 - \delta_2$) nên suy ra d | ($x_1 - x_2$). Ta định nghĩa:

$$x' = (x_1 - x_2)/d$$

$$\delta' = (\delta_1 - \delta_2)/d$$

$$p' = (p - 1)/d$$

Khi đó đồngdư thức trở thành:

$$x' \equiv k\,\delta' \pmod{p'}$$

vì UCLN(δ', p') = 1,nên có thể tính:

$$\varepsilon = (\delta')^{-1} \bmod p'$$

Khi đó giá trị k xác định theo modulo p' sẽ là:

$$k = x'\,\varepsilon \bmod p'$$

Phương trình này cho d giá trị có thể của k

$$k = x'\,\varepsilon + i\,p' \bmod p$$

với i nào đó, $0 \le i \le d-1$. Trong số d giá trị có có thế này, có thể xác định được một giá trị đúng duy nhất qua việc kiểm tra điều kiện

$$\gamma \equiv \alpha^k \pmod{p}$$

5.3. Chuẩn chữ kí số.

Chuẩn chữ kí số(DSS) là phiên bản cải tiến của sơ đồ chữ kí Elgamal. Nó được công bố trong Hồ Sơ trong liên bang vào ngày 19/5/94 và được làm

chuẩn voà 1/12/94 tuy đã được đề xuất từ 8/91. Trước hết ta sẽ nêu ra những thay đổi của nó so với sơ đồ Elgamal và sau đó sẽ mô tả cách thực hiện nó.Trong nhiều tinh huống, thông báo có thể mã và giải mã chỉ một lần nên nó phù hợp cho việc dùng với hệ mật bất kì (an toàn tại thời điểm được mã). Song trên thực tế, nhiều khi một bức điện được dùng làm một tài liệu đối chứng, chẳng hạn như bản hợp đồng hay một chúc thư và vì thế cần xác minh chữ kí sau nhiều năm kể từ lúc bức điện được kí. Bởi vậy, điều quan trọng là có phương án dự phòng liên quan đến sự an toàn của sơ đồ chữ kí khi đối mặt với hệ thống mã. Vì sơ đồ Elgamal không an toàn hơn bài toán logarithm rời rạc nên cần dung modulo p lớn. Chắc chắn p cần ít nhất là 512 bít và nhiều người nhất trí là p nên lấy p=1024 bít để có độ an toàn tốt.

Tuy nhiên, khi chỉ lấy modulo p =512 thì chữ kí sẽ có 1024 bít. Đối với nhiều ứng dụng dùng thẻ thông minh thì cần lại có chữ kí ngắn hơn. DSS cải tiến sơ đồ Elgamal theo hướng sao cho một bức điện 160 bít được kí bằng chữ kí 302 bít song lại p = 512 bít. Khi đó hệ thống làm việc trong nhóm con Z_n^* kích thước 2^{160}. Độ mật của hệ thống dựa trên sự an toàn của việc tìm các logarithm rời rạc trong nhóm con Z_n^*.

Sự thay đổi đầu tiên là thay dấu " - " bằng "+" trong định nghĩa δ, vì thế:

$$\delta = (x + \alpha\gamma)k^{-1} \bmod (p-1)$$

thay đổi kéo theo thay đổi điều kiện xác minh như sau:

$$\alpha^x \beta^\gamma \equiv \gamma^\delta \pmod p \quad (6.1)$$

Nếu UCLN $(x + \alpha\gamma, p-1) = 1$thì $\delta^{-1} \bmod (p-1)$ tồn tại và ta có thể thay đổi điều kiện (6.1) như sau:

$$\alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv \gamma \pmod p \quad (6.2)$$

Đây là thay đổi chủ yếu trong DSS. Giả sử q là số nguyên tố 160 bít sao cho q | (q-1) và α là căn bậc q của một modulo p. (Dễ dàng xây dựng một α như vậy: cho α_0 là phần tử nguyên thuỷ của Z_p và định nghĩa $\alpha = \alpha_0^{(p-1)/q} \bmod p$).

Khi đó β và γ cũng sẽ là căn bậc q của 1. vì thế các số mũ Bất kỳ của α, β và γ có thể rút gọn theo modulo q mà không ảnh hưởng đến điều kiện xác minh (6.2). Điều rắc rối ở đây là γ xuất hiện dưới dạng số mũ ở vế trái của (6.2) song không như vậy ở vế phải. Vì thế, nếu γ rút gọn theo modulo q thì cũng phải rút gọn toàn bộ vế trái của (6.2) theo modulo q để thực hiện phép kiểm tra. Nhận xét rằng, sơ đồ (6.1) sẽ không làm việc nếu thực hiện rút gọn theo modulo q trên (6.1). DSS được mô tả đầy đủ trong sơ đồ dưới.

Chú ý cần có $\delta \not\equiv 0 \pmod q$ vì giá trị $\delta^{-1} \bmod q$ cần thiết để xác minh chữ kí (điều này tương với yêu cầu UCLN(δ, p-1) =1 khi biến đổi (6.1) thành (6.2). Nếu Bob tính $\delta \equiv 0 \pmod q$ theo thuật toán chữ kí, anh ta sẽ loại đi và xây dựng chữ kí mới với số ngẫu nhiên k mới. Cần chỉ ra rằng, điều này có thể không gần vấn đề trên thực tế: xác xuất để $\delta \equiv 0 \pmod q$ chắc sẽ xảy ra cở 2^{-160} nên nó sẽ hầu như không bao giờ xảy ra.

Dưới đây là một ví dụ minh hoạ nhỏ

Chuẩn chữ kí số.

Giả sử p là số nguyên tố 512 bít sao cho bài toán logarithm rời rạc trong Z_p không giải được, cho p là số nguyên tố 160 bít là ước của (p-1). Giả thiết $\alpha \in Z_p$ là căn bậc q của 1modulo p: Cho $p = Z_p$. $a = Z_q \times Z_p$ và định nghĩa :

$$A = \{(p,q,\alpha ,a,\beta) : \beta \equiv \alpha^a \pmod p\}$$

các số p, q, α và β là công khai, có a mật.

Với K = (p,q,α ,a,β)và với một số ngẫu nhiên (mật) k ,$1 \le k \le q-1$, ta định nghĩa:

$$sig_k (x,k) = (\gamma ,\delta)$$

trong đó $\gamma = (\alpha^k \bmod p) \bmod q$

và $\delta = (x + a\gamma)k^{-1} \bmod q$

Với $x \in Z_p$ và $\gamma ,\delta \in Z_q$, qua trình xác minh sẽ hoàn toàn sau các tính toán :

$$e_1 = x\delta^{-1} \bmod q$$

$$e_2 = \gamma\delta^{-1} \bmod q$$

$$ver_k(x, \gamma, \delta) = true \Leftrightarrow (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma$$

Ví dụ:

Giả sử q =101, p = 78 q+1 =7879.3 là phần tử nguyên thuỷ trong Z_{7879} nên ta có thể lấy: $\alpha = 3^{78} \bmod 7879 = 170$

Giả sử a =75, khi đó :

$$\beta = \alpha^a \bmod 7879 = 4576$$

Bây giờ giả sữ Bob muốn kí bức điện x = 1234 và anh ta chọn số ngẫu nhiên k =50, vì thế :

$$k^{-1} \bmod 101 = 99$$

khi đó $\gamma = (170^{30} \bmod 7879) \bmod 101$

$= 2518 \bmod 101$

$= 94$

và $\delta = (1234 + 75 \times 94) \bmod 101$

$= 96$

Chữ kí (94, 97) trên bức điện 1234 được xác minh bằng các tính toán sau:

$$\delta^{-1} = 97^{-1} \bmod 101 = 25$$

$e_1 = 1234 \times 25 \bmod 101 = 45$

$e_2 = 94 \times 25 \bmod 101 = 27$

$(170^{45}\ 4567^{27} \bmod 7879) \bmod = 2518 \bmod 101 = 94$

vì thế chữ kí hợp lệ.

Khi DSS được đề xuất năm 1991, đã có một vài chỉ trích đưa ra. Một ý kiến cho rằng, việc xử lý lựa chọn của NIST là không công khai. Tiêu chuẩn đã được Cục An ninh Quốc gia (NSA) phát triển mà không có sự tham gia của khôi công nghiệp Mỹ. Bất chấp những ưu thế của sơ đồ, nhiều người đã đóng chặt cửa không tiếp nhận.

Còn những chỉ trích về mặt kĩ thuật thì chủ yếu là về kích thước modulo p bị cố định = 512 bít. Nhiều người muốn kích thước này có thể thay đổi được nếu cần, có thể dùng kích cỡ lớn hơn. Đáp ứng những đòi hỏi này, NIST đã chọn tiêu chuẩn cho phép có nhiều cở modulo, nghĩa là cỡ modulo bất kì chia hết cho 64 trong phạm vi từ 512 đến 1024 bít.

Một phàn nàn khác về DSS là chữ kí được tạo ra nhanh hơn việc xác minh nó. Trong khi đó, nếu dùng RSA làm sơ đồ chữ kí với số mũ xác minh công khai nhỏ hơn (chẳng hạn = 3) thì có thể xác minh nhanh hơn nhiều so với việc lập chữ kí. Điều này dẫn đến hai vấn đề liên quan đến những ứng dụng của sơ đồ chữ kí:

1.Bức điện chỉ được kí một lần, song nhiều khi lại cần xác minh chữ kí nhiều lần trong nhiều năm. Điều này lại gợi ý nhu cầu có thuật toán xác minh nhanh hơn.

2.Những kiểu máy tính nào có thể dùng để kí và xác minh ? Nhiều ứng dụng, chẳng hạn các thẻ thông minh có khả năng xử lý hạn chế lại liên lạc với máy tính mạnh hơn. Vi thế có nhu cầu nhưng thiết kế một sơ đồ để có thực hiện trên thẻ một vài tính toán. Tuy nhiên, có những tình huống cần hệ thống mình tạo chữ kí, trong những tình huống khác lại cần thẻ thông minh xác minh chữ kí. Vì thế có thể đưa ra giải pháp xác định ở đây.

Sự đáp ứng của NIST đối với yêu cầu về số lần tạo xác minh chữ kí thực ra không có vấn đề gì ngoài yêu cầu về tốc độ, miễn là cả hai thể thực hiện đủ nhanh.