Chương trình KC-01:
Nghiên cứu khoa học
phát triển công nghệ thông tin
và truyền thông

Đề tài KC-01-01:
Nghiên cứu một số vấn đề bảo mật và an toàn thông tin cho các mạng dùng giao thức liên mạng máy tính IP

Báo cáo kết quả nghiên cứu

ĐẢM BẢO TOÁN HỌC CHO CÁC HỆ MẬT

Quyển 3C: "Nghiên cứu xây dựng thuật toán mã khối an toàn hiệu quả"

HÀ NỘI-2004

Báo cáo kết quả nghiên cứu

ĐẢM BẢO TOÁN HỌC CHO CÁC HỆ MẬT

Quyển 3C: "Nghiên cứu xây dựng thuật toán mã khối an toàn hiệu quả"

Chủ trì nhóm nghiên cứu
T.S Trần Văn Trường

MỤC LỤC

CHƯƠNG 1: MỞ ĐẦU VỀ MÃ KHỐI

I. GIỚI THIỆU CHUNG

I.1. Hệ mã khối khóa bí mật

Một khối lượng lớn các thông tin được truyền trên các kênh thông tin và mạng máy tính hiện nay đang ngày càng gia tăng đặc biệt đòi hỏi cần phải được bảo vệ khỏi các dò dỉ không mong muốn, tức là đảm bảo tính bí mật, đồng thời cũng cần phải được bảo vệ tránh sự giả mạo và sự từ chối trách nhiệm, tức là đảm bảo tính xác thực. Kỹ thuật mật mã được phát triển và vận dụng để đảm bảo cả tính bí mật và tính xác thực đó.

Các hệ mật hiện nay được chia thành hai loại: hệ mật khóa bí mật và hệ mật khóa công khai. Trong hệ mật khóa bí mật, những người sử dụng hợp pháp (người gửi và người nhận) phải chia sẻ một khóa bí mật chung và khóa đó không được biết đối với thám mã đối phương. Trong hệ mật khóa công khai, người sử dụng hợp pháp chỉ cần các thông tin trung thực công khai nào đó. Mặc dù các hệ mật khóa công khai tỏ ra là lý tưởng đối với nhiều ứng dụng mật mã, nhưng tốc độ thấp và giá thành cao đã ngăn cản việc sử dụng chúng trong nhiều trường hợp. Trong phần này chúng ta chỉ thảo luận về các hệ mật khóa bí mật.

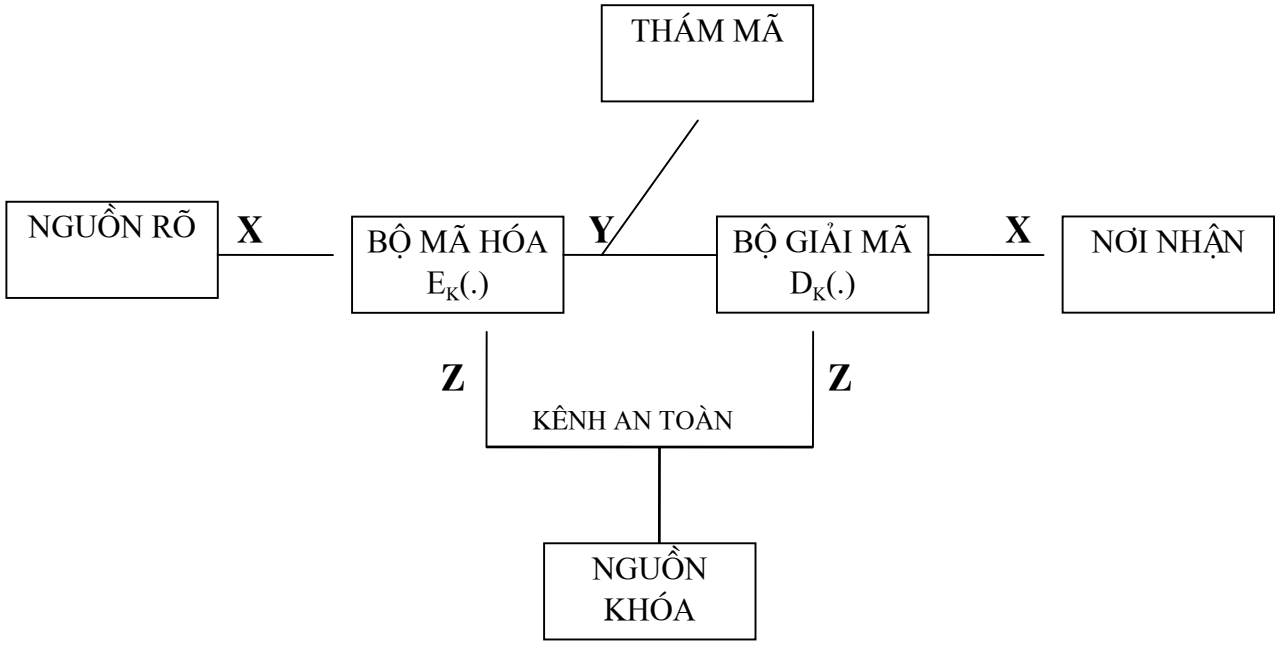
Chúng ta sẽ sử dụng mô hình hệ mật của Shannon trong Hình 1.1.

Trong mô hình này, khóa bí mật Z được phân phối tới người gửi và người nhận theo một kênh an toàn. Khóa này sau đó được sử dụng để mã hóa bản rõ X thành bản mã Y bởi người gửi và được dùng để giải mã bản mã Y thành bản rõ X bởi người nhận. Bản mã được truyền trên kênh không an toàn, và chúng ta giả thiết là thám mã đối phương luôn có thể truy nhập để nhận được các bản mã. Tất nhiên thám mã không thể truy nhập được tới khóa bí mật. Hệ mật khóa bí mật như thế được gọi là hệ mật đối xứng để phân biệt với hệ mật khóa công khai không đối xứng trong đó các khóa khác nhau được sử dụng bởi người mã và người dịch. Chú ý rằng X, Y, và Z trong mô hình này là các biến ngẫu nhiên. Trong mô hình này chúng ta cũng luôn giả thiết bản rõ X và khóa Z là độc lập thống kê.

Các hệ mật khóa bí mật thường được chia thành các hệ mã khối và hệ mã dòng. Đối với mã khối bản rõ có dạng các khối "lớn" (chẳng hạn 128-bit) và dãy các khối đều được mã bởi cùng một hàm mã hóa, tức là bộ

mã hóa là một hàm không nhớ. Trong mã dòng, bản rõ thường là dãy các khối "nhỏ" (thường là 1-bit) và được biến đổi bởi một bộ mã hóa có nhớ.

Các hệ mã khối có ưu điểm là chúng có thể được chuẩn hóa một cách dễ dàng, bởi vì các đơn vị xử lý thông tin hiện này thường có dạng block như bytes hoặc words. Ngoài ra trong kỹ thuật đồng bộ, việc mất một block mã cũng không ảnh hưởng tới độ chính xác của việc giải mã của các khối tiếp sau, đó cũng là một ưu điểm khác của mã khối.



HÌNH 1.1: MÔ HÌNH HỆ MẬT KHÓA BÍ MẬT

Nhược điểm lớn nhất của mã khối là phép mã hóa không che dấu được các mẫu dữ liệu: các khối mã giống nhau sẽ suy ra các khối rõ cũng giống nhau. Tuy nhiên nhược điểm này có thể được khắc phục bằng cách đưa vào một lượng nhỏ có nhớ trong quá trình mã hóa, tức là bằng cách sử dụng cách thức móc xích khối mã (CBC-Cipher Block Channing mode) trong đó hàm mã hóa không nhớ được áp vào tổng XOR của block rõ và block mã trước đó. Phép mã lúc này có kiểu cách kỹ thuật như mã dòng áp dụng đối với các khối "lớn".

Giả sử F_2 là trường Galois hai phần tử. Ký hiệu F_2^m là không gian véc tơ các bộ m-tuples các phần tử của F_2. Trong phần này chúng ta giả thiết không mất tổng quát rằng, bản rõ X, bản mã Y lấy các giá trị trong

không gian véc tơ F_2^m, còn khóa Z lấy giá trị trong không gian véc tơ F_2^k. Như vậy m-là độ dài bít của các khối rõ và mã, còn k-là độ dài bit của khóa bí mật.

Định nghĩa 1.1. Hệ mã khối khóa bí mật là một ánh xạ E: F_2^m x $S_z \rightarrow F_2^m$, sao cho với mỗi $z \in S_z$, E(., z) là một ánh xạ có ngược từ F_2^m vào F_2^m.

Hàm có ngược E(., z) được gọi là hàm mã hóa tương ứng với khóa z. Ánh xạ nghịch đảo của E(., z) được gọi là hàm giải mã tương ứng với khóa z và sẽ được ký hiệu là D(., z). Chúng ta viết Y = E(X, Z) đối với một mã khối có nghĩa là bản mã Y được xác định bởi bản rõ X và khóa bí mật Z theo ánh xạ E. Tham số m được gọi là độ dài khối còn tham số k được gọi là độ dài khóa của hệ mã khối đó. Cỡ khóa đúng của hệ mã khối được xác định bởi số $k_t = \log_2 (\#(S_z))$ bit. Như vậy độ dài khóa sẽ bằng cỡ khóa đúng nếu và chỉ nếu $S_z = F_2^k$, tức là mọi bộ k-bit nhị phân đều là một khóa có hiệu lực. Chẳng hạn đối với chuẩn mã dữ liệu DES, độ dài khóa là k = 64 bit, trong khi cỡ khóa đúng của nó là k_t = 56 bit. Chú ý rằng ở đây ta xem xét các mã khối có độ dài khối mã bằng độ dài khối rõ.

I.2. Độ an toàn của các hệ mã khối

Như đã nói ở trên, một mã khối được sử dụng nhằm bảo vệ chống sự dò dỉ không mong muốn của bản rõ. Nhiệm vụ của thám mã đối phương là phá hệ mã này theo nghĩa anh ta có thể mở ra được các bản rõ từ các bản mã chặn bắt được. *Một hệ mã là bị phá hoàn toàn nếu như thám mã có thể xác định được khóa bí mật đang sử dụng và từ đó anh ta có thể đọc được tất cả các thông báo một cách dễ dàng như là một người dùng hợp pháp. Một hệ mã là bị phá thực tế nếu thám mã có thể thường xuyên mở ra được các bản rõ từ các bản mã nhận được, nhưng vẫn chưa tìm ra được khóa.*

Độ an toàn luôn gắn với các đe dọa tấn công. Như đã nói ở trên, chúng ta giả sử rằng kẻ tấn công luôn có thể truy nhập tới mọi thứ được truyền thông qua kênh không an toàn. Tuy nhiên, có thể có các thông tin khác đối với thám mã. Khả năng tính toán của thám mã phải luôn được xem xét trước khi xem xét độ an toàn của một mã có thể bị truy nhập.

I.2.1. Các kiểu tấn công

Một giả thiết được chấp nhận phổ biến nhất trong mật mã đó là thám mã đối phương luôn có thể truy nhập hoàn toàn tới các bản mã được truyền trên kênh không an toàn. Một giả thiết đã được chấp nhận khác nữa là:

Giả thiết Kerckhoff: Thám mã đối phương là được biết toàn bộ chi tiết của quá trình mã hóa và giải mã chỉ trừ giá trị khóa bí mật.

Giả thiết Kerckhoff suy ra rằng độ an toàn của một hệ mật khóa bí mật chỉ còn phụ thuộc vào chính khóa mật mà thôi. Dưới giả thiết Kerckhoff, các tấn công có thể được phân loại theo các tri thức của thám mã như sau:

- *Tấn công chỉ biêt bản mã*: thám mã đối phương không biết thêm tí thông tin gì ngoài bản mã nhận được.

- *Tấn công bản rõ đã biết*: Thám mã đối phươnng biết thêm một vài cặp Rõ/Mã đối với khóa đang dùng.

- *Tấn công bản rõ lựa chọn*: Thám mã đối phươnng có thể đạt được các bản mã tương ứng với các bản rõ ấn định đặc biệt bất kỳ đối với khóa đang dùng.

Tấn công bản rõ lựa chọn là tấn công mạnh nhất trong các tấn công trên. Nếu một hệ mã là an toàn chống lại tấn công bản rõ lựa chọn thì nó cũng an toàn trước các tấn công khác. Trong thực tế, ta nên dùng hệ mã có độ an toàn chống lại tấn công bản rõ lựa chọn, ngay cả khi thám mã đối phương hiếm có cơ hội thu lượm được thông tin gì đó hơn so với tấn công chỉ biết bản mã.

I.2.2. Độ an toàn vô điều kiện và độ an toàn tính toán

Độ an toàn của một hệ mật phụ thuộc rất lớn vào khả năng tính toán của thám mã đối phương. Một hệ mật được gọi là an toàn vô điều kiện nếu nó an toàn chống lại thám mã đối phương có khả năng tính toán vô hạn. Độ an toàn vô điều kiện cũng được gọi là độ an toàn lý thuyết liên quan tới tính không thể phá được của một hệ mật. Một hệ mật là an toàn chống lại đối phương có khả năng tính toán bị hạn chế nào đó được gọi là an toàn tính toán. Độ an toàn tính toán cũng được gọi là độ an toàn thực tế, liên quan tới tính khó phá của một hệ mật. Tất cả các hệ mật an toàn vô điều kiện đều là không có tính thực tế vì lý do sẽ được nói dưới đây. Tuy nhiên cũng không có một hệ mật thực tế nào là đã được chứng minh là an toàn theo nghĩa tính toán.

Độ an toàn vô điều kiện

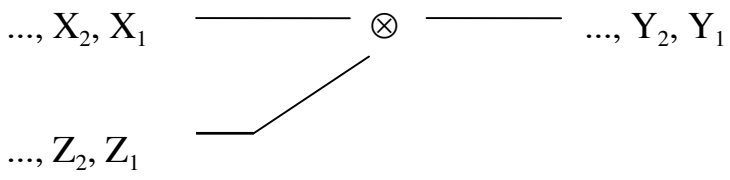
Mặc dù trong hầu hết các ứng dụng độ an toàn vô điều kiện là không cần thiết và cũng là không thể thực hiện được trên thực tế, nhưng nghiên cứu về độ an toàn vô điều kiện cho chúng ta nhiều gợi ý có ích cho việc thiết kế và sử dụng các hệ mật thực tế. Chẳng hạn lý do cơ bản của hệ mã dòng

đó là độ mật hoàn thiện được cung cấp bởi hệ thống đệm một lần "one-time-pad".

Định nghĩa 1.2 (Shannon 1949): Một hệ mật sẽ cung cấp độ mật hoàn thiện nếu các khối rõ và các khối mã là độc lập thống kê.

Khả năng thực thi hệ mật bí mật hoàn thiện đã được cho thấy bởi Shannon trong bài báo của ông ta năm 1949. Hệ "Mã nhóm khóa dùng một lần"sau đây (được mô tả trong ví dụ 1) cung cấp một hệ mật bí mật hoàn thiện như thế. Ý tưởng sử dụng hệ thống khóa dùng một lần đầu tiên được đề xuất bởi Vernam trong năm 1926. Mã Vernam thường được gọi là hệ mật một lần "one-time-pad". Mặc dù trong một thời gian dài người ta tin rằng hệ mật một là là không thể bị phá, nhưng phải đến công trình của Shannon mới chứng minh được tính bí mật hoàn thiện của nó.

Ví dụ 1: (hệ mã khối nhóm khóa dùng một lần): Xét hệ mã khối cho trong Hình 1.2, ở đây $\otimes$ là phép toán nhóm định nghĩa trên tập hợp F_2^m. Hệ mã này có độ bí mật hoàn thiện nếu khóa được chọn ngẫu nhiên đều và độc lập với mỗi khối rõ.



Hình 1.2: Hệ mã khối nhóm khóa dùng một lần. Các khóa Z_i là được chọn ngẫu nhiên đều và độc lập.

Hệ thống bí mật hoàn thiện thường là không thực tế, bởi vì Shannon đã cho thấy một lượng khóa không giới hạn cần phải có nếu như ta cho phép một lượng thông báo không hạn chế. Tuy nhiên, ý tưởng của hệ mật hoàn thiện thiết lập nên một nguyên lý đã biết trong thực tế mật mã là để đảm bảo độ an toàn thì nên thay khóa một cách thường xuyên.

Độ an toàn vô điều kiện cũng có thể đạt được bằng cách nén dữ liệu. Shannon đã định nghĩa một hệ mật là lý tưởng chặt nếu với một khóa cố định, dãy các khối mã không cho một thông tin gì về khóa. Shannon cũng chú ý rằng nếu bản rõ không còn độ dư, tức là nếu tất cả các khối rõ

là độc lập ngẫu nhiên đều thì hầu hết các mã khối đều là lý tưởng chặt, tức là hệ mật như thế sẽ an toàn chống lại tấn công chỉ biết bản mã ngay cả khi cùng một khóa được sử dụng để mã cho nhiều bản rõ. Không may thay, chưa có một kỹ thuật nén dữ liệu hiện đã biết là có thể đạt được việc nén hoàn hảo như vậy. Nhưng công trình của Shannon cũng lại thiết lập nên một nguyên lý khác nữa trong mật mã là để đảm bảo an toàn thì các dữ liệu rõ nên là ngẫu nhiên như có thể làm được. Điều này có thể thực hiện hoặc bằng cách nén dữ liệu hoặc bằng các hệ thay thế đồng cấu.

Hệ mật lý tưởng chặt chỉ an toàn chống lại tấn công chỉ biết bản mã. Tuy nhiên không phải mọi hệ lý tưởng chặt là đều có thể chống lại tấn công bản rõ đã biết (hay bản rõ lựa chọn). Chẳng hạn, xét hệ mã khối nhóm trong Hình 1.2. Ngay cả khi cùng một khóa dùng để mã nhiều lần, hệ này vẫn là lý tưởng chặt nếu các khối rõ là độc lập phân bố đều. Tuy nhiên, cho trước một cặp Rõ/Mã ta có thể dễ dàng xác định được khóa. Như vậy hệ mật này dù là an toàn vô điều kiện chống lại tấn công chỉ biết bản mã, nhưng nó có thể dễ dàng bị phá trong tấn công bản rõ đã biết nếu khóa mật được sử dụng hơn một lần.

Đối với một hệ mã khối sử dụng theo cách thức không phải một lần, tức là khi một khóa được sử dụng để mã nhiều khối rõ, thì tính bí mật hoàn thiện định nghĩa bởi Shannon không bao giờ đạt được do các khối mã như nhau sẽ suy ra các khối rõ cũng giống nhau. Độ an toàn vô điều kiện chống lại tấn công bản rõ đã biết (hoặc bản rõ lựa chọn) khi khoá được sử dụng nhiều hơn một lần đã được xem xét bởi Massey. Từ nghiên cứu của Massey gợi ý rằng để tăng cường độ mật chống lại tấn công bản rõ đã biết (hoặc bản rõ lựa chọn) nên thay đổi khoá thường xuyên, và mỗi khoá cần tương ứng với các ánh xạ 1-1 ngẫu nhiên.

Độ an toàn tính toán

Trong thực tế không kẻ tấn công nào có khả năng tính toán vô hạn. Độ an toàn của một hệ mật thực tế phụ thuộc vào tính không thể phá hệ mã đó về mặt lý thuyết mà đúng hơn là phụ thuộc độ khó thực tế của các tấn công. Một hệ mật được gọi là an toàn tính toán nếu độ khó của tấn công tối ưu vượt quá khả năng tính toán của thám mã. Shannon đã mô tả độ khó của tấn công như thế (tấn công chỉ biết bản mã) bởi đặc trưng W(n) xem như là khối lượng công việc đòi hỏi để xác định khóa khi n-bản mã là được biết. Ta cũng có thể xem xét W(n) đối với các kiểu tấn công khác. Trong suốt phần này, chúng ta sử dụng từ "độ phức tạp" để mô tả độ khó như thế. Độ phức tạp của một tấn công hiểu một cách chung chung là số

trung bình các phép toán (thao tác) dùng trong tấn công đó. Chú ý rằng một hệ mã là an toàn tính toán có nghĩa là độ phức tạp của tấn công tối ưu vượt quá khả năng tính toán của thám mã đối phương. Để chứng minh một hệ mật là an toàn tính toán cần phải chỉ ra được cận dưới hữu ích về độ phức tạp của việc giải quyết một bài toán tính toán nào đó. Hiện tại, điều này là không thể đối với tất cả các bài toán tính toán. Do vậy, trong thực tế, việc đánh giá độ an toàn của mọt hệ mật phụ thuộc vào độ phức tạp của tấn công tốt nhất cho tới hiện tại. Một mã khối thực tế được xem là an toàn tính toán nếu không có tấn công đã biết nào có thể làm tốt hơn so với tấn công vét cạn khóa. Trong tấn công vét cạn khóa chỉ biết bản mã trên một mã khối, mỗi một khóa có thể đều được thử để giải mã của một hoặc nhiều hơn các khối mã chặn bắt được cho tới khi nào một khóa cho kết quả khối rõ có thể đọc được. Độ phức tạp của tấn công này, xem như là số các phép giải mã thử, về mặt trung bình sẽ bằng 2^{k_t-1} đối với một hệ mã khối có cỡ khóa đúng là k_t. Tấn công vét cạn khóa là một tấn công "brute-force" nó có thể áp vào hệ mã khối bất kỳ. Như vậy một hệ mã khối muốn an toàn thì cỡ khóa đúng của nó là phải đủ lớn để tạo cho tấn công vét cạn khóa là không thể thực hiện được.

I.2.3. Độ phức tạp xử lý và độ phức tạp dữ liệu của một tấn công cụ thể

Độ phức tạp của một tấn công được chia ra làm hai phần: *độ phức tạp dữ liệu và độ phức tạp xử lý*. Độ phức tạp dữ liệu là lượng dữ liệu đầu vào cần cho tấn công đó trong khi độ phức tạp xử lý là lượng các tính toán cần để xử lý dữ liệu như thế. Thành phần dominant-trội hơn thường được mô tả như là độ phức tạp của tấn công này. Chẳng hạn, trong tấn công vét cạn khóa, lượng dữ liệu đầu vào cần cho tấn công này là số các khối mã chặn bắt được (hoặc số các cặp rõ/mã trong tấn công bản rõ đã biết), nói chung đó là một số lượng rất nhỏ so với số các phép toán (trung bình cần 2^{k_t-1} phép giải mã với các khóa khác nhau trong việc tìm ra khóa đúng) cần thiết của tấn công này. Do vậy độ phức tạp của tấn công duyệt khóa thường chính là độ phức tạp xử lý. Ví dụ khác là tấn công vi sai của Biham và Shamir, đó là kiểu tấn công bản rõ lựa chọn. Đối với tấn công vi sai độ phức tạp vượt trội lên bởi số các cặp rõ/mã cần trong tấn công đó, trong khi số các tính toán sử dụng trong tấn công này lại tương đối nhỏ. Do đó độ phức tạp của tấn công vi sai thực chất là độ phức tạp dữ liệu.

Nói chung đối với một mã khối độ dài khối m-bit và cỡ khóa đúng là k_t-bit, độ phức tạp dữ liệu của tấn công bản rõ đã biết (hoặc bản rõ lựa chon) có thể được đo bởi số các cặp rõ/mã đã biết (hay lựa chọn) cần cho

tấn công này, nhiều nhất là 2^m là số toàn bộ các cặp như thế đối với một khóa cố định. Độ phức tạp xử lý có thể bị chặn trên bởi số 2^{k_t} phép mã hóa do đặc tính của tấn công vét cạn khóa và do nói chung thao tác mã hóa là được tính toán nhanh, hiệu quả. Như vậy chúng ta có thể nói rằng một hệ mật là an toàn tính toán nếu như không có tấn công nào trên hệ mật đó có độ phức tạp dữ liệu nhỏ hơn đáng kể 2^m phép mã và độ phức tạp xử lý nhỏ hơn đáng kể 2^{k_t} phép mã hóa. Một hệ mật được gọi là an toàn thực tế chống lại một tấn công cụ thể nếu với tấn công này, độ phức tạp dữ liệu vào khoảng 2^m cặp rõ/mã hoặc độ phức tạp xử lý là vào khoảng 2^{k_t} phép mã hóa. Đối với thám mã, độ phức tạp dữ liệu là loại độ phức tạp bị động, anh ta phải chờ người sử dụng tạo ra các cặp rõ /mã cho anh ta. Mặt khác, độ phức tạp xử lý lại là kiểu độ phức tạp chủ động và có thể khắc phục nói chung bằng cách sử dụng nhiều máy tính mạnh.

I.2.4. Các tham số của mã khối

Độ dài khối m

Để một hệ mã khối là an toàn, độ dài khối m của nó phải đủ lớn ngăn cản các tấn công phân tích thống kê, tức là để không cho đối phương thu được thông tin có ích nào về khối rõ nào đó thường xuất hiện nhiều hơn các khối rõ khác. Ngoài ra độ dài khối m cũng phải được chọn sao cho số các cặp rõ/mã mà đối phương có thể thu nhận được trong thực tế phải nhỏ hơn rất nhiều so với 2^m.

Khi độ dài khối của hệ mã trở nên lớn thì độ phức tạp của ứng dụng cũng tăng theo. Dù rằng độ phức tạp trong ứng dụng chọn ngẫu nhiên hàm có ngược là tăng theo cỡ mũ so với độ dài khối, nhưng chỉ có hàm đơn giản mới xuất hiện ngẫu nhiên, điều này tạo cơ hội phục vụ hàm mã hóa thực tế khi độ dài khối m là lớn. Tuy nhiên, Shannon đã chỉ ra rằng sự dễ dàng trong tính toán các hàm mã hóa E(., z) và hàm giải mã D(., z) với mọi z không suy ra được việc giải tìm khóa z từ các phương trình y = E(x, z) và x = D(y, z) sẽ là dễ dàng khi biết x và y.

Độ dài khóa k và cỡ khóa đúng k_t

Để hệ mã khối an toàn chống lại tấn công vét cạn khóa, cỡ khóa đúng cần phải đủ lớn sao cho 2^{k_t-1} phép mã hóa cần cho tấn công này là vượt xa khả năng của thám mã. Mặt khác, độ dài khóa k cũng cần nhỏ ở mức nào đó sao cho việc tạo, phân phối và lưu trữ khóa có thể thực hiện được hiệu quả và an toàn. Chẳng hạn, DES có độ dài khóa là 64 bít, còn cỡ khóa đúng là 56 bit. Tấn công vét cạn khóa là không thể nhưng cũng không là

quá xa vời. Nhiều gợi ý muốn tăng cỡ khóa đúng của DES. Chẳng hạn, mở rộng cỡ khóa dúng của DES tới 128 bit bằng phép mã bội ba dùng hai khóa xem là một cách thức chuẩn để sử dụng DES.

I.3. Nguyên lý thiết kế mã khối

Một hệ mã khối tốt là phải "khó phá và dễ sử dụng". Cả hai hàm mã hóa E(., z) và hàm giải mã D(., z) nên dễ dàng tính toán. Còn việc giải khóa z từ y = E(x, z) và x = D(y, z) nên là bài toán khó. Nguyên lý thiết kế cho một hệ mã khối có thể chia thành các nguyên lý ứng dụng và các nguyên lý an toàn.

I.3.1. Nguyên lý thiết kế chung về độ an toàn

Chỉ có hai nguyên lý thiết kế được chấp nhận chung đối với các mã an toàn thực tế là các nguyên lý về độ méo (confusion) và độ khuyếch tán (diffusion) đã được gợi ý bởi Shannon.

Nguyên lý về độ méo (confusion):

Sự phụ thuộc của khóa trên bản rõ và bản mã nên phải phức tạp sao cho nó không có ích gì đối với thám mã. Chẳng hạn, phương trình nhị phân mô tả mã khối nên là phi tuyến và phức tạp sao cho để việc giải khóa z từ x và y = E(x, z) là không thể.

Nguyên lý về độ khuyếch tán (diffusion):

Với mỗi khóa cụ thể hàm mã hóa không nên có sự phụ thuộc thống kê nào giữa các cấu trúc đơn giản trong bản rõ và các cấu trúc đơn giản trong bản mã và rằng không có quan hệ đơn giản nào giữa các hàm mã hóa khác nhau. Nguyên lý khuyếch tán đòi hỏi, chẳng hạn một hệ mã khối cần được thiết kế có tính đầy đủ-hay hoàn thiện "complete", tức là mỗi bit rõ và mỗi bit khóa đều ảnh hưởng tới mỗi bit mã.

I.3.2 Nguyên lý thiết kế cho ứng dụng

Một hệ mã khối có thể ứng dụng cả phần cứng và phần mềm. Trong ứng dụng cứng thường được thực hiện bởi các chíp VLSI có tốc độ cao. Trong ứng dụng mềm phải có tính mềm dẻo và giá thành thấp. Trên cơ sở đặc tính khác nhau của phần cứng và phần mềm, các nguyên lý thiết kế cho mã khối cũng chia thành hai phần.

Nguyên lý thiết kế cho ứng dụng mềm

Sử dụng khối con: Các thao tác mã khối nên thực hiện trên các khối con có độ dài tự nhiên cho phần mềm là 8, 16, 32 bit. Hoán vị bit là khó thực hiện trong phần mềm nên tránh.

Sử dụng các phép toán đơn giản: Các thao tác mã trên các khối con nên chọn dễ dàng cho ứng dụng với các tập lệnh cơ sở của các bộ xử lý chuẩn chẳng hạn như phép cộng, phép nhân, phép dịch ...

Nguyên lý thiết kế cho ứng dụng phần cứng

Sự tương tự trong phép mã hóa và phép giải mã: Quá trình mã hóa và giải mã nên chỉ khác nhau ở cách sử dụng khóa mật sao cho cùng một thiết bị có thể sử dụng được cho cả phép mã hóa và phép giải mã.

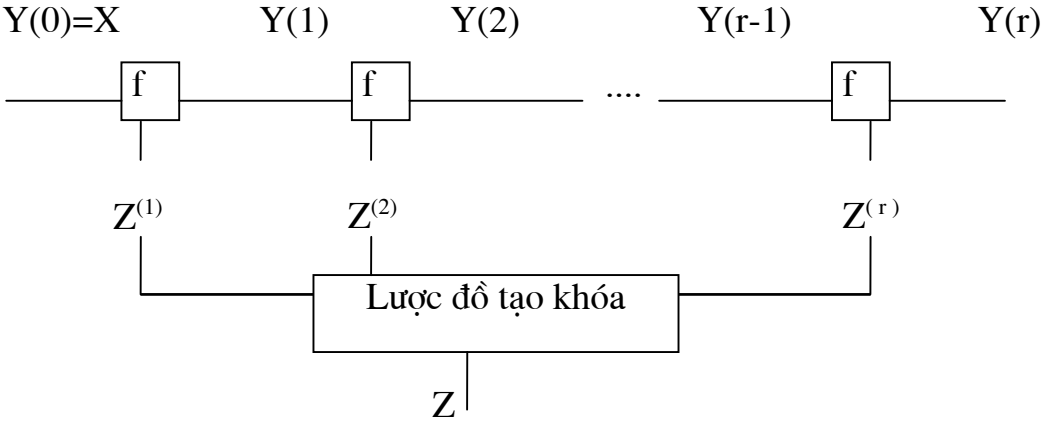
Cấu trúc đều

Hệ mã nên có cấu trúc môđun đều để có thể dễ ứng dụng công nghệ VLSI.

I.4. Các mã lặp

I.4.1 Mã lặp và hàm vòng

Một mã khối được gọi là mã lặp nếu nó dựa trên cơ sở lặp một hàm đơn giản f một vài lần như thấy trong Hình 1.3. Mỗi phép lặp được gọi là một vòng. Đầu ra của mỗi vòng là hàm của đầu ra của vòng trước đó và của một khóa con được thiết kế từ khóa bí mật đầy đủ bởi một lược đồ tạo khóa. Một mã khối khóa bí mật như thế với r-phép lặp được gọi là một mã lặp r-vòng. Hàm f được gọi là hàm vòng. Ví dụ, DES là một mã lặp 16-vòng.



Hình 1.3. Một mã lặp r-vòng với hàm vòng f

Phương pháp lặp được sử dụng trong thiết kế mã khối là do nó bao hàm tất cả các nguyên lý thiết kế cơ bản đã nêu trên. Một hàm vòng đơn giản có thể được ứng dụng hiệu quả, trong khi phép lặp của một hàm vòng được chọn hợp lý có thể cung cấp độ méo và độ khuyếch tán cần thiết. Sau này ta thấy rằng trong thám vi sai đối với một mã Markov độ phức tạp dữ liệu của tấn công này sẽ tăng theo hàm mũ với số vòng lặp trong khi độ phức tạp ứng dụng chỉ tăng cỡ tuyến tính.

I.4.2. Cấu trúc của mã lặp tương tự E/D

Trong thực tế, hầu hết các đề xuất mã khối đều tuân thủ qui tắc bất thành văn đó là nên cấu trúc hệ mã sao cho thuận tiện cho quá trình mã dịch. Cấu trúc Feistel là một trong những kiểu có cấu trúc tương tự E/D. Quá trình giải mã hoàn toàn giống như quá trình mã hoá, chỉ khác là dùng các khoá con với thứ tự ngược lại. Gần tương tự như thế, đó là hệ mã IDEA cũng có cấu trúc kiểu tương tự E/D.

II. CÁC CẤU TRÚC MÃ KHỐI CƠ BẢN.

II.1 Cấu trúc mã Feistel.

Phần lớn các hệ mã khối trên thế giới hiện nay là dựa trên cấu trúc mã-dịch Feistel có các đặc tính cơ bản sau:

* Độ dài của mỗi khối (block) rõ bằng độ dài của mỗi khối mã, và là một số

chẵn m= 2. L.

* Bản rõ được chia thành các khối P = (x_0, x_1) có độ dài 2. L, và $|x_0| = |x_1| =$ L.

* Khoá k là một tập khoá con: k_1, k_2 , .., k_n.

* Mỗi k_i được tương ứng với một phép biến đổi F_i trên khối cỡ L.

* Bản rõ P được mã hoá theo n-bước như sau:

Bản rõ: $P = (x_0, x_1)$

Vòng 1: $(x_0, x_1) \rightarrow (x_1, x_2)$

Vòng 2: $(x_1, x_2) \rightarrow (x_2, x_3)$

Vòng i: $(x_{i-1}, x_i) \rightarrow (x_i, x_{i+1})$

Vòng n: $(x_{n-1}, x_n) \rightarrow (x_n, x_{n+1})$

Bản mã là: $C = (x_{n+1}, x_n)$

Trong đó $x_{i+1} = x_{i-1} \oplus F_i(x_i)$

Với cấu trúc mã hoá trên đây, quá trình dịch mã sẽ rất đơn giản: Giữ nguyên các thao tác như quá trình mã hoá, chỉ cần thay đổi thứ tự sử dụng khoá và các hàm vòng tương ứng:

$$k_n, k_{n-1}, .., k_1$$

$$F_n, F_{n-1}, .., F_1.$$

Nhận xét: a/- Cấu trúc mã Feistel trên đây rất thuận tiện cho mã dịch đảm bảo tốc độ nhanh và tiện lợi cho việc cứng hoá các chương trình mã dịch khối. Các hàm vòng F_i có thể có cấu trúc hoàn toàn giống nhau, tức là $F_i = F$, miễn sao chúng là hàm có tính chất mật mã tốt, và do đó sẽ càng thuận tiện cho thao tác mã dịch.

b/ Qua mô hình cấu trúc mã dịch Feistel trên có thể thấy ngay các dạng khoá coi là yếu như sau (với giả thiết $F_i \equiv F$):

- Khoá yếu là các khoá có dạng:

$$k_n = k_1;$$

$$k_{n-1} = k_2;$$

$$k_{n-2} = k_3;$$

Tức là D(.) = E(.), hay là $E^2 = I$. Như vậy thám mã chỉ cần mã hoá chính bản mã thu được là sẽ có được bản rõ cần tìm.

- Cặp khoá nửa yếu là các cặp khoá có dạng:

$$k_n(A) = k_1(B);$$

$$k_{n-1}(A) = k_2(B);$$

$$k_{n-2}(A) = k_3(B);$$

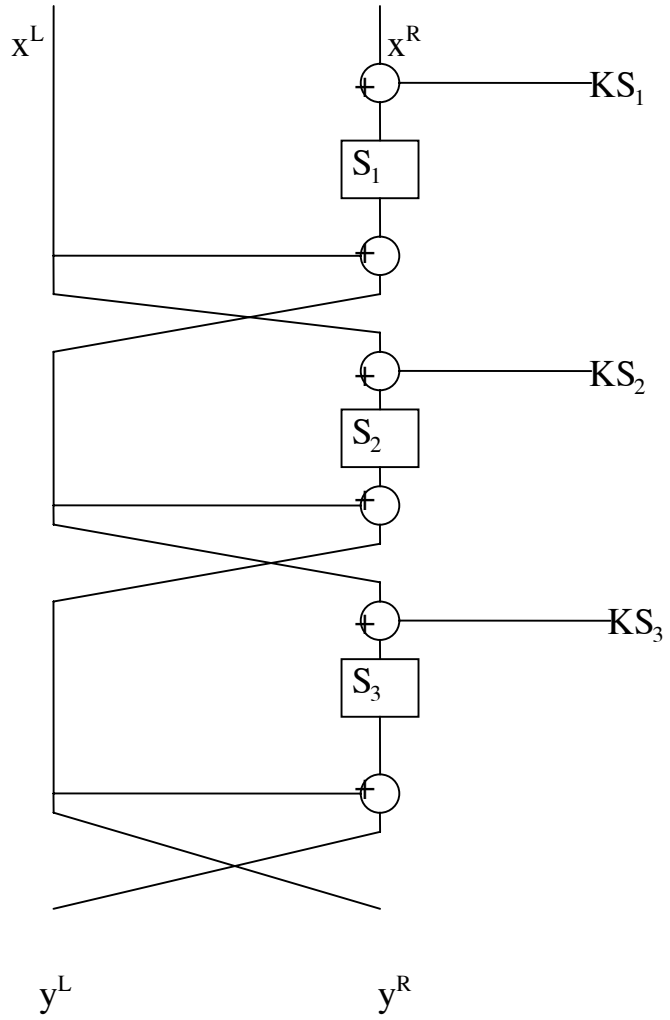
Điều này có nghĩa là thám mã có thể dùng thao tác mã hoá của người B để giải mã các bản mã của người A và ngược lại. Tức là ta có

$$E_A = D_B, \text{ và } E_B = D_A.$$

Tất nhiên các dạng khoá trên đây là không được phép sử dụng trong các mô hình mã khối tương ứng.

II.2. Cấu trúc Matsui

Cấu trúc mã khối của Matsui là cấu trúc có tính truy hồi, gồm ba lớp: lớp trong cùng, lớp giữa và lớp ngoài cùng. Mỗi một lớp đều có cùng một hình thức biến đổi xáo trộn dữ liệu, được mô tả dưới các hình sau.



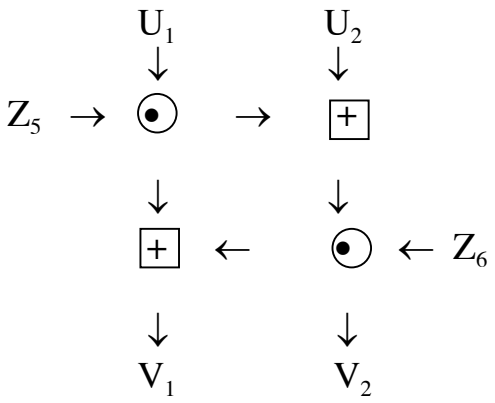
Hình 1.4: Cấu trúc lớp trong cùng (xem là một F_i)

Cấu trúc lớp giữa giống như lớp trong cùng chỉ khác là các hộp S_i được thay bởi hàm F_i như đã mô tả trên, ngoài ra mỗi F_i cũng được tác động với một khoá con (như là khoá đại diện cho lớp trong cùng của nó). Tiếp theo là cấu trúc lớp ngoài cùng cũng giống như lớp giữa chỉ khác là các hàm F_i được thay bởi hàm FO_i.

Với cấu trúc truy hồi Matsui, các dữ liệu ở nửa đi vào hộp thế hay hàm biến đổi sẽ không được chuyển nguyên thành dữ liệu bên nửa trái của vòng mới. Điều này làm cho cấu trúc này có độ đo vi sai và độ đo độ lệch tuyến tính tốt hơn so với cấu trúc Feistel. Tuy nhiên chúng phải trả giá là không có cấu trúc mã dịch đối xứng như cấu trúc Feistel, và ứng dụng cứng hoá có vẻ là khó hơn so với cấu trúc Feistel.

II.3. Cấu trúc cộng-nhân

Cấu trúc cộng-nhân có thể xem như là một trong các kiểu hạt nhân cấu tạo nên các hàm vòng, trong đó hoàn toàn sử dụng các phép toán số học tương đối đơn giản và được chọn lọc cẩn thận. Một số cấu trúc biến đổi khác mà ta đã làm quen như các hộp nén, các phép hoán vị, các phép dịch vòng, chúng đã được sử dụng trong DES, trong hệ mã dữ liệu Xôviết... Cấu trúc cộng-nhân được đề xuất bởi J. L. Massey và X. Lai khi họ xây dựng nên một chuẩn mã dữ liệu mới là PES và sau đó được cải tiến đổi tên thành IDEA. Hình 1.4 cho ta mô hình của cấu trúc cộng-nhân



Hình 1.5 : Sơ đồ cấu trúc cộng-nhân (MA).

Trong sơ đồ trên thì các phép toán ⊙ và ⊞ là các phép nhân môdulo hoặc cộng môdulo trên các nhóm tương ứng với không gian đầu vào của các hạng tử: U_1, U_2 là các véc tơ đầu vào, V_1, V_2 là các véc tơ đầu ra, Z_1, Z_2 là các khoá.

Theo các tác giả của thuật toán, thực hiện biến đổi theo sơ đồ cấu trúc cộng-nhân trên đây sẽ đảm bảo tính chất khuyếch tán tốt cho phép mã hoá.

II.4 Giới thiệu một số loại hình mã khối.

a/ Chuẩn mã dữ liệu Xô viết (GOST).

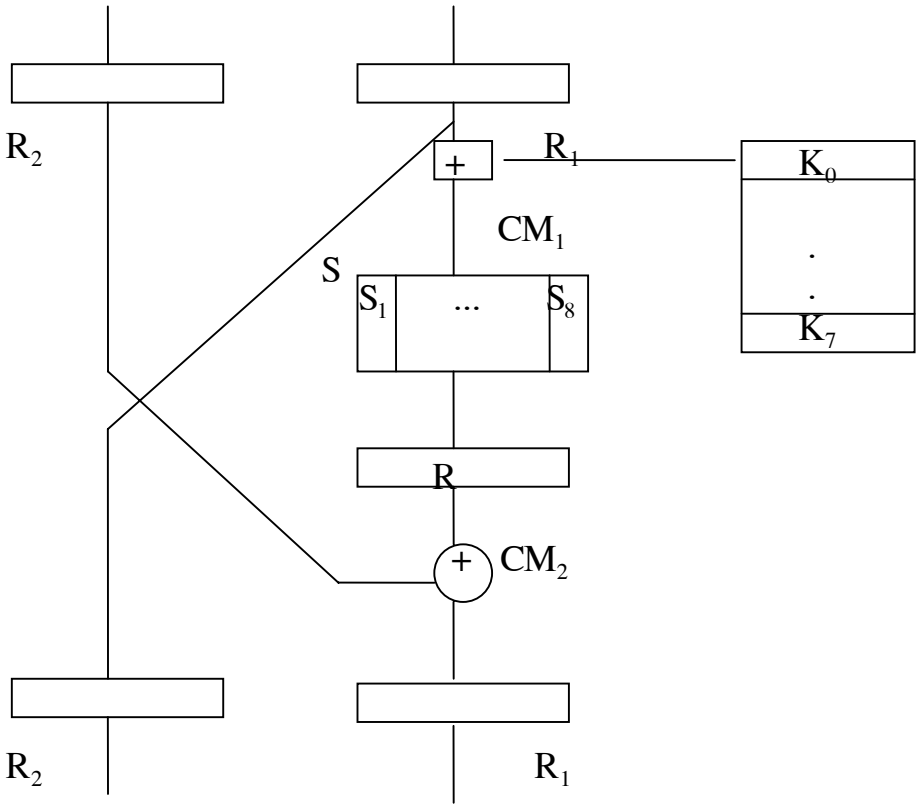
Ngoài chuẩn mã dữ liệu DES đã được biết, chuẩn mã dữ liệu Xô viết là một trong những kiểu đặc trưng của hệ mã khối sử dụng cấu trúc Feistel

với hạt nhân là các hộp thế, phép dịch vòng, kết hợp với các phép toán số học như phép XOR và phép cộng môdulo.

Mô hình mã dịch của chuẩn mã dữ liệu Xô viết cũng gần tương tự như DES, tuy nhiên nó dùng một độ dài khoá lớn hơn là 256 bit để mã hoá bản rõ 64-bit. Ngoài ra, tám hộp thế của chuẩn mã dữ liệu Xô viết là hoàn toàn bí mật, không được công khai như trong DES. Dưới đây là mô hình cụ thể.

Thuật toán GOST bao gồm 32 vòng lặp, trong đó mỗi một vòng lặp được cho trong Hình 1.6. Khoá bí mật là một xâu bít độ dài 256. Hộp cộng CM_1 là phép cộng môdulo 2^{32}, còn hộp cộng CM_2 là phép cộng XOR. Thao tác R là phép dịch vòng về bên trái đi 11 vị tri (theo hướng bít có nghĩa lớn nhất), còn S_1, S_2, .. ., S_8 là các hộp thế với không gian đầu vào và đầu ra đều là $GF(2^4)$, các phép tương ứng trong các hộp thế này cũng được giữ bí mật. Với 32 vòng lặp thuật toán GOST sử dụng khoá bí mật tương ứng theo thứ tự sau:

K_0,..., K_7, K_0,..., K_7,K_0,..., K_7,K_7,..., K_0.

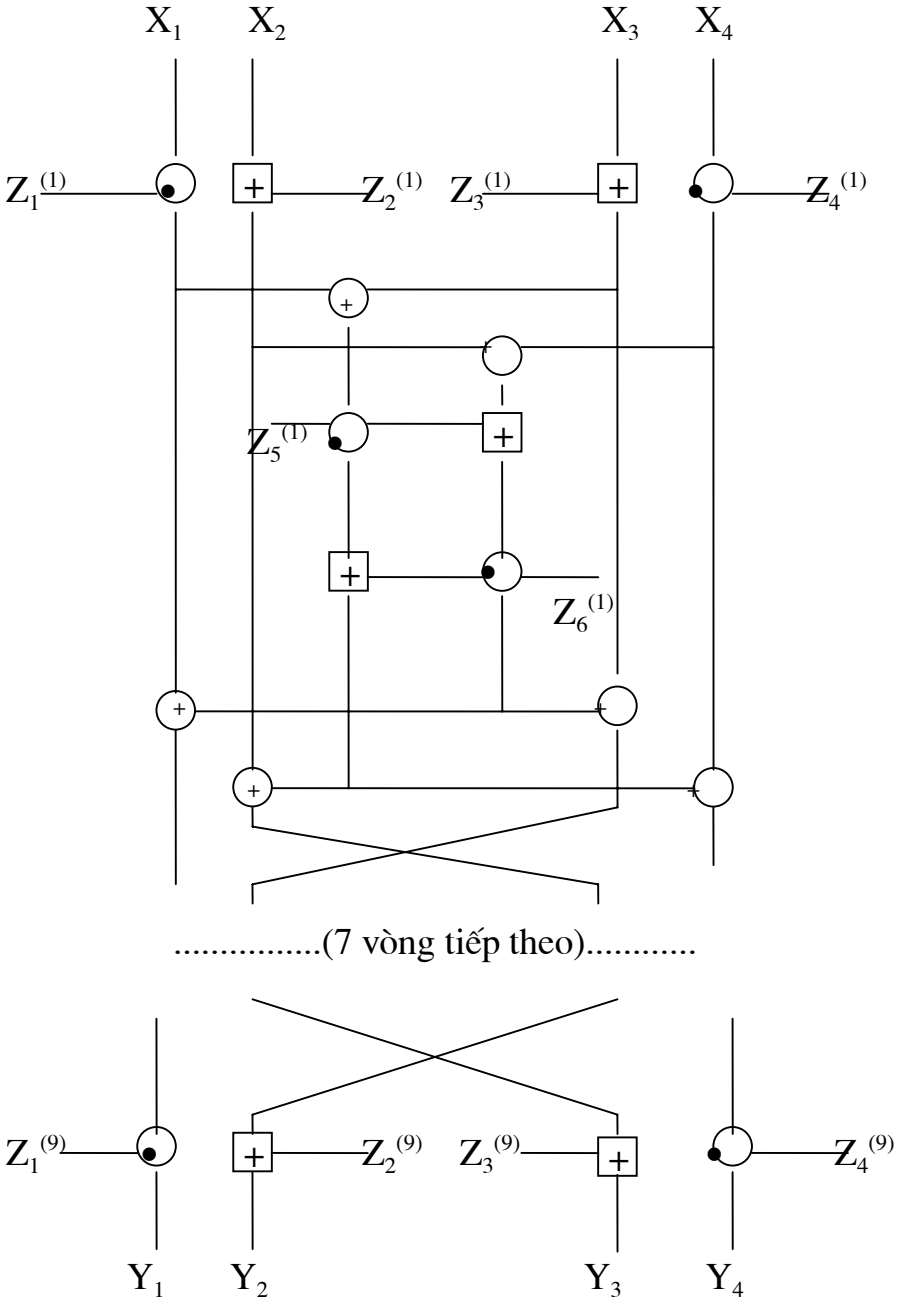


Hình 1.6: Sơ đồ một vòng lặp của thuật toán GOST.

Sơ bộ có thể thấy thuật toán GOST tuân thủ cấu trúc mã Feistel, quá trình mã dịch thực hiện dễ dàng, đồng thời có một số yếu tố cần lưu ý đó là độ dài khoá bí mật khá lớn cùng với việc giữ kín các hộp thế trong sơ đồ mã hoá.

b/ Thuật toán mã dữ liệu quốc tế IDEA.

Thuật toán mã dữ liệu IDEA là một thuật toán điển hình chỉ sử dụng các phép toán số học thông qua việc liên kết các cấu trúc cộng-nhân. Sơ đồ cụ thể của thuật toán được cho trong Hình 1.7 dưới đây.



Hình 1.8: Sơ đồ thuật toán IDEA.

Thuật toán mã khối IDEA thực hiện sơ đồ mã dịch khối, biến đổi các khối rõ 64-bit thành các khối mã 64-bit, nhờ sử dụng một khoá mật dài 128-bit. Các phép biến đổi trong thuật toán đều là các phép toán số học, trong đó $\oplus$ là phép XOR, $\boxplus$ là phép cộng môdulo 2^{16}, $\odot$ là phép nhân mod $(2^{16}+1)$ với qui ước 0000_{hex} bằng 2^{16}. Cấu trúc cộng-nhân được sử dụng thông qua các khoá $Z_5^{(i)}$, và $Z_6^{(i)}$. Tám vòng lặp được thực hiện giống nhau, còn vòng thứ chín chỉ thực hiện một nửa để đảm bảo qui cách mã dịch được dễ dàng. 52 bộ khoá con 16-bit được tạo từ 128-bít khoá chính theo một sơ đồ dễ thực hiện, quá trình dịch mã được thực hiện theo thứ tự ngược lại của các khoá con. Có thể thấy IDEA được thiết kế mã dịch hướng word, và nó đã được các tác giả J.L Massey, X. Lai và S. Murphy cải tiến từ hệ PES nhằm tránh tấn công vi sai.

Trên đây là hai hệ mã khối đại diện cho hai cấu trúc điển hình là cấu trúc Feistel và cấu trúc cộng-nhân. Hệ mã khối đại diện cho cấu trúc truy hồi Matsui đó là mã khối MISTY được thiết kế bởi chính tác giả Matsui [24]. Ngoài ra các hệ mã khối hiện nay thường phối hợp các cấu trúc cơ bản này để phát huy đặc tính tốt của mỗi loại hình.

CHƯƠNG 2: THÁM MÃ KHỐI

Để tiến tới xây dựng được một hệ mã khối an toàn hiệu quả, có nhiều công việc cần phải làm. Một số những công việc quan trọng khởi đầu cho quá trình đó trong điều kiện hiện nay là cần thiết nghiên cứu những phương pháp thám mã khối điển hình từ đó rút ra những đặc trưng an toàn cơ bản của một hệ mã khối. Chương này tập trung nghiên cứu lý thuyết về các phương pháp thám mã khối cơ bản như thám mã vi sai, thám mã vi sai bậc cao, thám mã tuyến tính và các dạng đặc biệt của thám mã tuyến tính, thám mã nội suy, thám mã khoá quan hệ.. chủ yếu áp dụng trên chuẩn mã dữ liệu DES. Về mặt lý thuyết chúng tôi chỉ nêu những nguyên tắc thám mã cơ bản đối với mã khối (dựa trên chuẩn mã dữ liệu DES) mà không trình bày chi tiết thuật toán (vì có thể tìm thấy trong nhiều tài liệu khác). Phần thực hành, chúng tôi tập trung nghiên cứu khai thác phương pháp thám mã phi tuyến dựa trên ý tưởng thám mã tuyến tính để xây dựng thuật toán thám hệ DES rút gọn 8-vòng nhằm tìm đủ 56 bít khoá của chúng. Phần cuối của chương nêu lên những đặc trưng cơ bản của một hệ mã khối an toàn-hiệu quả.

I. THÁM MÃ VI SAI ĐỐI VỚI DES VÀ CÁC HỆ MÃ KHỐI LẶP DES-LIKE

I.1. Mô hình hệ DES

DES là một thuật toán mã khối, thực hiện mã hoá một xâu bít rõ độ dài 64-bit bằng một khoá độ dài 56-bit. Bản mã nhận được cũng là một xâu bít độ dài 64. Thuật toán tiến hành theo ba giai đoạn:

- Với một xâu rõ x độ dài 64-bit, một xâu bit x_0 được xây dựng bằng cách hoán vị các bit của x theo một hoán vị cố định ban đầu IP. Ta viết $x_0 = IP(x) = L_0R_0$, trong đó L_0 là 32 bit đầu, R_0 là 32 bit cuối của x_0.

- Sau đó x_0 được biến đổi qua 16 vòng lặp theo một hàm xác định để được các xâu L_iR_i, $1 \le i \le 16$ theo qui tắc

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

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

ở đây F là hàm sẽ được mô tả sau, còn $K_1, K_2, .., K_{16}$ là các xâu bít độ dài 48 được tính như là hàm của khoá K (thực tế, mỗi K_i là một phép hoán vị

bít của K đã được chọn trước). K_1, K_2,.., K_{16} sẽ tạo thành một bảng khoá có thể truy cập mã dịch dễ dàng.

- Áp phép hoán vị ngược IP^{-1} cho xâu bit $R_{16}L_{16}$ ta sẽ thu được bản mã y. Tức là $y = IP^{-1}(R_{16}L_{16})$.

* Hàm F có hai biến vào: biến thứ nhất A là một xâu bit độ dài 32, biến thứ hai J là một xâu bit độ dài 48. Đầu ra của F là một xâu bit độ dài 32. Các bước biến đổi của hàm F được mô tả như sau:

+ Biến thứ nhất A được mở rộng thành xâu bit độ dài 48 theo hàm mở rộng cố định E.

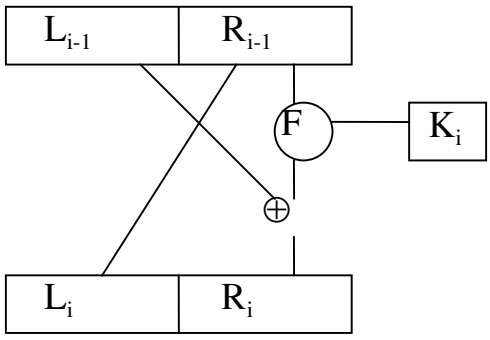
+ Tính $E(A) \oplus J$ và viết thành kết quả một chuỗi 8 xâu 6 bit: $B = B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7\ B_8$.

+ Dùng 8 hộp nén S_1, S_2.., S_8 để biến đổi 8 xâu độ dài 6-bit B_1, B_2,.., B_8 thành

8 xâu độ dài 4-bit C_1, C_2,.. ,C_8.

+ Xâu bit $C = C_1\ C_2\ ..\ C_8$ có độ dài 32 được hoán vị theo một hoán vị cố định P, được kết quả P(C) chính là F(A,J).

*



Hình 2.1: Một vòng của DES

* ***Nhận xét***

Tất cả các thao tác mã hoá của DES đều là phép biến đổi tuyến tính, trừ phép biến đổi qua các hộp nén S_i. Do đó độ an toàn của DES chủ yếu dựa

vào tính phi tuyến của các hộp nén này. Ở đây, chúng ta sẽ bàn kỹ hơn một chút về các hộp nén đó.

Theo thiết kế, mỗi hộp S_i, $1 \leq i \leq 8$, là một bảng 4 x 16 cố định. Trong đó, mỗi một hàng của nó là một hoán vị của các số nguyên từ 0 đến 15. Mỗi một hộp S_i, có thể xem là một phép biến đổi từ không gian V_2^6 vào không gian V_2^4, với $V_2 = \{0,1\}$. Cách thức thực hiện chúng như sau.

Ký hiệu $B_i = b_1b_2b_3b_4b_5b_6$ là xâu đầu vào 6-bit của hộp S_i. Hai bít b_1b_6 xác định biểu diễn nhị phân của hàng thứ r của S_i ($0 \leq r \leq 3$), và 4-bit $b_2b_3b_4b_5$ xác định biểu diễn nhị phân của cột c của hộp S_i ($0 \leq c \leq 15$). Khi đó, $S_i(B_i)$ được thiết lập từ phần tử $S_i(r,c)$ nằm trên hàng r và cột c của S_i. Phần tử này viết dưới dạng nhị phân là một xâu bit C_i độ dài 4, chính là đầu ra của B_i qua hộp nén S_i: $C_i = S_i(B_i)$.

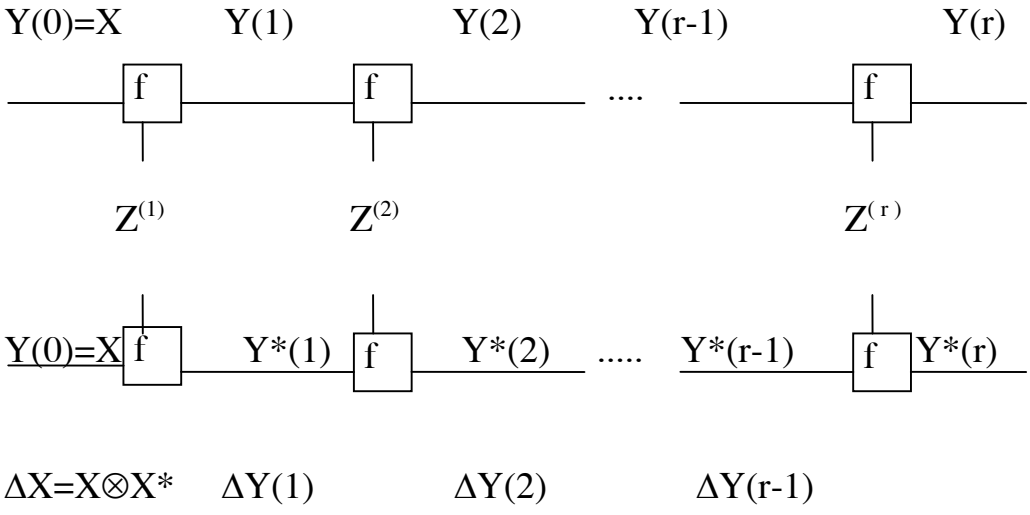
Cấu trúc và tiêu chuẩn thiết kế các hộp nén ở đây là rất quan trọng để đảm bảo độ an toàn cao cho hệ DES, và có thể thấy rằng các tấn công mạnh nhất hiện nay đều khai thác triệt để các yếu điểm tiềm tàng của các hộp nén. Để tiện tham khảo, chúng tôi liệt kê ra đây hai hộp nén S_1 và S_5 sẽ được sử dụng trong các tấn công sau này.

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

S_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

I.2. Thám mã vi sai đối với các mã khối lặp

Thông thường các hệ mã khối khoá bí mật được thiết kế *dựa trên cơ sở lặp một hàm tương đối yếu về mặt mật mã nào đó* (để có tốc độ cao, thiết kế đơn giản..). Mỗi một phép lặp được gọi là một vòng. Đầu ra của mỗi vòng là hàm của đầu ra của vòng trước và một khoá con được thiết kế từ khoá bí mật ban đầu theo lược đồ tạo khoá. Một mã khối khoá bí mật với r - phép lặp như thế được gọi là một mã lặp r - vòng. Các hệ DES hay IDEA đều là mã khối lặp theo quan niệm trên. Phần này ta sẽ xem xét nguyên lý tấn công vi sai đối với mã lặp có dạng tổng quát như hình 2.2



Hình 2.2. Phép mã hoá một cặp bản rõ với mã lặp r -vòng.

Với các ký hiệu như trong hình 2.2, ta có Y = f(X, Z) là một hàm vòng sao cho với mỗi khoá con Z, hàm f(., Z) thiết lập một tương ứng 1-1 giữa đầu vào X và đầu ra Y.

Vi sai ΔX giữa hai bản rõ (hay hai bản mã) X và X* được xác định bởi

$$\Delta X = X \otimes X^{*-1},$$

ở đây ⊗ ký hiệu là phép toán nhóm đã xác định nào đó trên tập các bản rõ (= tập các bản mã), và X^{*-1} là nghịch đảo của phần tử X* trong nhóm

đó. Hàm vòng f(X, Z) được gọi là yếu nếu cho trước một vài bộ ba (ΔX, Y, Y*) là có thể xác định được khoá Z. Từ cặp các phép mã hoá chúng ta có thể xác định được một dãy các vi sai ΔY(0), ΔY(1),..., ΔY(r), ở đâyY(0) = X, và Y*(0) = X* là căp bản rõ, cũng vậy ΔY(0) = ΔX, còn Y(i) và Y*(i) là đầu ra của vòng thứ - i, chúng cũng là đầu vào của vòng thứ - (i+1). Khoá con cho vòng thứ -i ký hiệu là $Z^{(i)}$. Trong các lập luận sau này ta luôn giả thiết X ≠ X*.

Thám mã vi sai dựa trên yếu tố là hàm vòng f trong một mã khối lặp là một hàm yếu về mật mã. Cụ thể là nếu cặp bản mã là được biết và bằng cách nào đó có thể đạt được vi sai của cặp đầu vào tại vòng cuối cùng của mã khối lặp đó, thì tấn công vi sai có thể áp dụng để đạt được hoặc xác định được khoá hay một phần khoá con tại vòng cuối cùng. Trong thám mã vi sai, điều đó được thực hiện bằng cách chọn cặp bản rõ (X, X) có vi sai là α sao cho vi sai ΔY(r -1) của cặp đầu vào tại vòng cuối cùng sẽ lấy giá trị cụ thể β với một xác suất cao. Vì có xác suất cao nên các khoá con tại vòng cuối được giải từ bộ ba (ΔY(r -1), Y (r), Y*(r)) sẽ thường tập trung vào một* ***số*** *phần tử có khả năng nhất. Từ các phần tử xuất hiện nhiều nhất, thám mã sẽ quyết định để tìm ra khoá con đúng tại vòng cuối cùng. Từ khoá con của vòng cuối này, ta có thể xác định được lại khoá bí mật ban đầu (nếu lược đồ tạo khoá là đơn giản).*

Định nghĩa 2.1: Một vi sai i - vòng là một cặp (α, β), ở đây α là vi sai của một cặp bản rõ khác nhau X và X*, và β là một vi sai có thể đối với kết quả đầu ra vòng thứ - i là Y(i) và Y*(i). Xác suất của vi sai i - vòng (α, β) là xác suất có điều kiện sao cho β là vi sai ΔY(i) của cặp bản mã sau i - vòng với điều kiện cho trước cặp bản rõ (X, X*) có vi sai ΔX = α khi bản rõ X và các khoá con $Z^{(1)}$, ...$Z^{(i)}$ là ngẫu nhiên độc lập phân bố đều. Ta ký hiệu xác suất của vi sai này là P(ΔY(i) = β | ΔX = α).

**** Thủ tục cơ bản của tấn công vi sai đối với một mã khối lặp r-vòng:***

1/ Tìm một vi sai (r - 1) - vòng (α, β) sao cho xác suất

$$P(\Delta Y(i-1) = \beta \mid \Delta X = \alpha)$$

là cực đại, hoặc gần cực đại.

2/ Lấy bản rõ X một cách ngẫu nhiên đều và tính toán X* sao cho vi sai ΔX giữa X và X* là α. Tiến hành mã hoá X và X* dưới một khoá bí mật cụ thể cần tìm Z (mà đối phương đang sử dụng). Từ các bản mã kết quả Y(r) và Y*(r), tìm mỗi một giá trị có thể của khoá con $Z^{(r)}$ của vòng cuối cùng tương ứng với vi sai đã định trước $\Delta Y(i - 1) = \beta$ (tức là sử dụng bộ ba

($\Delta Y(i - 1)$ đặt bằng β, Y (r), Y*(r) để tính toán tìm $Z^{(r)}$). Thêm 1 vào bộ đếm số lần xuất hiện của mỗi giá trị có thể của khoá con $Z^{(r)}$.

3/ Lặp bước 2/ cho tới khi một hoặc nhiều giá trị của khoá con $Z^{(r)}$ là được đếm nhiều hơn hẳn các giá trị khác. Lấy ra khoá con được đếm nhiều nhất hoặc một tập nhỏ các khoá có số đếm lớn nhất. Sau đó việc quyết định khoá đúng $Z^{(r)}$ thuộc về người thám mã.

Chú ý rằng trong tấn công vi sai, tất cả các khoá con là cố định và chỉ có bản rõ là được lấy ngẫu nhiên. Tuy nhiên, trong tính toán xác suất vi sai, ta luôn giả thiết bản rõ và tất cả các khoá con là độc lập ngẫu nhiên đều. Do đó chúng ta cần tạo một giả thiết sau.

**** Giả thiết về tính tương đương ngẫu nhiên (Stochastic Equivalence):***

Đối với một vi sai (r - 1) - vòng (α, β), ta có

$$P(\Delta Y(i - 1) = \beta \mid \Delta X = \alpha) \approx P(\Delta Y(i - 1) = \beta \mid \Delta X = \alpha, Z^{(1)} = \omega_1, ..., Z^{(r-1)} = \omega_{r-1})$$

với hầu hết tập giá trị khoá con (ω_1 ,...,ω_{r-1}).

Do có $2^m - 1$ giá trị của $\Delta Y(i - 1)$, nên chúng ta có thể rút ra kết luận sau:

Giả sử Giả thiết về tính tương đương ngẫu nhiên là đúng, khi đó một mã lặp r - vòng với tập khoá con độc lập sẽ có thể bị tổn thương đối với tấn công vi sai nếu và chỉ nếu hàm vòng là yếu và tồn tại một vi sai (r - 1) - vòng (α, β), sao cho $P(\Delta Y(i - 1) = \beta / \Delta X = \alpha) >> 2^{-m}$, ở đây m là độ dài của khối mã.

Ký hiệu Comp(r) là độ phức tạp của một tấn công thám mã với mã lặp r -vòng, xem như là số phép mã hoá cần sử dụng trong tấn công. Khi đó ta có thể chứng minh kết quả sau.

Định lý 2.2. (Cận dưới về độ phức tạp của tấn công vi sai đối với một mã lặp r -vòng)

Giả sử Giả thiết về tính tương đương ngẫu nhiên là đúng, khi đó độ phức tạp của tấn công vi sai sẽ thoả mãn đánh giá

$$Comp(r) \geq 2/\left(p_{\max} - \frac{1}{2^m - 1} \right)$$

ở đây $p_{max} = \max_\alpha \max_\beta (P(\Delta Y(i - 1) = \beta \mid \Delta X = \alpha)$,

trong đó m - là độ dài khối mã.

Thực tế nếu $p_{max} \approx 1/ (2^m - 1)$, thì tấn công vi sai là không thành công.

Chứng minh: Chú ý rằng giá trị tính trước β của vi sai ΔY(i - 1) ít nhất phải lấy nhiều hơn một lần so với giá trị trung bình khi chọn ngẫu nhiên β', nếu như tấn công vi sai thành công. Như vậy, ta có $Tp_{max} \geq (T/ (2^m - 1)) + \mathbf{1}$ là điều kiện cần cho sự thành công sau T phép thử, ở đây mỗi phép thử gồm phép chọn một cặp bản rõ có vi sai α cho trước.

Từ đó ta có

$$2.T.(p_{max} - 1/(2^m - 1)) \geq 2,$$

mà Comp(r) = 2.T (mỗi phép thử có hai phép mã cho cặp bản rõ), nên

$$Comp(r) = 2.T \geq 2/ (p_{max} - 1/ (2^m - 1)) \qquad \text{ĐPCM.}$$

I.3. Sơ bộ về phương pháp tấn công vi sai trên DES

Phương pháp tấn công vi sai (DC) trên DES do Biham và Shamir đề xuất là một trong những phương pháp tấn công nổi tiếng nhất đối với hệ DES. Đây là phép tấn công với bản rõ chọn lọc, và nó đã khai thác triệt để điểm yếu của DES tại các hộp nén. Bây giờ ta sẽ mô tả ý tưởng cơ bản dùng trong tấn công này.

Trước hết, ta sẽ bỏ qua phép hoán vị đầu IP và hoán vị ngược của nó (không ảnh hưởng tới kết quả phân tích mã của chúng ta), khi đó có thể xem L_0R_0 là bản rõ và L_nR_n là bản mã với DES n-vòng.

Phương pháp thám mã vi sai xoay quanh việc so sánh kết quả của phép XOR giữa hai bản rõ với kết quả của phép XOR giữa hai bản mã

tương ứng. Với giả thiết rằng các bản rõ được lấy ngẫu nhiên đều trên không gian các đầu vào có thể, hãy thử xem phân bố của các kết quả phép XOR đầu ra có tuân theo phân bố ngẫu nhiên đều hay không. Nếu bảng phân bố là không đều, thì thám mã có thể lợi dụng để xây dựng phương pháp tấn công lên hệ mật bằng kiểu tấn công bản rõ chọn lọc mà chúng tôi sẽ sơ bộ nêu ra ở đây.

Định nghĩa 2.3:

Giả sử S_j là một hộp nén ($1 \le j \le 8$). Xét một cặp đã xắp xếp của các xâu bit độ dài 6 (ký hiệu là (B_j, B^*_j)). Ta nói rằng XOR vào của S_j là $B_j \oplus B^*_j$ và XOR ra của S_j là $S_j(B_j) \oplus S_j(B^*_j)$.

Với bất kỳ $B'_j \in Z_2^6$, ta đặt

$$\Delta(B'_j) = \{ (B_j, B_j \oplus B'_j) : B'_j \in Z_2^6 \}$$

Định nghĩa 2.4:

Với $1 \le j \le 8$, $B'_j \in Z_2^6$, $C'_j \in Z_2^4$, ta định nghĩa

$$IN_j(B'_j, C'_j) = \{ (B_j \in Z_2^6 : S_j(B_j) \oplus S_j(B_j \oplus B'_j) = C'_j \}$$

và $N_j(B'_j, C'_j) = |IN_j(B'_j, C'_j)|$

Như vậy, $N_j(B'_j, C'_j)$ là số các cặp có XOR vào là B'_j và có XOR ra là C'_j.

Nhớ lại rằng, đầu vào của các hộp nén ở vòng thứ i là $B = E \oplus J$, trong đó $E = E(R_{i-1})$ là giá trị của hàm mở rộng E tác động vào R_{i-1} và $J = K_i$ là các bit khoá của vòng thứ i. Khi đó XOR vào của tất cả 8 hộp nén có thể được tính như sau: $B \oplus B^* = (E \oplus J) \oplus (E^* \oplus J) = E \oplus E^*$

Đến đây ta thấy một điều rất quan trọng là XOR vào không phụ thuộc vào các bít khoá J, nhưng chắc chắn các XOR ra sẽ phụ thuộc các bít khoá này, và các XOR vào của các hộp nén sẽ được tính qua giá trị của hàm mở rộng E đã được biết công khai.

Bây giờ ta viết B, E và J là một dãy liên tiếp 8 xâu 6 bit:

$$B = B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7\ B_8$$

$$E = E_1\ E_2\ E_3\ E_4\ E_5\ E_6\ E_7\ E_8$$

$$J = J_1\ J_2\ J_3\ J_4\ J_5\ J_6\ J_7\ J_8$$

và B*, E* theo cách tương tự. Khi đó, nếu biết các giá trị E_j và E^*_j và giá trị XOR ra của S_j là C'_j, thì chắc chắn rằng

$$E_j \oplus J_j \in IN_j(E'_j, C'_j).$$

Từ đó ta định nghĩa các tập TEST cho các đoạn khoá con J_j như sau

Định nghĩa 2.5:

Với các ký hiệu đã nêu ta xác định

$$TEST_j (E_j, E^*_j, C'_j) = \{ B_j \oplus E_j : B_j \in IN_j(E'_j, C'_j)\}$$

trong đó $E'_j = E_j \oplus E^*_j$

Từ định nghĩa này ta có ngay kết quả sau đây:

Định lý 2.6.

Giả sử E_j và E^*_j là hai xâu vào của hộp nén S_j còn XOR ra của S_j là C_j. Khi đó các bít khoá J_j sẽ nằm trong tập $TEST_j(E_j, E^*_j, C'_j)$.

Bây giờ ta áp dụng các ý tưởng trên đây để mô tả phương pháp tấn công vi sai trên DES

* ***VỚI DES 3 VÒNG***. Ta có thể viết

$$\begin{aligned} R_3 &= L_2 \oplus F(R_2, K_3) \\ &= R_1 \oplus F(R_2, K_3) \\ &= L_0 \oplus F(R_0, K_1) \oplus F(R_2, K_3) \end{aligned}$$

Biểu diễn R^*_3 một cách tương tự và do đó có

$$R'_3 = L'_0 \oplus F(R_0, K_1) \oplus F(R^*_0, K_1) \oplus F(R_2, K_3) \oplus F(R^*_2, K_3).$$

Nếu ta chọn $R'_0 = 00..0$, thì

$$R'_3 = L'_0 \oplus F(R_2, K_3) \oplus F(R^*_2, K_3).$$

Bây giờ, do R'_3, L'_0 đã biết nên có thể tính được

$$F(R_2, K_3) \oplus F(R^*_2, K_3) = R'_3 \oplus L'_0$$

Mặt khác, ta có $F(R_2, K_3) = P(C)$, $F(R^*_2, K_3) = P(C^*)$ và do tính đồng cấu tuyến tính của phép hoán vị P nên ta tính được

$$C' = C \oplus C^* = P^{-1}(R'_3 \oplus L'_0)$$

Đây là XOR ra của 8 hộp nén ở vòng thứ ba.

Còn $R_2 = L_3$ và $R^*_2 = L^*_3$ cũng được biết (chúng là một phần của các bản mã), nên có thể tính được các dữ liệu liên quan đến đầu vào của các hộp nén là $E = E(L_3)$, và $E^* = E(L^*_3)$.

Như vậy ta đã biết E, E* và C' của vòng thứ ba, và từ đó có thể xây dựng các tập $TEST_1$.. $TEST_8$ chứa các giá trị có thể của các bít khoá J_1..J_8.

Chú ý: Trong phương pháp tấn công DES 3 vòng ta sẽ phải dùng tới một số bộ ba E, E* và C' để tìm được duy nhất 48 bít khoá K_3, sau đó tính 56 bít khoá đúng bằng cách vét cạn 2^8 khả năng cho 8 bit khoá còn lại.

Trong mô tả tấn công DES 3 vòng, ta chưa nói nhiều tới sự phân bố không đều của các XOR ra của các hộp nén, mà chỉ chú ý rằng để thực hiện được kiểu tấn công này ta luôn luôn phải tìm ra được một số bộ ba E, E* và C', ở đây E, E* xác định XOR đầu vào còn C' là XOR đầu ra ***tại vòng cuối cùng*** của hệ DES. Điều chú ý này sẽ làm cho ta hiểu rõ trong tấn công DES nhiều vòng.

* ***TẤN CÔNG DES N-VÒNG***.

Để thực hiện tấn công DES n-vòng ta cần khái niệm sau đây

Định nghĩa 2.7:

Cho $n \geq 1$ là một số nguyên. Một đặc trưng n-vòng là một danh sách có dạng:

$L'_0, R'_0, L'_1, R'_1, p_1, \ldots, L'_n, R'_n, p_n$

thoả mãn các tính chất sau:

+ $L'_i = R'_{i-1}$ với $1 \leq i \leq n$,

+ Cho $1 \leq i \leq n$ và giả sử L'_{i-1}, R'_{i-1} và L^*_{i-1}, R^*_{i-1} được chọn sao cho

$L_{i-1} \oplus L^*_{i-1} = L'_{i-1}$ và $R_{i-1} \oplus R^*_{i-1} = R'_{i-1}$.

Giả sử L_i, R_i và L^*_i, R^*_i được tính bằng cách áp dụng một vòng mã hoá của DES. Khi đó xác suất để $L_i \oplus L^*_i = L'_i$ và $R_i \oplus R^*_i = R'_i$ đúng bằng p_i (chú ý rằng xác suất này được tính trên mọi bộ 48- bit $J = J_1$.. J_8 có thể).

Xác suất của đặc trưng này sẽ được xác định bởi giá trị:

$$p = p_1 \times p_2 .. \times p_n.$$

Nhận xét:

Từ định nghĩa trên đây, nếu ta có thể xây dựng được các đặc trưng n-vòng với xác suất đủ lớn thì khi đó việc tìm được các cặp đúng cùng với các cặp XOR vào và XOR ra tương ứng ở vòng cuối cùng (thứ - n) sẽ có nhiều khả năng hơn, và do đó có thể sử dụng các dữ liệu tại vòng cuối cùng này để tìm lại được các bít khoá K_n tương tự như tấn công DES 3-vòng như đã nói trên.

Nếu các hộp nén có phân bố đều đối với XOR vào và XOR ra tương ứng, thì các đặc trưng trên sẽ không có tác dụng làm giảm phép tìm kiếm khoá so với phương pháp vét kiệt. Nhưng do các nhược điểm cụ thể của từng S- hộp, nên người ta có thể chỉ ra được các đặc trưng mà khi sử dụng chúng sẽ làm giảm đáng kể công việc tìm khoá đúng của DES.

Sau đây là một số ví dụ về các đặc trưng n-vòng

$L'_0 = 00000000_{16}$ $R'_0 = 60000000_{16}$
$L'_1 = 60000000_{16}$ $R'_1 = 00808200_{16}$ $p = 14/64$

Một đặc trưng 1-vòng

$L'_0 = 40080000_{16}$ $R'_0 = 04000000_{16}$
$L'_1 = 04000000_{16}$ $R'_1 = 00000000_{16}$ $p = 1/4$
$L'_2 = 00000000_{16}$ $R'_2 = 04000000_{16}$ $p = 1$
$L'_3 = 04000000_{16}$ $R'_3 = 40080000_{16}$ $p = 1/4$

Một đặc trưng 3-vòng

Sử dụng các đặc trưng cụ thể, Biham và Shamir đã phân tích tấn công DES với các số liệu như sau

Số vòng	Số bản rõ chọn	Độ phức tạp
8	2^{14}	2^{16}
10	2^{24}	2^{35}
12	2^{31}	2^{43}
14	2^{39}	2^{51}
16	2^{47}	2^{58}

II. THÁM MÃ TUYẾN TÍNH ĐỐI VỚI HỆ DES

II.1. Nguyên lý chung của phương pháp thám mã tuyến tính đối với hệ DES

Như ta đã biết ở trên, hệ DES đã công khai toàn bộ các phép biến đổi trong nó, trong đó chỉ có các hộp nén mới là các phép biến đổi phi tuyến. Cái bí mật còn lại duy nhất khi sử dụng DES đó là khoá K được sử dụng cụ thể. Nếu tất cả các phép biến đổi của DES đều là tuyến tính, thì với ẩn số là khoá K cho trước cố định, bằng công cụ mô phỏng trên máy tính và sử dụng các cặp bản rõ-mã tương ứng ta có thể thiết lập được hệ thống phương trình tuyến tính để tìm lại được các bít khoá K đó trong thời gian đa thức. Tuy nhiên, các hộp nén (thành phần quan trọng nhất của hệ DES) là các phép biến đổi phi tuyến được chọn lựa cẩn thận, nên muốn thám DES thì phải tấn công vào chính thành trì này. Mục đích của phương pháp thám mã tuyến tính trên DES là tìm một biểu diễn xấp xỉ tuyến tính cho hệ này để có thể phá chúng nhanh hơn phương pháp tấn công vét kiệt. Và tất nhiên, những nhược điểm của các hộp nén sẽ lại được tiếp tục khai thác cho mục đích này.

Trước hết ta qui ước lại các yếu tố có liên quan đến sơ đồ thuật toán mã DES. Đầu tiên, cũng giống như trong phần thám mã vi sai đối với DES, ta có thể bỏ qua các hoán vị đầu IP và hoán vị cuối IP^{-1}. Và trong suốt phần này, ta qui ước bít tận cùng bên phải luôn được xem là bit thứ không (O^{th}).

Các ký hiệu được sử dụng trong mô hình là:

P : là bản rõ 64-bit;

C : là bản mã 64-bit tương ứng;

P_H : 32-bit bên trái của P;

P_L : 32-bit bên phải của P;

C_H : 32-bit bên trái của C;

C_L : 32-bit bên trái của C;

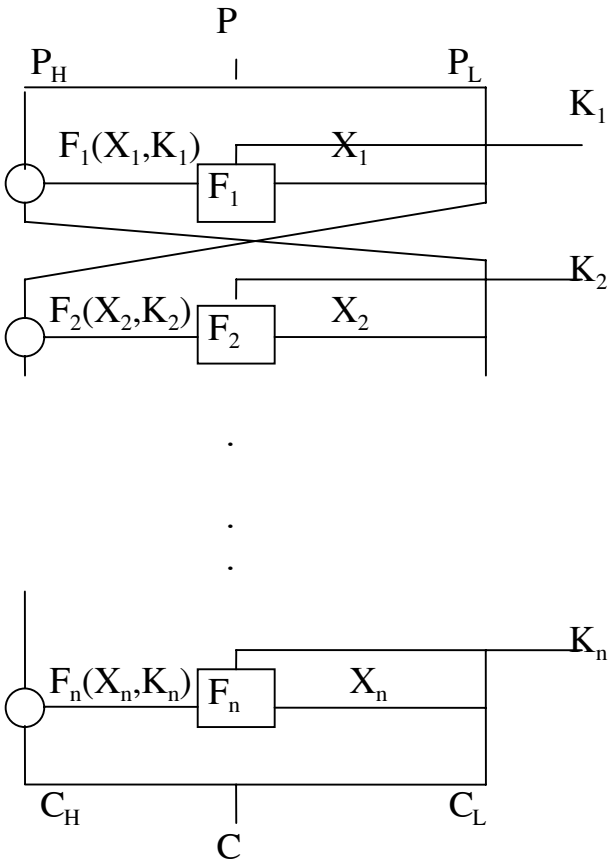
X_i : giá trị 32-bit trung gian tại vòng thứ i;

K_i : khoá con 48-bit được sử dụng tại vòng thứ i;

$F_i(X_i, k_i)$: hàm vòng thứ i;

A[i] : bít thứ i của A;

A[i,j,..,k] : là tổng A[i]⊕ A[j]⊕ .. ⊕ A[k].



Hình 2.3: Mô hình hệ DES với qui ước mới.

Mục đích của phương pháp thám mã tuyến tính đối với hệ DES là tìm các biểu diễn tuyến tính hiệu quả có dạng sau:

$$P[i_1, i_2, .. , i_a] \oplus C[j_1, j_2, .. , j_b] = K[k_1, k_2, .. , k_c], \quad (2.1)$$

trong đó $i_1, i_2, .. , i_a, j_1, j_2, .. , j_b$ và $k_1, k_2, .. , k_c$ là ký hiệu các vị trí bít cố định, và phương trình (2.1) đúng với xác suất $p \neq 1/2$ với giả thiết bản rõ P

được lấy ngẫu nhiên còn C là bản mã tương ứng với khoá K cố định cho trước nào đó. Giá trị tuyệt đối $|p - 1/2|$ được xem như độ hiệu quả của phương trình (2.1).

Nếu ta có thể thành công trong việc tìm một biểu diễn tuyến tính hiệu quả, thì khi đó có thể sử dụng nó để tìm ra được bít dạng khoá quan trọng $K[k_1, k_2,..,k_c]$ theo thuật toán sau dựa trên phương pháp hợp lý cực đại.

Thuật toán 1

Bước 1: Gọi T là số các bản rõ sao cho vế trái của phương trình (2.1) là bằng không. Ký hiệu N là số các bản rõ được sử dụng trong tấn công.

Bước 2: Nếu $T > N/2$ thì

lấy $K[k_1, k_2, .. , k_c] = 0$ (khi $p > 1/2$) hoặc bằng 1 (khi $p < 1/2$),

ngược lại

lấy $K[k_1, k_2, .. , k_c] = 1$ (khi $p > 1/2$) hoặc bằng 0 (khi $p < 1/2$).

Rõ ràng là khả năng thành công của thuật toán sẽ tăng khi N tăng hoặc biên độ $|p - 1/2|$ tăng lên. Chúng ta gọi biểu diễn tuyến tính có biên độ $|p - 1/2|$ cực đại là biểu diễn tốt nhất.

Thuật toán 1 trên đây về nguyên tắc có thể áp vào bất kỳ hệ mã khối nào. Tuy nhiên khi sử dụng nó để tấn công hệ DES, chúng ta có thể thực hiện như sau. Để tấn công DES n-vòng, chúng ta sử dụng các biểu diễn tuyến tính tốt nhất đối với DES (n-1)-vòng. Bản mã sau vòng thứ (n-1) có thể được thiết lập từ bản mã tại vòng thứ n, bằng cách thực hiện phép giải mã với khoá con K_n của vòng thứ n. Như vậy, ta chấp nhận một số hạng có hàm vòng F trong biểu diễn tuyến tính đang sử dụng để tấn công DES n-vòng. Kết quả là ta sẽ nhận được một dạng biểu diễn sau đúng với xác suất tốt nhất của (n-1)-vòng của DES:

$$P[i_1, i_2, .. , i_a] \oplus C[j_1, j_2, .. , j_b] \oplus F_n(C_L, K_n) = K[k_1, k_2, .. , k_c], \qquad (2.2)$$

Thực chất biểu thức $C[j_1, j_2, .. , j_b] \oplus F_n(C_L, K_n)[l_1, l_2, .. , l_d]$ chính là bản mã tại vòng thứ (n-1).

Như vậy, trong phương trình (2.2) ta thấy số lượng các bít khoá tham gia nhiều hơn so với phương trình (2.1). Trong đó việc tìm một số bít khoá trong K_n chính xác sẽ có thể được thực hiện dễ dàng hơn nhờ nhận xét sau. Nếu khoá K_n trong (2.2) là không chính xác thì độ hiệu quả của phương trình (2.1) sẽ giảm đi rõ rệt, tức là nó sẽ làm ảnh hưởng lớn đến xác suất đúng của (2.1). Do vậy, có thể sử dụng thuật toán hợp lý cực đại sau để cùng một lúc tìm và quyết định các thành phần khoá tham gia trong (2.2)

Thuật toán 2

Bước 1: Với mỗi một ứng cử viên $K_n^{(i)}$ (i =1,2,..) của K_n, gọi T_i là số các bản rõ sao cho vế trái của phương trình (2.2) bằng không. Ký hiệu N là số các bản rõ được sử dụng trong tấn công.

Bước 2: Giả sử T_{max} là giá trị cực đại và T_{min} là giá trị cực tiểu của tất cả các giá trị của T_i.

+ Nếu $|T_{max} - N/2| > |T_{min} - N/2|$, thì ta chấp nhận khoá ứng cử viên tương ứng với T_{max} và lấy $K[k_1, k_2, .. , k_c] = 0$ (khi $p > 1/2$) hoặc bằng 1 (khi $p < 1/2$),

+ Nếu $|T_{max} - N/2| < |T_{min} - N/2|$ thì ta chấp nhận khoá ứng cử viên tương ứng với T_{min} và lấy $K[k_1, k_2, .. , k_c] = 1$ (khi $p > 1/2$) hoặc bằng 0 (khi $p < 1/2$).

II.2. Xấp xỉ tuyến tính các hộp nén

Trong phần này, chúng ta sẽ nêu phương pháp xấp xỉ tuyến tính các hộp nén của DES làm cơ sở cho việc xấp xỉ tuyến tính cho cả hệ DES được sử dụng trong tấn công sau này.

Định nghĩa 2.8:

Cho trước hộp nén S_a (a = 1, 2, .., 8), $1 \le \alpha \le 63$ và $1 \le \beta \le 15$. Chúng ta định nghĩa số $NS_a(\alpha, \beta)$ là số tất cả các đầu ra của 64 mẫu đầu vào của S_a sao cho giá tri XOR của các bit đầu vào được đánh dấu bởi α trùng với giá trị XOR của các bít đầu ra được đánh dấu bởi β. Cụ thể hơn, ta có:

$NS_a(\alpha,\beta) = \#\{x: 0 \le x < 64, (\oplus^5_{s=0} (x[s] \bullet \alpha[s])) =$

$$(\oplus^{3}_{t=0} (S_a(x)[t] \bullet \beta[t]))\} \quad (2.3)$$

ở đây ký hiệu • là phép nhân lôgic AND giữa từng bít tương ứng của hai véc tơ.

Chú ý: Từ định nghĩa trên ta có thể thấy rằng, khi $NS_a(\alpha, \beta)$ không bằng 32, thì sẽ có một sự tương quan nào đó giữa đầu vào và đầu ra của hộp nén S_a. Chẳng hạn, từ việc khảo sát trực tiếp hộp S_5 ta có

$$NS_5(16,15) = 12 \quad (2.4)$$

Điều đó có nghĩa là bít vào thứ tư của S_5 là trùng với giá trị XOR của tất cả các bít đầu ra của S_5 với xác suất 12/64 = 0,19. Nói cách khác, chúng ta sẽ có xác suất để có biểu thức tuyến tính

$$(\alpha, X) \oplus (\beta, Y) = 0, \text{ hoặc}$$

$$(\alpha, X) \oplus (\beta, Y) = 1$$

sẽ tương ứng bằng $(NS_a / 64)$ hoặc bằng $[1- (NS_a / 64)]$, trong đó Y là ảnh của X qua phép biến đổi S_a.

Như vậy, nếu giá trị $NS_a(\alpha, \beta)$ càng xa giá trị 1/2 thì khả năng nhận được các tương quan tuyến tính thực sự giữa đầu vào và đầu ra qua hộp nén S_a càng có ý nghĩa hơn đối với thám mã.

Bây giờ, căn cứ vào (2.4) và tính đến các phép mở rộng E và hoán vị P trong hàm vòng F ta sẽ nhận được phương trình tuyến tính sau đúng với xác suất 0,19 khi khoá K cố định và X ngẫu nhiên:

$$X[15] \oplus F(X,K)[7, 18, 24, 29] = K[22] \quad (2.5)$$

Trong (2.5), ta có thể thấy giá trị $X[15] \oplus K[22]$ chính là bít đầu vào thứ tư của S_5, còn F(X,K)[7, 18, 24, 29] chính là giá trị XOR của 4 -bit đầu ra cũng của hộp S_5 đó.

Bằng việc khảo sát toàn bộ 8 hộp nén của DES, các tác giả trong [26] đã chỉ ra rằng phương trình (2.4) là một xấy xỉ tuyến tính hiệu quả nhất đối với tất cả các S-hộp, và do đó phương trình (2.5) sẽ là xấp xỉ tốt nhất của hàm vòng.

Trong [26], các tác giả cũng đã liệt kê 5 xấp xỉ tuyến tính tốt nhất đối với các hộp nén của DES như sau:

A: X[15] ⊕ F(X,K)[7, 18, 24, 29] = K[22], p = 12/64;

B: X[27, 28, 30, 31] ⊕ F(X,K)[15] = K[42, 43, 45, 46], p = 22/64;

C: X[29] ⊕ F(X,K)[15] = K[44], p = 30/64;

D: X[15] ⊕ F(X,K)[7, 18, 24] = K[22], p = 12/64;

E: X[12, 16] ⊕ F(X,K)[7, 18, 24] = K[19, 23], p = 15/64.

Các xấp xỉ tuyến tính trên đây sẽ được sử dụng để tấn công DES trong từng trường hợp cụ thể, ở đây chúng tôi chỉ nêu một số ví dụn minh hoạ cho kiểu tân công này.

II.3. Xấp xỉ tuyến tính hệ mã DES

Trong phần này ta sẽ mở rộng xấp xỉ tuyến tính của hàm vòng thành xấp xỉ tuyến tính cho chính hệ mã DES.

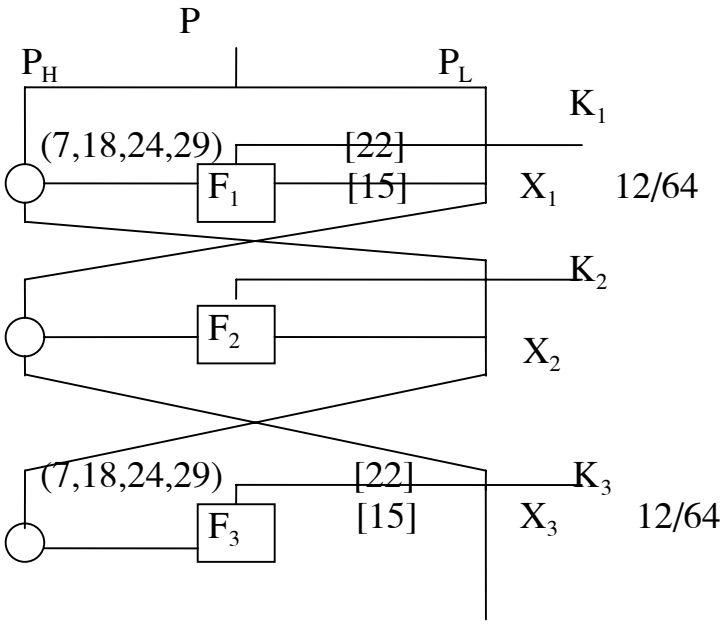
+ Đối với DES 3-vòng

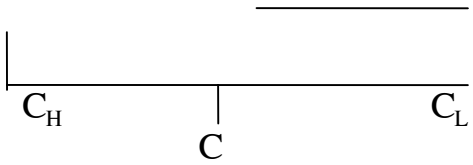
Trước hết, bằng cách áp phương trình (2.5) vào vòng đầu tiên, chúng ta sẽ nhận được phương trình sau đây đúng với xác suất 12/64:

$$X_2[7,18,24,29] \oplus P_H[7,18,24,29] \oplus P_L[15] = K_1[22] \qquad (2.6)$$

Ta cũng nhận được kết quả như vậy khi thực hiện phép giải mã đối với bản mã C thông qua khoá K_3, tức là ta có phương trình sau đây đúng với xác suất 12/64:

$$X_2[7,18,24,29] \oplus C_H[7,18,24,29] \oplus C_L[15] = K_3[22] \qquad (2.7)$$





Hình 2.4: Sơ đồ xấp xỉ tuyến tính hệ mã DES 3-vòng.

Kết hợp (2.6), (2.7) ta sẽ nhận được một biểu diễn xấp xỉ tuyến tính cho hệ mã DES 3-vòng như sau:

$$P_H[7,18,24,29] \oplus C_H[7,18,24,29] \oplus P_L[15] \oplus C_L[15] = K_1[22] \oplus K_3[22] \qquad (2.8)$$

Xác suất để có phương trình (2.8) có thể được tính như sau, với giả thiết P là bản rõ ngẫu nhiên còn C là bản mã tương ứng với một khoá K cố định.

Trước hết ta có thấy (2.6) có thể được phát biểu tương đương với

$$X_2[7,18,24,29] \oplus P_H[7,18,24,29] \oplus P_L[15] \oplus K_1[22] = 0,$$

với xác suất 12/64 và

$$X_2[7,18,24,29] \oplus P_H[7,18,24,29] \oplus P_L[15] \oplus K_1[22] = 1,$$

với xác suất (1- 12/64).

Tương tự với (2.7), ta có

$$X_2[7,18,24,29] \oplus C_H[7,18,24,29] \oplus C_L[15] \oplus K_3[22] = 0,$$

với xác suất 12/64 và

$$X_2[7,18,24,29] \oplus C_H[7,18,24,29] \oplus C_L[15] \oplus K_3[22] = 1,$$

với xác suất (1- 12/64).

Vậy xác suất để có (2.8) có dạng xác suất để có tập hợp dạng $(0 \oplus 0) \cup (1 \oplus 1)$. Do đó, xác suất này đúng bằng

$$(12/64) \times (12/64) + (1-12/64) \times (1-12/64) = 0,70. \qquad (2.9)$$

Từ chỗ phương trình (2.5) là xấy xỉ tuyến tính tốt nhất của hàm vòng F, nên (2.8) là biểu diễn tuyến tính tốt nhất đối với DES 3-vòng.

Áp dụng thuật toán 1 vào phương trình (2.8) ta có thể tìm được bít khoá $K_1[22] \oplus K_3[22]$.

+ Đối với DES 5-vòng

Đối với hệ mã DES 5-vòng, trước hết ta áp phương trình (2.5) vào các vòng thứ hai và thứ tư, còn các vòng thứ nhất và thứ năm sẽ sử dụng biểu diễn tuyến tính sau

B: $X[27, 28, 30, 31] \oplus F(X,K)[15] = K[42, 43, 45, 46]$, $p = 22/64$ (2.10)

xuất phát từ hệ thức $NS_1(27,4) = 22$.

Cách thức áp dụng các biểu thức tuyến tính A, hay B trên đây cũng giống như trong trường hợp DES 3-vòng. Tức là đối với các vòng thứ nhất và thứ hai, ta coi X là bản rõ P, còn đối với các vòng thứ năm và thứ tư, ta coi X là bản mã C và thực hiện phép giải mã ngược trở lại tới vòng thứ 3 nhờ sử dụng các khoá K_5 và K_4. Khi đó phối hợp các kết quả lại ta sẽ thu được biểu diễn xấp xỉ tuyến tính tốt nhất cho hệ DES 5-vòng như sau:

$P_H[15] \oplus P_L[7,18,24,27,28,29,30,31] \oplus C_H[15] \oplus$
$C_L[7,18,24,27,28,29,30,31]$

$= K_1[42,43,45,46] \oplus K_2[22] \oplus K_4[22] \oplus K_5[42,43,45,46]$ (2.11)

Để tính xác suất cho phương trình (2.11) ta sẽ sử dụng bổ đề sau đây.

Bổ đề 2.9.

Giả sử X_i $(1 \le i \le n)$ là các biến ngẫu nhiên độc lập, nhận giá trị 0 với xác suất p hoặc nhận giá trị 1 với xác suất (1-p). Khi đó xác suất để cho

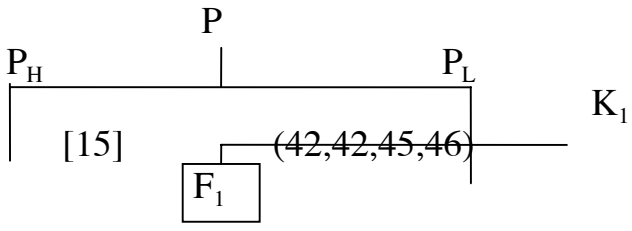
$X_1 \oplus X_2 \oplus .. \oplus X_n = 0$ sẽ bằng

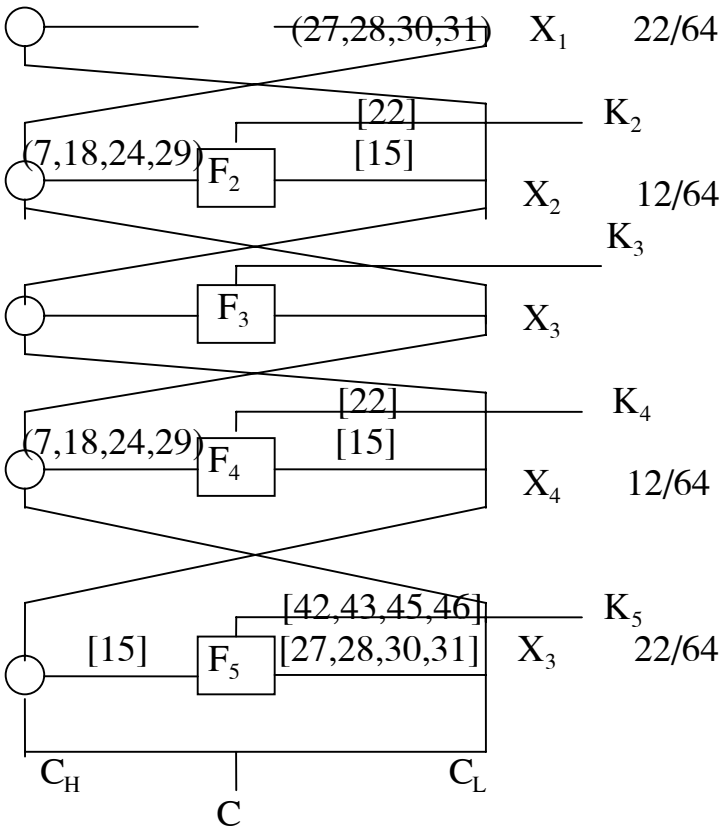
$$1/2 + 2^{n-1} \prod_{i=1}^{n} (p_i - 1/2). \qquad (2.12)$$

Bổ đề trên có thể được chứng minh bằng qui nạp, và người ta gọi là bổ đề ***Piling-up***.

Áp dụng bổ đề này ta thấy phương trình (2.11) đúng với xác suất

$1/2 + 2^3 (-10/64)^2 (-20/64)^2 = 0,519$





Hình 2.5: Sơ đồ xấp xỉ tuyến tính hệ mã DES 5-vòng.

Bằng cách thức đã nêu, trong [26] các tác giả đã liệt kê ra các biểu diễn xấp xỉ tuyến tính tốt nhất đối với hệ DES tới 20 vòng. Ví dụ, biểu diễn tuyến tính tốt nhất cho DES 8-vòng có dạng sau:

$$P_H[7,18,24] \oplus P_L[12,16] \oplus C_H[15] \oplus C_L[7,18,24,29,27,28,30,31]$$

$$= K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[42,43,45,46]$$

đúng với xác suất $1/2 - 1{,}22 \times 2^{-11}$. (2.13)

Biểu diễn tuyến tính tốt nhất cho DES 16-vòng có dạng :

$$P_H[7,18,24] \oplus P_L[12,16] \oplus C_H[15] \oplus C_L[7,18,24,29,27,28,30,31]$$

$$= K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[44] \oplus K_9[22] \oplus K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \oplus K_{15}[22] \oplus K_{16}[42,43,45,46];$$

đúng với xác suất $1/2 - 1{,}49 \times 2^{-24}$. (2.14)

II.4 Tấn công bản rõ đã biết đối với hệ mã DES

Sử dụng các biểu diễn xấp xỉ tuyến tính tốt nhất đã nghiên cứu trong phần trên, ta sẽ trình bày phương pháp tấn công bản rõ đã biết đối với hệ DES.

** VỚI DES 8-VÒNG*

Sử dụng thuật toán 1 áp vào phương trình (2.13) ta sẽ thu được hệ thống phương trình để tìm ra tổng của 10 bít khoá: $K_1[19]$, $K_1[23]$, $K_3[22]$, $K_4[44]$, $K_5[22]$, $K_7[22]$, $K_8[42]$, $K_8[43]$, $K_8[45]$, $K_8[46]$. Và theo [10], để làm được điều đó ta phải sử dụng cỡ $1,22^{-2}.\ 2^{22} \approx 2^{21}$ bản rõ đã biết mới có thể thiết lập được hệ thống phương trình cần thiết. Tuy nhiên để tăng hiệu quả tấn công tuyến tính ta sẽ thiết lập các điều kiện để sử dụng thuật toán 2.

Sử dụng thuật toán 2: Để tấn công DES 8-vòng, ta sẽ sử dụng biểu diễn tuyến tính tốt nhất đối với DES 7-vòng, còn vòng thứ 8 dùng để giải mã bản mã C dùng khoá K_8. Khi đó ta sẽ thu được biểu diễn tuyến tính tốt nhất đối với DES 8- vòng có dạng sau:

$$P_H[7,18,24] \oplus P_L[12,16] \oplus C_H[15] \oplus C_L[7,18,24,29] \oplus F_8(C_L, K_8)[15]$$

$$= K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22]$$

đúng với xác suất $1/2 + 1{,}95 \times 2^{-10}$. (2.15)

Phương trình này chứa khoá con 48-bit K_8, tuy nhiên chỉ có 6-bit khoá thực sự tác động tới hàm vòng $F_8(C_L, K_8)[15]$ đó là các bít khoá $K_8[42]$, $K_8[43]$, $K_8[44]$, $K_8[45]$, $K_8[46]$, $K_8[47]$. Do đó ta cần 64 bộ đếm để thực hiện theo thuật toán 2, nhằm mục đímh tìm ra 6-bít khoá con của K_8 đã được sử dụng thực sự trong mã dịch. Tóm lại, nếu áp dụng thuật toán 2 vào phương trình (2.15) ta sẽ thu được hệ thống phương trình tuyến tính để tìm ra được 06- bít khoá là: $K_8[42]$, $K_8[43]$, $K_8[44]$, $K_8[45]$, $K_8[46]$, $K_8[47]$; và tổng của các bít khoá: $K_1[19]$, $K_1[23]$, $K_3[22]$, $K_4[44]$, $K_5[22]$, $K_7[22]$, bằng việc sử dụng khoảng 2^{20} bản rõ đã biết.

** VỚI DES 16-VÒNG:* Tương tự như phương pháp tấn công DES 8-vòng, để tấn công DES 16 -vòng ta sẽ sử dụng biểu diễn tuyến tính tốt nhất đối với DES 15-vòng, còn vòng thứ 16 dùng để giải mã bản mã C dùng khoá K_{16}. Khi đó ta sẽ thu được biểu diễn tuyến tính tốt nhất đối với DES 16 - vòng có dạng sau:

$$P_H[7,18,24] \oplus P_L[12,16] \oplus C_H[15] \oplus C_L[7,18,24,29] \oplus F_{16}(C_L, K_{16})[15]$$
$$= K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[44] \oplus K_9[22] \oplus$$
$$K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \oplus K_{15}[22];$$

đúng với xác suất $1/2 + 1{,}19 \times 2^{-22}$. (2.16)

Sử dụng thuật toán 2 ta sẽ thu được hệ thống phương trình tuyến tính để tìm ra 06-bit khoá sau: $K_{16}[42]$, $K_{16}[43]$, $K_{16}[44]$, $K_{16}[45]$, $K_{16}[46]$, $K_{16}[47]$; cùng với bít tổng của: $K_1[19]$, $K_1[23]$, $K_3[22]$, $K_4[44]$, $K_5[22]$, $K_7[22]$, $K_8[44]$, $K_9[22]$, $K_{11}[22]$, $K_{12}[44]$, $K_{13}[22]$, $K_{15}[22]$, bằng cách sử dụng khoảng $8.(1{,}19.\ 2^{-22})^{-2} \approx 2^{47}$ bản rõ đã biết. Tương tự có thể tìm được 06-bít khoá tại vòng đầu tiên và bít khoá tổng tương ứng. Như vậy, với DES 16-vòng ta có thể dùng tấn công tuyến tính để tìm được 14-bít khoá trong số 56-bít. Các bít khoá còn lại sẽ được thám bằng phương pháp vét kiệt, và rõ ràng tổng độ phức tạp tính toán sẽ nhỏ hơn phương pháp vét kiệt toàn bộ không gian khoá 2^{56}.

III. THÁM MÃ PHI TUYẾN

Như chúng ta đã biết, không có quan hệ tuyến tính nào giữa đầu ra và đầu vào của các S-hộp của DES. Mặt khác bằng cách biểu diễn các S-hộp như là các đa thức Boolean thì dễ dàng có thể thiết lập các quan hệ đại số nào đó giữa các bít đầu ra và đầu vào của các S-hộp. Chúng ta cũng biết rằng bậc của các đa thức này là nhỏ hơn hay bằng 6. Do đó một cách tự nhiên bài toán sau đây có thể được đặt ra: Bậc nhỏ nhất của các quan hệ đại số của các S-hộp là bao nhiêu, quan hệ đại số nào có bậc nhỏ nhất? Có thể thấy rằng luôn tồn tại một quan hệ bậc 3 trong tất cả các S-hộp. Bởi vậy câu hỏi trên được viết lại như sau: liệu có tồn tại quan hệ bậc hai hay không? Trong bài báo [34] các tác giả cho thấy có 7 quan hệ bậc hai của các S-hộp S_1, S_4 và S_5 của DES với xác suất 1. Bằng cách sử dụng một trong các quan hệ bậc hai này, họ đã xây dựng một thuật toán tấn công tuyến tính cải tiến cho DES đủ 16-vòng. Cách thức thực hiện là tổ hợp phương pháp xấp xỉ phi tuyến và phương pháp xấp xỉ nhiều lần. Sự cải tiến này có thể rút gọn số cặp Rõ-Mã đòi hỏi xuống còn 25/34 (73,5%) của con số 2^{43} đòi hỏi bởi tấn công của Matsui.

III.1 Thiết lập các quan hệ bậc hai của S-hộp

Về nguyên tắc thông qua bảng giá trị của các S-hộp chúng ta có thể thiết lập được tất cả các quan hệ đại số Boolean giữa đầu vào và đầu ra của chúng. Chẳng hạn, với hộp S_1, giá trị đầu ra là 13(=(1,1,0,1)) tương ứng với giá trị đầu vào là 4(=(0,0,0,1,0,0)) sẽ cho một quan hệ đại số giữa các biến Boolean vào $x_1, x_2,\ldots, x_6$ và các biến Boolean ra $y_1, y_2,\ldots, y_4$ có dạng:

$$(x_1+1)\,(x_2+1)\,(x_3+1)\,(x_4+0)\,(x_5+1)\,(x_6+1)$$
$$((y_1+0)\,(y_2+0)\,(y_3+1)\,(y_4+0)+1) = 0.$$

Như vậy mỗi S-hộp cho ta 64 quan hệ đại số. Dùng kỹ thuật hệ cơ sở Gnobner, các tác giả trong [34] đã thu được các kết quả sau.

Bổ đề 2.10.

+Không tồn tại quan hệ tuyến tính của các S-hộp

+Tồn tại quan hệ bậc 3 cho mỗi S-hộp

+Có 7 quan hệ bậc hai đối với các hộp S_1, S_4 và S_5.

Chú ý rằng các quan hệ trên đúng với xác suất 1. Ta chú ý quan hệ bậc hai sau của hộp S_5: $(x_1, x_2,\ldots, x_6) \rightarrow (y_1, y_2,\ldots, y_4)$.

$$x_1y_1 + x_1y_2 + x_1y_3 + x_1y_4 + x_2y_1 + x_2y_2 + x_2y_3 +$$
$$x_2y_4 + x_2x_1 + x_5y_1 + x_5y_2 + x_5y_3 + x_5y_4 + x_5x_2 +$$
$$+y_1 + y_2 + y_3 + x_1 + x_2 + x_5 + 1 = 0. \qquad (2.17)$$

Có thể phân tích vế trái của (4) để được quan hệ sau:

$$(y_1 + y_2 + y_3 + y_4 + x_2 + 1).(x_1 + x_2 + x_5 + 1) = 0. \qquad (2.18)$$

Ở đây ta thấy thừa số thứ nhất của vế trái của (2.18) chính là xấp xỉ tuyến tính tốt nhất của hộp S_5 với độ lệch 5/16 đã được tìm ra bởi Matsui trong tấn công tuyến tính là:

$$y_1 + y_2 + y_3 + y_4 + x_2 = 0. \qquad (2.19)$$

Phần sau chúng tôi giới thiệu cách thức vận dụng quan hệ phi tuyến (2.18) để cải tiến tấn công tuyến tính với DES đủ 16-vòng.

III.2. Áp dụng vào thám mã phi tuyến

Với các ký hiệu đã qui ước trong phần thám mã tuyến tính của Matsui, từ quan hệ đại số (2.18) ta có thể mở rộng thành quan hệ đại số của hàm vòng thứ i, F_i: $(X_i, K_i) \rightarrow F_i(X_i, K_i)$ như sau:

(A*) $(F_i[3, 8, 15, 24] + X_i[17] + K_i[26] + 1).$

$$.(X_i[16, 17, 20] + K_i[25, 26, 29] + 1) = 0, \qquad (2.20)$$

ở đây $X_i \in GF(2)^{32}$ là đầu vào của vòng thứ –i và $K_i \in GF(2)^{48}$ là khoá vòng thứ –i của hàm vòng F_i.

Như trên đã nói, trong quan hệ bậc hai (2.20), ta có thể tìm thấy xấp xỉ tuyến tính tốt nhất (A) cho hàm vòng thứ-i là F_i với độ lệch tuyệt đối là 5/16 trong tấn công của Matsui:

(A) $F_i[3, 8, 15, 24] + X_i[17] + K_i[26] = 0$ (2.21)

Matsui cũng đã thiết kế xấp xỉ tuyến tính sau đây đối với DES 16-vòng bằng cách sử dụng xấp xỉ tuyến tính tốt nhất của 14-vòng có dạng A-ACD-DCA-ACD- với độ lệch tuyệt đối là 1,19. 2^{-21} (chúng là nối ghép ba xấp xỉ tuyến tính A, C, D của hàm vòng):

$$P_r[3, 8, 14, 25] + P_l[17] + C_l[8, 14, 25] + F_1(P_r, K_1)[17] + F_{16}(C_r, K_{16})[8, 14, 25] = K_2[26] + K_4[26] + K_5[4] + K_6[26] + K_8[26] + K_9[4] + K_{10}[26] + K_{12}[26] + K_{13}[4] + K_{14}[26]. \quad (2.22)$$

ở đây P_l, P_r là các các nửa trái phải của bản rõ, còn C_l, C_r là các các nửa trái phải của bản mã.

Từ chỗ A* là xấp xỉ phi tuyến với độ lệch (1/2) chúng ta có thể đạt được xấp xỉ phi tuyến sau của DES 16-vòng có dạng A*-ACD-DCA-ACD- bằng cách thay thế xấp xỉ tuyến tính A bằng quan hệ bậc hai A* có độ lệch cao hơn:

$$(P_r[3, 8, 14, 25] + P_l[17] + C_l[8, 14, 25] + F_1(P_r, K_1)[17] + F_{16}(C_r, K_{16})[8, 14, 25] + K_2[26] + K_4[26] + K_5[4] + K_6[26] + K_8[26] + K_9[4] + K_{10}[26] + K_{12}[26] + K_{13}[4] + K_{14}[26] + 1).(P_l[16, 17, 20] + F_1(P_r, K_1)[16, 17, 20] + K_2[25, 26, 29] + 1) = 0. \quad (2.23)$$

Độ lệch tuyệt đối của xấp xỉ (2.23) cao hơn của (2.22). Tuy nhiên chúng ta không thể sử dụng trực tiếp (2.23) để rút gọn số bản Rõ-Mã trong tấn công tìm các bít khoá hiệu quả của DES đủ 16-vòng có mặt trong (2.23). Bởi vì số các bít text hiệu quả và các bít khoá hiệu quả trong (2.23) lớn hơn nhiều so với chúng ở trong (2.22). Trong phần sau chúng ta sẽ trình bày (2.23) theo cách áp dụng phép xấp xỉ nhiều lần để tránh vấn đề trên.

III.3. Sử dụng xấp xỉ tuyến tính nhiều lần

Ý tưởng cơ bản của thám mã tuyến tính là tìm các xấp xỉ tuyến tính nào đó tác động trên mã khối gắn trong một biểu thức với các bít bản rõ $P_{i_1},..,P_{i_a}$, bản mã $K[\chi_K]$ và khoá $K_{k_1},..,K_{k_c}$. Chúng ta sẽ viết $P_{i_1} \oplus P_{i_2} \oplus .. \oplus P_{i_\alpha}$ dưới dạng gọn là $P[\chi_P]$ và có thể viết một xấp xỉ tuyến tính dưới dạng

$$P[\chi_P] \oplus C[\chi_C] = K[\chi_K] \quad (2.24)$$

Nếu phương trình (11) đúng với xác suất p = $\frac{1}{2}+\varepsilon$, với phép chọn bản rõ ngẫu nhiên và khoá cố định thì ta nói chúng có độ lệch là ε. Để tấn công DES, ta thường áp dụng phương pháp 2-R với phương trình có dạng sau đây

$$P[\chi_P] \oplus C[\chi_C] \oplus F_1(P_L, K_1)[\chi_{F1}] \oplus F_r(C_L, K_r)[\chi_{Fr}] = K[\chi_K] \qquad (2.25)$$

ở đây các bít khoá xuất hiện trong phương trình trên là nhiều hơn so với dạng (2.24). Mỗi phương trình dạng (2.25) sẽ cho ta một thuật toán tấn công tuyến tính kiểu 2-R đối với DES. Phương trình nào có độ lệch lớn nhất sẽ cho ta thuật toán tấn công hiệu quả nhất.

Tuy nhiên, ta cũng biết rằng có nhiều xấp xỉ tuyến tính khác nhau đối với một hệ mã khối trên một số vòng cho trước. Trong trường hợp các xấp xỉ này lại cùng bao hàm các bít khoá hiệu quả, liệu độ phức tạp tấn công có thể làm giảm đi được hay không. Vấn đề này đã được giải quyết bởi phương pháp tấn công sử dụng xấp xỉ nhiều lần [5].

Giả sử ta có *n* xấp xỉ tuyến tính bao hàm cùng các bít khoá nhưng khác nhau các bít bản rõ bản mã trong các xấp xỉ đó có dạng:

$$P[\chi_P^i] \oplus C[\chi_C^i] \oplus F_1(P_L, K_1)[\chi_{F_1}^i] \oplus F_r(C_L, K_r)[\chi_{F_r}^i] = K[\chi_K]. \qquad (2.26)$$

Giả thiết không mất tổng quát rằng tất cả các phương trình trên đều có độ *lệch* ε_i *là dương*. Khi đó ta có thuật toán sử dụng *n* xấp xỉ trên để quyết định các bít khoá cùng xuất hiện trong chúng như sau.

Thuật toán 2M.

Bước 1. Giả sử $K_1^{(g)}$ (g = 1, 2, …) và $K_r^{(h)}$ (h = 1, 2, …) là các khoá ứng cử viên của K_1 và K_r. Khi đó với mỗi cặp $(K_1^{(g)}, K_r^{(h)})$ và với mỗi xấp xỉ *i*, giả sử $T_{g,h}^i$ là số các bản rõ sao cho vế trái của phương trình (2.26) là bằng 0 khi K_1được thay thế bởi $K_1^{(g)}$ và K_r được thay thế bởi $K_r^{(h)}$. Giả sử N là số tất cả các bản rõ được dùng trong thuật toán.

Bước 2. Đặt $a_i = \varepsilon_i / \sum_{i=1}^n \varepsilon_i$. Tính với mỗi g, h các số

$$U_{g,h} = \sum_{i=1}^{n} a_i T_{g,h}^i$$

Bước 3. Giả sử U_{max} là giá trị cực đại và U_{min} là giá trị cực tiểu trong tất cả các giá trị $U_{g,h}$.

+nếu $|\ U_{max} - N/2\ | > |\ U_{min} - N/2\ |$, thì chấp nhận bộ khoá ứng cử viên tương ứng với U_{max} và cho $K[\chi_K] = 0$. (chú ý ở đây giả thiết ε_i dương)

+nếu $|\ U_{max} - N/2\ | < |\ U_{min} - N/2\ |$, thì chấp nhận bộ khoá ứng cử viên tương ứng với U_{min} và cho $K[\chi_K] = 1$. (chú ý ở đây giả thiết ε_i dương)

Theo S. Burton, Jr. Kaliski, M.J.B. Robshaw [5], thuật toán trên đây sẽ cho tỉ lệ thành công cao hơn so với thuật toán 2 nguyên thuỷ của Matsui tương ứng với cùng số bản rõ được chọn trong tấn công. Điều này được thể hiện qua định lý sau.

Giả thiết 1: Với mọi i và j, với i≠j, $x_i = x_j$ với xác suất 1/2, ở đây xác suất được lấy trên tất các các bản rõ lựa chọn ngẫu nhiên.

Giả thiết 2: Phân bố của thống kê U có thể được mô hình hoá một cách chính xác nhờ sử dụng một phân phối chuẩn.

Định lý 2.11. Dưới các Giả thiết 1 và 2, tỉ lệ thành công của thuật toán 2M với các trọng số a_i tối ưu là

$$\Phi\left(2\sqrt{N}\sqrt{\frac{\sum_{i=1}^{n}\varepsilon_i^2}{1-4.\sum_{i=1}^{n}\varepsilon_i^2}}\right).$$

Khi $\sum\varepsilon_i^2$ là nhỏ, chúng ta có thể xấp xỉ tỉ lệ thành công bởi $\Phi\left(2\sqrt{N}\sum\varepsilon_i^2\right)$, đó chính là tổng quát hoá của tỉ lệ thành công trong thuật toán nguyên thuỷ của Matsui $\Phi\left(2\sqrt{N}\varepsilon\right)$

III.4. Áp dụng tổ hợp xấp xỉ nhiều lần và xấp xỉ phi tuyến để tấn công DES

Trong phần trước chúng ta đã thiết lập xấp xỉ (2.23) của DES 16-vòng. Số các bit text và số các bít khoá hiệu quả của (2.23) tương ứng là 24 và 26. Do đó sẽ là không *hiệu quả* nếu thiết kế tìm 26 bít khoá hiệu quả ngay một lúc, vì cỡ bảng đếm cho các bít khoá này quá lớn. Để tránh vấn đề này, chúng ta giải quyết mỗi thừa số trong (2.23) một cách độc lập. Phương trình sau đây là thừa số thứ hai trong (2.23):

$$P_l[16, 17, 20] + F_1(P_r, K_1)[16, 17, 20] = K_2[25, 26, 29]. \quad (2.27)$$

Khi (2.27) đúng, độ lệch của (2.20) thay đổi thành $\varepsilon_0 = (1/2)(5/16)p_{14} = (8/5)p_{14}$. Khi (2.27) không đúng, nó thay đổi thành $\varepsilon_1 = 2.(1- (8/5)/2)p_{14} = (2/5)p_{14}$. Như vậy chúng ta sẽ làm việc với xấp xỉ tuyến tính (9) như là hai phép xấp xỉ, một là khi (2.27) đúng, và hai là khi (2.27) không đúng.

Giả sử N là số các cặp Rõ-Mã. T_0 (, T_1) là số các cặp Rõ-Mã sao cho vế trái của (9) là bằng 0 và phương trình (14) là đúng (, hoặc không đúng). Ta có thể tính toán thống kê $U = a_0T_0 + a_1T_1$ với các trọng số a_0, a_1 sao cho

$a_0 + a_1 = 2$. Để cực đại hoá khoảng cách giữa N/2 và giá trị trung bình E[U] theo hướng độ lệch chuẩn, chúng ta sử dụng Bổ đề sau.

Bổ đề 2.12. (Kaliski và Robshaw): *Khoảng cách* $|N/2 - E[U]|/\sigma_U$ *là cực đại với mỗi N cho trước khi các trọng số* a_i *tỉ lệ thuận với các độ lệch của các xấp xỉ đó.*

Từ Bổ đề 2.12, ta thấy rằng phép chọn tốt nhất cho các trọng số a_0, a_1 là

$$a_0 : a_1 = \varepsilon_0 : \varepsilon_1 = 4:1.$$

III.5 Thuật toán cải tiến để tấn công DES 16-vòng

Trong phần này, chúng tôi sẽ nêu thuật toán cải tiến để tấn công DES đủ 16-vòng. Nó vẫn còn đòi hỏi một khối lượng lớn các bản rõ mã hiệu quả và các khoá hiệu quả trong phương trình (2.22). Để tối thiểu hoá lượng công việc sử dụng trong xử lý dữ liệu, các tác giả [34] đã chia thuật toán ra làm hai phần. Phần đầu là thuật toán nguyên thuỷ của Matsui (các bước 1,2,3). Phần hai là phần cải tiến dùng để thay thế phép vét cạn tìm khoá trong tấn công của Matsui với các xấp xỉ nhiều lần (các bước 4,5,6).

Thuật toán:

1. Tính toán các cặp bản rõ và bản mã và cộng thêm vào các bit text hiệu quả của phương trình (2.22) và (2.27).
2. Cộng thêm (tăng) các bộ đếm trong tập K tương ứng với các bít khoá hiệu quả của (2.22) nếu vế trái của (2.22) bằng 0.
3. Xắp xếp, lựa chọn các khoá hiệu quả của (2.22) bằng cách sử dụng các bộ đếm K theo thứ tự thực tế có thể được.
4. Với các khoá hiệu quả có khả năng nhất của (2.22) khi vế phải của (2.27) bằng 0, hãy cộng thêm vào các bộ đếm trong tập H_0 *tương ứng với các bít khoá hiệu quả của (2.27) một lượng là 4 hay 1 tuỳ theo vế trái của (2.27) là bằng 0 hay 1 một cách tương ứng, hoặc là cộng thêm vào các bộ đếm trong tập* H_1 *một lượng là 1 hay 4 tương ứng như thế khi vế phải của (2.27) bằng 1.* H_0 *được sử dụng khi vế phải của (2.27) bằng 0, còn* H_1 *được sử dụng khi vế phải của (2.27) bằng 1)*
5. Xắp xếp thứ tự, lựa chọn các khoá hiệu quả của (2.27) bằng cách sử dụng các bộ đếm H_0 *và* H_1 *theo thứ tự độ tin cậy thực tế có thể được.*
6. Từ các khoá hiệu quả có khả năng nhất của (2.22) và (2.27), hãy tìm các bít khoá còn lại.

Trong bài của Matsui, các bít text và bít khoá hiệu quả của (2.22) đã được chỉ rõ. Cụ thể là có 13 bít text hiệu quả là:

$P_r[32], P_r[1],\ldots, P_r[5], P_r[16],\ldots., P_r[21], P_r[3, 8, 14, 25] + P_l[17] + C_l[3, 8, 14],$ (2.28)

Và 12 bít khoá hiệu quả là:

$$K_1[1],\ldots, K_1[6], K_1[25],\ldots, K_1[30]. \quad (2.29)$$

Có 17 bít text hiệu quả của (2.27) nếu các bít khoá trong (2.29) là cố định:

$$P_l[16, 17, 20], P_r[8], \ldots., P_r[17], \quad (2.30)$$

Và 13 bít khoá hiệu quả của (2.27) nếu các bít khoá trong (2.29) là cố định là:

$$K_1[13],\ldots, K_1[24], K_2[25, 26, 29]. \quad (2.31)$$

Ngoài ra chúng ta có thể sử dụng các phép xấp xỉ khác thay thế các bản rõ mã trong (2.22) và (2.27).

Trong thuật toán này, họ đã dùng bộ đếm cỡ 12-bít cho các bít khoá hiệu quả trong (2.22) và bộ đếm 13-bit cho các bít khoá hiệu quả trong (2.27). Như vậy ta đã rút gọn cỡ bộ đếm từ $2x2^{25}$ xuống còn $2x(2^{12} + 2^{13})$ nhờ thuật toán cải tiến này.

Với thuật toán cải tiến trên đây, các tác giả T. Shimoyama Shimoyama và T. Kaneko [34] *đã thực hiện tấn công DES-16 vòng chỉ sử dụng 25/34.2^{43} bản rõ đã biết để tìm đủ 56-bit khoá với tỉ lệ thành công cũng như thuật toán nguyên thuỷ của Matsui.*

III.6 Thực hành tấn công phi tuyến với DES tìm đủ 56-bít khoá

Ở đây chúng ta quy ước các từ (word) là các thanh ghi 32 bit và thứ tự các bit được đánh từ bên trái sang phải. P_r, P_l lần lượt là nửa bên phải và bên trái của khối bản rõ; C_r, C_l lần lượt là nửa bên phải và bên trái của khối bản mã; K_i là khoá ở vòng thứ i.

1. Áp dụng quan hệ bậc hai của các hộp S cho việc thám mã DES–8 vòng

Để phục vụ cho việc thám mã tuyến tính DES 8 vòng, chúng ta sử dụng xấp xỉ tuyến tính tốt nhất của DES-6 vòng, được ghép nối từ 3 xấp xỉ tuyến tính A, C, D của hàm vòng theo thứ tự A-ACD-. Xấp xỉ tuyến tính hiệu quả này có độ lệch tuyến tính là $p_6 = 1{,}95.2^{-9}$.

$$P_r[8,14,25] + C_h[3,8,14,25] + C_l[15]$$

$$= K_2[26] + K_3[4] + K_4[26] + K_6[26] \quad (2.32)$$

Từ xấp xỉ tuyến tính tốt nhất của DES – 6 vòng ở trên (phương trình (2.32)), áp vào hệ mã DES 8-vòng (từ vòng 2 tới vòng 7) ta thu được xấp xỉ tuyến tính cho hệ mã DES 8 vòng (2.33) và theo chiều ngược lại (từ vòng 7 tới vòng 2) ta thu được xấp xỉ tuyến tính (2.34).

$$P_r[3,8,14,25] + P_l[17] + C_l[8,14,25] + F_1(P_r,K_1)[17] + F_8(C_r,K_8)[8,14,25] = K_2[26] + K_4[26] + K_5[4] + K_6[26] \quad (2.33)$$

$$C_r[3,8,14,25] + C_l[17] + P_l[8,14,25] + F_1(P_r,K_1)[8,14,25] + F_8(C_r,K_8)[17] = K_3[26] + K_4[4] + K_5[26] + K_7[26] \quad (2.34)$$

Theo kết quả của T. Shimoyama Shimoyama và T. Kaneko, ta có xấp xỉ phi tuyến A* cho hàm vòng thứ i:

$$A^*: \ (F_i[3,8,14,25] + X_i[17] + K_i[26] + 1)(X_i[16,17,20] + K_i[25,26,29]+1) = 0 \quad (2.35)$$

Điểm đặc biệt ở đây là phần tử thứ nhất trong biểu thức phi tuyến A* trùng với xấp xỉ tuyến tính của hàm vòng A mà Matsui đã thu được (độ lệch 5/16). Cũng theo lập luận của T. Shimoyama Shimoyama và T. Kaneko; A* là biểu thức phi tuyến với độ lệch 1/2 và bằng việc thay xấp xỉ phi tuyến A* cho A trong biểu thức xấp xỉ tuyến tính của DES 8 vòng A*-ACD-, ta thu được biểu thức xấp xỉ phi tuyến sau:

$$(P_r[3,8,14,25] + P_l[17] + C_l[8,14,25] + F_1(P_r,K_1)[17] + F_8(C_r,K_8)[8,14,25] + K_2[26] + K_4[26] + K_5[4] + K_6[26] + 1) \ . \ (P_l[16,17,20] + F_1(P_r,K_1)[16,17,20] + K_2[25,26,29] + 1) = 0 \quad (2.36)$$

Hoàn toàn tương tự, ta thu được xấp xỉ phi tuyến (2.37):

$$(P_l[8,14,25] + C_r[3,8,14,25] + C_l[17] + F_1(P_r,K_1)[8,14,25] + F_8(C_r,K_8)[17] + K_3[26] + K_4[4] + K_5[26] + K_7[26] + 1) \ . \ (C_l[16,17,20] + F_8(C_r,K_8)[16,17,20] + K_7[25,26,29] + 1) = 0 \quad (2.37)$$

Ở đây ta chỉ tập trung xét xấp xỉ phi tuyến (2.36), bởi xấp xỉ (2.37) cũng hoàn toàn tương tự. Độ lệch của xấp xỉ phi tuyến (2.36) cao hơn xấp xỉ tuyến tính của (2.33). Tuy nhiên, chúng ta không thể trực tiếp sử dụng (2.36) để khôi phục lại các bit khoá hiệu quả của DES-8 vòng, bởi lẽ số

bit text hiệu quả và khoá hiệu quả trong (2.36) lần lượt là 24 và 26. Để tránh vấn đề này, chúng ta giải quyết mỗi thừa số trong (2.36) một cách độc lập. Biểu thức sau là thừa số thứ hai của (2.36):

$$P_l[16,17,20] + F_1(P_r,K_1)[16,17,20] = K_2[25,26,29] \qquad (2.38)$$

Khi (2.38) xảy ra, độ lệch của (2.33) là $\varepsilon_0 = (1/2)(5/16)p_6 = 8/5p_6$. Khi (2.38) không xảy ra nó thay đổi thành $\varepsilon_1 = 2(1-(8/5)/2)p_6 = 2/5p_6$. Do đó, ta coi xấp xỉ tuyến tính (2.33) như 2 xấp xỉ tuyến tính; một là khi (2.38) xảy ra và một là khi (2.38) không xảy ra. Và như vậy ta sẽ sử dụng phương pháp xấp xỉ bội để giải biểu thức phi tuyến (2.36).

Giả sử N là số cặp rõ/mã, T_0, T_1 lần lượt là số cặp rõ/mã sao cho vế trái của phương trình (2.33) bằng 0 và phương trình (2.38) xảy ra, và phương trình (2.38) không xảy ra. Chúng ta tính toán thống kê $U = a_0T_0 + a_1T_1$, với các trọng số $a_0 + a_1 = 2$ để quyết định khoá ứng cử viên. Việc chọn a_0, a_1 được Kaliski và Robshaw phát biểu trong Bổ đề, theo đó lựa chọn tốt nhất của a_0, a_1 là $a_0 : a_1 = \varepsilon_0 : \varepsilon_1 = 4:1$. Hoàn toàn tương tự ta cũng sẽ xét tới thừa số thứ hai của (2.27) theo một cách tương tự.

$$C_l[16,17,20] + F_8(C_r,K_8)[16,17,20] = K_7[25,26,29] \qquad (2.39)$$

2. Thuật toán cải tiến thám mã DES - 8 vòng

Thuật toán được chia thành 2 phần; phần thứ nhất là tấn công gốc của Matsui (Bước 1, 2, 3), phần thứ hai là phần cải tiến mà thay thế phần vét cạn khoá trong tấn công của Matsui bằng xấp xỉ bội (Bước 4, 5, 6).

Bước 1 Tạo N cặp rõ/mã và đếm các bít text hiệu quả của phương trình (2.330) và (2.38).

Bước 2 Tăng giá trị của các bộ đếm trong tập *K* tương ứng với các bit khoá hiệu quả của (2.33) nếu vế trái của (2.33) bằng không.

Bước3 Sắp xếp các bit khoá hiệu quả của (2.33) sử dụng các bộ đếm *K* theo độ tin cậy.

Bước 4 Với các khoá hiệu quả tin cậy nhất của (2.33) (đã tìm được ở *Bước 3*), khi vế phải của (2.38) bằng 0 tăng các bộ đếm trong tập H_0 tương ứng với các bit khoá hiệu quả của (2.38) một lượng là 1 nếu vế trái của (2.38) bằng không hoặc 4 nếu (2.38) khác không. Ngược lại, khi vế phải của (2.38) bằng 1 tăng các bộ đếm trong tập H_1 tương ứng với các bit khoá hiệu quả của (2.38) một lượng là 4 nếu vế trái của (2.38) bằng không, hoặc 1 nếu vế trái của (2.38) khác không.

Bước 5 Sắp xếp các bit khoá hiệu quả của (2.38) sử dụng các bộ đếm H theo thứ tự độ tin cậy.

Bước 6 Từ các bit khoá hiệu quả tin cậy nhất của (2.33) và (2.38), ta tìm các bit khoá còn lại.

Chú ý:

- 13 bit text hiệu quả của nửa trái phương trình (2.33) là: $P_r[32]$, $P_r[1]$ ... $P_r[5]$, $C_l[16]$ ... $C_l[21]$, $P_r[3,8,14,25] + P_l[17] + C_l[8,14,25]$. (2.40)
- và 12 bit khoá hiệu quả của nửa trái phương trình (2.33) là: $K_1[1]$ ... $K_1[6]$, $K_8[25]$ ... $K_8[30]$. (2.41)
- 17 bit text hiệu quả của nửa trái phương trình (2.38), nếu các khoá trong (2.40) đã cố định là: $P_r[32]$, $P_r[1]$ ... $P_r[5]$, $P_l[16,17,20]$, $P_r[8]$ ... $P_r[17]$. (2.42)
- Và 13 bit khoá hiệu quả của (2.38) nếu các bit khoá trong (2.41) đã được cố định là: $K_1[13]$ ... $K_1[24]$, $K_2[25, 26, 29]$ (2.43)

Việc tìm các bit khoá hiệu quả trong biểu thức phi tuyến (2.27) hoàn toàn tương tự như với biểu thức phi tuyến (2.36). Cuối cùng, chúng tôi trình bày Thuật toán cải tiến chi tiết để tấn công DES-8 vòng :

Chi tiết thuật toán thám mã DES 8 vòng của chúng tôi

Ta định nghĩa một số vector sau:

a = $P_r[3,8,14,25] + P_l[17] + C_l[8,14,25] + F_1(P_r,K_1)[17] + F_8(C,K_8)[8,14,25] \in GF(2)$,
là vế trái của biểu thức (2.33).

b = $P_r[32], P_r[1] ... P_r[5], C_l[16] ... C_l[21], P_r[3,8,14,25] + P_l[17] + C_l[8,14,25] \in GF(2)^{13}$,
là 13 bit text hiệu quả của (2.33).

c = $K_2[26] + K_4[26] + K_5[4] + K_6[26] \in GF(2)$,
là vế phải của (2.33).

k = $K_1[1] ... K_1[6], K_8[25] ... K_8[30] \in GF(2)^{12}$,
là vector 12 bit khoá hiệu quả của (2.33).

d = $P_r[32], P_r[1] ... P_r[5], P_l[16,17,20], P_r[8] ... P_r[17] \in GF(2)^{17}$,
là tập 17 bit text hiệu quả của (2.38).

e = $K_2[25,26,29] \in GF(2)$,
là vế phải của (2.38).

m = $K_1[13] ... K_1[24] \in GF(2)^{12}$,
là tập 12 bit khoá hiệu quả của (2.38).

Tương tự, ta cũng định nghĩa các vector bit khoá, text hiệu quả liên quan tới phương trình phi tuyến (2.37): a', b', c', k', d', e', m'.

Bước 1 *(Chuẩn bị dữ liệu và đếm dữ liệu)*
Tạo N cặp rõ mã ngẫu nhiên $\{(P_1, C_1), ..., (P_N, C_N)\}$. Với mỗi cặp rõ/mã tương ứng ta tính vector các bit text hiệu quả b, b' và tăng các bộ đếm cho bit text hiệu quả của (2.33) là TA[b], TB[b'] lên 1.

Bước 2 *(Đếm khoá hiệu quả của phương trình (2.33))*
Với mỗi một vector khoá ứng cử viên k (k'), tăng bộ đếm KA[k] (KB[k']) lên TA[b] (hoặc TB[b']) nếu vế trái của phương trình (2.33) bằng 0.

Bước3 *(Quyết định các bit khoá hiệu quả của (2.33))*
Tìm giá trị lớn nhất và nhỏ nhất của KA[k], KB[k'] và quyết định các bit khoá theo ***Thuật toán 2*** của Matsui. Giả sử $(k, k') \in GF(2)^{26}$ là vector khoá thu được, ta thực hiện việc đếm dữ liệu lần thứ 2.
Với mỗi cặp rõ/mã tương ứng, ta tăng các bộ đếm cho các bit text hiệu quả của (2.38) (và (2.39)) là TC[d] (và TD[d']) lên 1 nếu vế trái của (2.33) bằng vế phải của (2.33) (tương ứng, nếu vế trái của (2.34) bằng vế phải của (2.34)).

Bước 4 *(Đếm khoá hiệu quả của phương trình (2.38))*
Với mỗi khoá hiệu quả của (2.38), ta tăng bộ đếm tương ứng với 12 bit khoá hiệu quả UC[e,m] một lượng 1 nếu vế trái của (2.38) bằng vế phải hoặc 4 trong trường hợp ngược lại. Tương tự với UD[e', m'] của biểu thức (2.39).

Bước 5 *(Sắp xếp khoá)*
Sắp xếp tập vector khoá $\{h_j = (e, a)\}$, $\{h'_{j'} = (e',a')\}$ thuộc không gian vector hiệu quả (e, m) hoặc (e', m') theo độ lớn của $|UC(h_j) - 5/4N|$, hoặc $|UD(h'_{j'}) - 5/4N|$ tương ứng.

Bước 6 *(Vét cạn số khoá còn lại)*
Với mỗi một vector khoá $(k_i, k'_{i'}, h_j, h'_{j'})$, thực hiện tìm vét cạn 18 bit khoá còn lại cho đến khi tìm ra khoá đúng.

Trong thuật toán cải tiến này, tổng số các bit khoá hiệu quả là 52 bit, tuy nhiên có 14 bit khoá trùng, cụ thể là:

$K_1[4] = K_8[3]$, $K_1[28] = K_8[26]$, $K_1[1] = K_8[5]$, $K_1[21] = K_8[1]$, $K_1[24] = K_8[18]$, $K_1[18] = K_8[13]$, $K_1[16] = K_8[14]$, $K_1[2] = K_8[16]$, $K_1[3] = K_8[17]$, $K_1[5] = K_8[19]$, $K_1[22] = K_8[20]$, $K_1[15] = K_8[21]$, $K_1[6] = K_8[22]$, $K_1[17] = K_8[23]$.

Như vậy, số bit khoá cần phải vét cạn là: 56 - (52-14) = 18 bit khoá.

Chú ý: Trong phần mô tả thuật toán chi tiết ở trên của chúng tôi có sửa lại một số điểm sau so với Thuật toán mà Takeshi Shimoyama và Toshinobu Kaneko đã trình bày:

- Tập các bit text hiệu quả của biểu thức (2.38) là 17 bit chứ không phải 11 bit như T. Shimoyama và T. Kaneko. Bởi lẽ khi xét vế trái của (2.38), khi đó 6 bit khoá ảnh hưởng tới $F_i(X, K_i)[17]$ (tương ứng với hộp S_1) đã được cố định thì 6 bit text (tương ứng với S_1) vẫn ảnh hưởng tới giá trị của $F_i(X, K_i)[17]$.
- Trong Bước 4 việc tăng các bộ đếm tương ứng với tập H_0 (hoặc H_1) lên 1 hoặc 4 ngược thứ tự với T. Shimoyama. Sự sửa đổi này xuất phát từ kết quả thực hành thám DES 8 vòng của chúng tôi.

3. Kết quả thực hành và đánh giá

Chúng tôi đã thể hiện chương trình Demo thám mã DES-8 vòng và chạy thử nghiệm trên Hệ điều hành Linux, và đã thu được các kết quả như sau. Chúng tôi đã chạy thử thám tìm đủ 56-bít khoá lấy ngẫu nhiên, kết quả là về trung bình chương trình chạy mất khoảng 1 ngày trên máy tốc độ 400 MHz sẽ tìm được đủ 56-bít khoá đã cài đặt. Thời gian chạy các bước 1,2,3 trong thuật toán là không đáng kể, thời gian chạy tập trung chủ yếu trong giai đoạn vét cạn tìm 18 bít khoá còn lại sau khi đã thực hiện thuật toán cải tiến đã nêu. Listing chương trình thám mã được cho trong Phụ lục A của tài liệu.

IV. TẤN CÔNG VI SAI BẬC CAO

IV.1. Khái niệm

Trước hết ta định nghĩa khái niệm vi sai (đạo hàm) bậc cao của các hàm mật mã.

Định nghĩa 2.13 (X. Lai): Giả sử (S, +) và (T, +) là các nhóm Abelian. Đối với hàm f: S $\rightarrow$ T, đạo hàm (vi sai) của f tại điểm a $\in$ S được định nghĩa bởi:

$$\Delta_a f(x) = f(x+a) - f(x).$$

Định nghĩa 2.14 (X. Lai): Giả sử hàm f như trong định nghĩa 2.13. Đạo hàm (vi sai) bậc i của f tại điểm a_1, a_2, ..., a_i được định nghĩa bởi công thức:

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = \Delta_{a_i}(\Delta^{(i-1)}_{a_1,\ldots,a_{i-1}} f(x))$$

Chú ý rằng các đặc trưng và vi sai được sử dụng bởi Biham và Shamir trong các tấn công của họ là tương ứng với vi sai (đạo hàm) bậc nhất được mô tả bởi X. Lai. Do đó dường như một cách tự nhiên ta mở rộng khái niệm vi sai thành vi sai bậc cao.

Định nghĩa 2.15: Vi sai một vòng bậc i là một bộ (i+1) có dạng (α_1, ..., α_i, β), sao cho:

$$\Delta^{(i)}_{\alpha_1,\ldots,\alpha_i} f(x) = \beta$$

Khi hàm được xét trên trường với trường cơ sở GF(2) (khi đó phép trừ cũng chính là phép công), các điểm a_1, ..., a_i phải là độc lập tuyến tính để đạo hàm bậc i không nhận giá trị zêro tầm thường.

Mệnh đề 2.16. (X. Lai) Giả sử L[a_1, a_2, ..., a_i] là danh sách của 2^i tổ hợp tuyến tính có thể của a_1, a_2, ..., a_i. Khi đó

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = \sum_{\gamma \in L[a_1,\ldots,a_i]} f(x \oplus \gamma)$$

Nếu a_i là phụ thuộc tuyến tính của a_1, a_2, ..., a_{i-1}, thì

$$\Delta^{(i)}_{\alpha_1,\ldots,\alpha_i} f(x) = 0$$

Ta cũng sử dụng mệnh đề sau đây của X. Lai.

Mệnh đề 2.17. (X. Lai) Giả sử ord(f) là bậc phi tuyến của hàm đa thức nhiều biến f(x). Khi đó:

$$\text{ord}(\Delta_a f(x)) \leq \text{ord}(f(x)) - 1.$$

Điều này dẫn tới mệnh đề sau đây.

Mệnh đề 2.18. Nếu $\Delta^{(i)}_{\alpha_1,\ldots,\alpha_i} f(x)$ là hằng số, khi đó bậc phi tuyến của hàm f là lớn hơn hay bằng i.

Chứng minh: Từ mệnh đề 2.17, có:

$$ord(f) \geq ord(\Delta_{a_1} f(x)) + 1 \geq ... \geq ord(\Delta^{(i)}_{\alpha_1,...,\alpha_i} f(x)) + i$$

IV.2. Tấn công sử dụng vi sai bậc cao

Xét một hệ mã Feistel với cỡ khối là 2n. Giả sử rằng x_R -nửa phải của bản rõ là được cố định như là một hằng số, và ta xem xét vế phải của bản mã y^*_R là đầu ra của hệ mã thu gọn. Từ chỗ x_R là hằng số, các bít trong y^*_R luôn có thể được biểu diễn như là đa thức trên $GF(2)[x_1, x_2,..., x_n]$ theo các bít của $x_L = (x_1, x_2,..., x_n)$. Giả sử rằng đa thức này có bậc không lớn hơn d. Khi đó theo các mệnh đề trên ta có

$$\sum_{x_L \in L_d} p(x_L) = c$$

(2.44)

ở đây L_d là không gian con d-chiều của $GF(2)^n$, c là như nhau đối với bất kỳ không gian con song song với L_d, p là hàm tính toán đầu ra của hệ mã thu gọn. Từ đó suy ra rằng

$$\sigma(w) = \sum_{x_L \in L_{d+1}} p(x_L + w) = 0, \text{ với mọi } w \in GF(2)^n$$

(2.45)

nếu và chỉ nếu p(x) là đa thức bậc d hoặc thấp hơn. Trong thuật toán sau đây các biến $x = (x_L, x_R)$ và $y = (y_L, y_R)$ đóng vai trò bản rõ và bản mã của hệ mật.

L là hạng của ma trận (d+1) × n chiều trên GF(2) và F là hàm vòng.

1. Giả sử x_R và w là các hằng số n-bit
2. Với mọi $a \in GF(2)^{d+1}$:
   (a) Giả sử $x_L = aL + w$.
   (b) Tính bản mã y(a) của bản rõ (x_L, x_R).
1. Với mọi giá trị k của khoá vòng cuối cùng:
   (a) Giả sử $\sigma = 0$.
   (b) Với mọi $a \in GF(2)^{d+1}$:
      i. Giả sử y=y(a).
      ii. Giả sử $y^*_R = y_L \oplus F(k, y_R)$.
      iii. Giả sử $\sigma = \sigma \oplus y^*_R$.

Các khoá làm cho σ trở nên bằng 0 là khoá đúng tại vòng cuối cùng với xác suất cao.

Hệ quả là với mỗi giá trị khoá k có thể tại vòng cuối cùng, chúng ta kiểm tra xem giá trị σ tương ứng có bằng 0 hay không, nếu nó bằng 0, tức là chúng ta đã tìm ra được khoá đúng với xác suất cao. Nếu chúng ta muốn chắc chắn hơn, thì có thể thực hiện lặp lại thuật toán với các lựa chọn khác của giá trị w. Phương pháp này có thể tổng quát hoá đối với các mã lặp, chúng ta phát biểu điều đó bằng định lý sau đây.

Định lý 2.19 [14]. *Cho trước một mã khối lặp, giả sử d là bậc của đa thức của các bít bản mã của vòng sát vòng cuối cùng như là hàm của các bít bản rõ. Ngoài ra, giả sử b là số các bít khoá vòng cuối cùng. Giả sử rằng bậc của đa thức của bản mã là tăng theo số vòng. Khi đó tồn tại một tấn công vi sai bậc d với độ phức tạp thời gian trung bình là 2^{b+d} đòi hỏi 2^{d+1} bản rõ lựa chọn sẽ thành công trong việc mở ra khoá tại vòng cuối cùng.*

Chứng minh. Chúng ta chứng minh trường hợp mã khối lặp dạng Feistel, và từ đó có thể tổng quát hoá lên cho các trường hợp khác. Xét phép lặp (3b). Giả sử k là giá trị khoá chính xác tại vòng cuối cùng, và giả sử k' là giá trị khoá sai. Khi đó

$$y^*_R = y_L \oplus F(k, y_R).$$

$$y^{*\prime}_R = y_L \oplus F(k', y_R).$$

$$= y^*_R \oplus F(k, y_R) \oplus F(k', y_R).$$

Sự sai khác giữa y_R nhận được từ sử dụng khoá đúng với $y^{*\prime}_R$ nhận được từ việc dùng khoá sai, là bằng hai lần áp dụng hàm F. Từ giả thiết bậc của đa thức là tăng theo số vòng, ta có thể hy vọng rằng σ sẽ bằng 0 chỉ với giá trị đúng của khoá vòng cuối cùng với xác suất cao. Việc chạy một thuật toán tương tự như thuật toán trên sẽ mất khoảng 2^{d+1} bước với mỗi giá trị của khoá vòng cuối cùng. Về trung bình, chúng ta phải kiểm tra một nửa số khoá trước khi tìm ra khoá đúng, điều này dẫn tới công thức tính độ phức tạp thời gian.

Tấn công trên có thể được cải tiến bởi một thừa số là 2, nếu hằng số của phương trình (2.44) là có thể dự đoán được. Trong trường hợp đó các phép lặp (2) và (3b) của thuật toán trên là được thực hiện chỉ với mọi a $\in GF(2)^d$. Giá trị khoá tương ứng với σ=c sẽ là khoá đúng với xác suất cao. Đối với hầu hết các mã khối, phụ thuộc vào hàm F có thể mở rộng tấn công trên. Ta cũng có thể tấn công vào một tập con của khoá vòng

cuối cùng, hoặc cũng có thể tìm kiếm khoá (một tập con) của khoá vòng đầu tiên.

Sau đây chúng ta áp dụng tấn công trên vào hệ mã $\mathcal{KN}$. Chúng ta chọn các bản rõ mà vế phải là cố định. Từ chỗ các bít đầu ra của hàm vòng chỉ là bậc hai đối với các bít đầu vào, các đa thức trong tấn công được mô tả trên đối với phiên bản 6-vòng có bậc không lớn hơn 8. Do đó tấn công chỉ đòi hỏi $2^{8+1} = 512$ bản rõ lựa chọn và thời gian chạy trung bình là vào cỡ 2^{41}. Một biến thể cho tấn công tìm khoá tại hai vòng cuối cùng đòi hỏi khoảng 32 bản rõ lựa chọn và thời gian chạy trung bình là 2^{70}. Tương tự có các tấn công trên các phiên bản 7 hay 8 vòng của hệ $\mathcal{KN}$, độ phức tạp tương ứng là 2^9, 2^{74}; và 2^{17}, 2^{82}. Việc tấn công sử dụng vi sai bậc cao đối với $\mathcal{KN}$ đã được chạy thử, và nó đã tìm ra khoá đúng tại vòng cuối cùng đúng như dự đoán. Chú ý rằng tấn công này có thể áp dụng cho các hệ mã với cỡ khối bất kỳ 2n, với số các bản rõ lựa chọn nhỏ hơn 2^n. Với các hệ mã cỡ khối lớn hơn hay số vòng nhiều hơn cũng có thể bị tấn công.

(Mô tả hệ mã $\mathcal{KN}$: là hệ mã khối Feistel 64-bit. Hàm F: $GF(2^{32}) \to GF(2^{32})$ cho bởi công thức:

$$F(k, x) = d(f(e(x) \oplus k)),$$

ở đây $f: GF(2^{33}) \to GF(2^{33}), f(x) = x^3$, $k \in GF(2^{33})$, $e: GF(2^{32}) \to GF(2^{33})$ là hàm mở rộng đối số của nó bằng cách nối thêm một tổ hợp affine của các bít đầu vào, và $d: GF(2^{33}) \to GF(2^{32})$ là hàm bỏ đi một bít từ đối số của nó.)

V. TẤN CÔNG NỘI SUY

Trong phần này chúng ta giới thiệu một kiểu tấn công mới đối với mã khối. Tấn công này dựa trên công thức nổi tiếng sau đây.

Giả sử R là một trường. Cho trước 2n phần tử $x_1,..., x_n, y_1,..., y_n \in R$, ở đây các x_i là khác nhau. Định nghĩa

$$f(x) = \sum_{i=1}^{n} y_i \prod_{1 \le j \le n, j \ne i} \frac{x - x_j}{x_i - x_j} \qquad (2.46)$$

Khi đó f(x) là đa thức trên R với bậc nhiều nhất là n-1 sao cho $f(x_i) = y_i$ với mọi i = 1,...,n. Phương trình (2.46) được gọi là công thức nội suy

Lagrange. Trong tấn công nội suy chúng ta sẽ xây dựng các đa thức bằng cách sử dụng các cặp bản rõ và mã. Chúng ta sẽ giả thiết rằng thời gian cần thiết để xây dựng các đa thức này là nhỏ so với thời gian cần thiết để mã hoá các bản rõ trong tấn công này.

Xét hệ mã khối *PURE* với r-vòng. Chúng ta lợi dụng yếu tố thao tác XOR được sử dụng trong hệ mã tương ứng với phép cộng trên trường hữu hạn với đặc số 2. Hệ quả là, hệ mã này chỉ bao gồm các phép toán đại số đơn giản, và do đó mỗi một trong hai nửa của bản mã y, chẳng hạn nửa trái của nó là có thể được mô tả như là đa thức $p(x_L, x_R) \in GF(2^{32})[x_L, x_R]$ của bản rõ với nhiều nhất là $3^{2r-1} + 3^r + 3^{r-1} + 1$ hệ số (chú ý rằng bậc của x_R và x_L cao nhất là 3^r và 3^{r-1} tương ứng và $(3^{2r-1} + 3^r + 3^{r-1} + 1) = (3^r + 1)(3^{r-1} + 1)$). Như vậy chúng ta có thể xây dựng đa thức này bằng cách xét nhiều nhất là $3^{2r-1} + 3^r + 3^{r-1} + 1$ cặp rõ/mã (p/c cặp) bằng cách sử dụng công thức nội suy Lagrange. Với r=6 tấn công này cần nhiều nhất 2^{18} cặp rõ/mã đã biết, chúng sẽ cung cấp một thuật toán để suy diễn tổng thể. Chú ý rằng số các hệ số là thấp hơn so với ấn định trên, từ chỗ không phải tất cả các phần tử $x_L^i x_R^j$ với $0 \le i \le 3^r$ và $0 \le j \le 3^{r-1}$ đều xuất hiện trong đa thức đó.

Chúng ta có định lý tổng quát sau

Định lý 2.20. *Xét một mã khối lặp với cỡ khối là m. Biểu diễn bản mã như là một đa thức của bản rõ và giả sử n ký hiệu là số các hệ số của đa thức này. Nếu $n \le 2^m$, thì sẽ tồn tại một tấn công nội suy với độ phức tạp thời gian là n đòi hỏi n bản rõ đã biết được mã bởi cùng một khoá mật K, chúng sẽ thiết lập một thuật toán tương đương với phép mã hoá (hay phép giải mã) với khoá K.*

Trong biến thể tấn công bản rõ lựa chọn của tấn công này, nó có thể cho kẻ tấn công thiết lập các đa thức với số hệ số ít hơn bằng cách cố định một vài bít trong các bản rõ lựa chọn. Trong trường hợp đó, kết quả là một sự suy diễn tức thì, từ chỗ thuật toán nhận được có thể mã các bản rõ với một số bít cố định bằng các giá trị nào đó. Như lấy hệ *PURE* làm ví dụ, *PURE* có thể bị tấn công theo cách như thế nhờ sử dụng chỉ 730 cặp rõ/mã lựa chọn. Hệ quả là kẻ tấn công đã có một thuật toán, chúng mã hoá 2^{32} bản rõ mà không cần biết khoá mật.

(*Mô tả hệ PURE: là hệ mã khối Feistel 64-bit. Hàm F: $GF(2^{32}) \to GF(2^{32})$ cho bởi công thức:*

$$F(k, x) = f(x \oplus k),$$

ở đây $f: GF(2^{32}) \rightarrow GF(2^{32}), f(x) = x^3$, tức là đầu vào của hàm lập phương là không được mở rộng cũng không bị cắt bớt như trong ví dụ hệ mã 𝒦𝒩.)

Chý ý: cả hai hệ mã 𝒦𝒩 và 𝒫𝒰ℛℰ đều là an toàn chống lại được tấn công vi sai và tấn công tuyến tính (chỉ cần số vòng bằng 6). Tuy nhiên chúng đều có nhược điểm là bậc phi tuyến của đầu ra là thấp theo các biến đầu vào hệ mã, và chúng có thể bị lợi dụng để tấn công như đã nói trên.

Khôi phục khoá

Trong phần này chúng ta sẽ mở rộng phương pháp trên để thực hiện tấn công khôi phục khoá. Trước hết xét tấn công bản rõ đã biết. Thay vì việc ấn định bản mã như là một hàm của bản rõ, chúng ta sẽ biểu diễn đầu ra từ hệ mã thu gọn y* như là đa thức $p(x) \in GF(2^m)[x]$ của bản rõ. Giả sử rằng đa thức này có bậc d và rằng (d+1) cặp rõ/mã là có thể có được. Khi đó với mọi giá trị của khoá vòng cuối cùng ta thực hiện giải mã các bản mã một vòng và thử xây dựng đa thức. Với một cặp rõ/mã thêm vào hãy kiểm tra xem đa thức có đúng hay không. Nếu nó đúng, thì giá trị chính xác của khoá vòng cuối cùng đã được tìm thấy với xác suất cao, bởi lý do tương tự như trong chứng minh định lý 2.19.

Biến thể cho tấn công bản rõ lựa chọn là hoàn toàn tương tự. Chúng ta sẽ minh hoạ phương pháp này bằng một ví dụ, lấy chính hệ mã 𝒫𝒰ℛℰ 6-vòng. Giả sử vế phải x_R của bản rõ là cố định, và xét vế phải của đầu ra $y^*_R = p(x_L)$ từ hệ mã rút gọn như là đa thức $p(x_L) \in GF(2^{32})[x_L]$. Đa thức này có bậc nhiều nhất là $3^3 = 27$ từ chỗ bậc của nó không tăng trong vòng đầu tiên và do y^*_R là bằng vế trái của đầu ra tại vòng thứ 4. Hệ quả là 28 cặp các giá trị tương ứng của x_L và y* là đủ để xác định nó một cách duy nhất (sử dụng nội suy Lagrange).

Chúng ta sẽ kiểm tra xem y* có phải là đầu ra thực sự của hệ mã rút gọn hay không. Thực hiện điều này bằng cách kiểm tra xem 29 cặp rõ/mã có thoả mãn đa thức đó hay không. Nếu đúng, khi đó có thể xem rằng chúng ta đã tìm ra khoá đúng tại vòng cuối cùng. Độ phức tạp thời gian trung bình sẽ là $29 \cdot 2^{32-1} \sim 2^{36}$. Tổng quát ta có định lý sau.

Định lý 2.21 [14]. *Xét một mã khối lặp với cỡ khối là m. Biểu diễn đầu ra từ vòng gần cuối cùng như là một đa thức của bản rõ và giả sử n ký hiệu là số các hệ số của đa thức này. Ngoài ra, giả sử ký hiệu b là số các bít khoá vòng cuối cùng. Khi đó tồn tại một tấn công nội suy với độ phức tạp thời gian trung bình là $2^{b-1}.(n+1)$ đòi hỏi n+1 bản rõ đã biết (hoặc lựa chọn) chúng sẽ thành công trong việc mở ra khoá tại vòng cuối cùng.*

Tương tự như tấn công trong định lý 2.19, ta có thể thực hiện tấn công chỉ trên một tập con của các khoá vòng cuối cùng hay chỉ tìm kiếm một tập con của khoá tại vòng đầu tiên (phụ thuộc vào cấu trúc của hàm vòng).

Tiếp cận kiểu tấn công gặp nhau ở giữa

Mục này sẽ mở rộng tấn công trên theo cách tiếp cận gặp nhau ở giữa. Chúng ta chỉ trình bày đối với tấn công mở khoá đã xét trên.

Ngay lập tức chúng ta thử gợi ra giá trị khoá đúng vòng cuối cùng và sử dụng nó để (hy vọng) đạt được y*, đầu ra từ hệ mã rút gọn. Sau đây ta chỉ xác minh cho y* đã mô tả. Cho trước một mã khối lặp r-vòng, giả sử z là đầu ra tại vòng s, ở đây $s \leq (r-1)$. Giá trị của z biểu diễn theo bản rõ x như là đa thức $g(x) \in GF(2^m)[x]$, ở đây m là cỡ khối. Tương tự, z có thể biểu diễn như đa thức $h(y^*)) \in GF(2^m)[y^*]$ của đầu ra y* của hệ mã rút gọn. Giả sử bậc của g(x) là d_g, bậc của h(y*) là d_h và ký hiệu $d_{gh} = d_g + d_h$. Như vậy, phương trình sau đây $g(x) = h(y^*)$

có nhiều nhất là d_{gh} +2 ẩn số. Phương trình này sẽ được giải bằng phép nhân và phép công cả g và h với một hằng số. Do đó, để đảm bảo rằng chúng ta sẽ đạt được một lời giải không tầm thường và duy nhất, chúng ta phải đặt các hệ số tương ứng với luỹ thừa cao nhất là bằng 1 và số hạng hằng số là bằng 0. Sau đó, chúng ta giải phương trình này bằng cách sử dụng d_{gh} bản rõ đã biết hoặc lựa chọn. Sau đó chúng ta kiểm tra xem với cặp rõ/mã (x, y*) khác có thoả mãn $g(x) = h(y^*)$ hay không. Nếu đúng, ta có thể xem rằng đã đoán đúng khoá tại vòng cuối cùng.

Ta cũng sẽ minh hoạ điều này bằng tấn công hệ *PURE* 6-vòng.

Giả sử rằng vế phải x_R của bản rõ là cố định. Ký hiệu z_L là nửa trái của đầu ra tại vòng thứ 4. Giá trị z_L được biểu diễn theo bản rõ như là đa thức $g(x_L) \in GF(2^{32})[x_L]$. Đa thức này có bậc tối đa là 3^2, tức là có nhiều nhất là 10 hệ số khác không trong $g(x_L)$. Tương tự z_L cũng có thể biểu diễn như là đa thức $h(y^*_L, y^*_R) \in GF(2^{32})[y^*_L, y^*_R]$ của đầu ra của hệ mã rút gọn. Từ đó suy ra rằng $h(y^*_L, y^*_R) = \left(y_L^*\right)^3 \oplus a\left(y_L^*\right)^2 \oplus by_L^* \oplus c \oplus y_R^*$, ở đây a, b, c là các hệ số phụ thuộc khoá nào đó. Như vậy có nhiều nhất là 10+3=13 hệ số chưa biết trong phương trình

$$g(x_L) = h(y^*_L, y^*_R) \qquad (2.47)$$

Đặt số hạng hằng số của g bằng 0 (hệ số tương ứng với luỹ thừa cao nhất trong h đã bằng 1 như đã thấy), sau đó ta tiến hành giải hệ thống kết quả

phương trình bằng cách sử dụng 12 cặp rõ/mã (x, y*) từ hệ mã rút gọn. Điều này sẽ cho ta các đa thức g, h. Sau đó ta kiểm tra bằng một cặp rõ/mã khác (x, y*) xem có cho đẳng thức $g(x_L) = h(y^*_L, y^*_R)$ đúng hay không. Nếu đúng, ta đã đoán ra được khoá đúng.

Tấn công tương tự có thể áp dụng cho phiên bản hệ *PURE* 32-vòng. Giả sử ký hiệu $g(x_L) \in GF(2^{32})[x_L]$ là biểu diễn đa thức của z_L tại đầu ra của vòng thứ 22. Bậc của đa thức này tối đa là 3^{20}. Giả sử $h(y^*_L, y^*_R) \in GF(2^{32})[y^*_L, y^*_R]$ là một biểu diễn đa thức của z_L đối với đầu ra của hệ mã đang xét. Trong dạng chuẩn đại số của $h(y^*_L, y^*_R)$, số của luỹ thừa trong y^*_Lvà y^*_R tối đa là $(3^9 + 1)$ và $(3^{10} + 1)$. Như vậy số các hệ số trong $h(y^*_L, y^*_R)$ tối đa là $(3^9 + 1).(3^{10} + 1) \sim 3^{19}$. Điều đó có nghĩa rằng số các hệ số trong phương trình (2.20) nhiều nhất là $3^{20} + 3^{19} \sim 2^{32}$, tức là độ phức tạp thời gian trung bình của tấn công này là 2^{63} và nó đòi hỏi khoảng 2^{32} bản rõ lựa chọn. Từ đó ta có kết quả sau.

Định lý 2.22. *Xét một mã khối lặp r-vòng với cỡ khối là m. Biểu diễn đầu ra từ vòng thứ s, $s \leq (r-1)$ như là một đa thức của bản rõ và giả sử n_1 ký hiệu là số các hệ số của đa thức này. Cũng vậy, biểu diễn đầu ra từ vòng thứ s như là một đa thức của đầu ra tại vòng thứ (r-1), và giả sử n_2 ký hiệu là số các hệ số của đa thức này. Ngoài ra, đặt $n=n_1 + n_2$ và giả sử ký hiệu b là số các bít khoá vòng cuối cùng. Khi đó tồn tại một tấn công nội suy với độ phức tạp thời gian trung bình là $2^{b-1}.(n-1)$ đòi hỏi n-1 bản rõ đã biết (hoặc lựa chọn) sẽ thành công trong việc mở ra khoá tại vòng cuối cùng.*

*Để tránh tấn công vi sai bậc cao và tấn công nội suy, cần phải đảm bảo bậc đại số cao của các S-hộp phi tuyến.

VI. TẤN CÔNG KHOÁ QUAN HỆ

Trong phần này chúng ta sẽ trình bày một kiểu tấn công mới trên các mã khối: rút gọn độ phức tạp của việc vét cạn khoá trên cơ sở tấn công bản rõ lựa chọn và tấn công khoá lựa chọn mà trong đó chỉ chỉ có quan hệ giữa cặp các khoá quan hệ là được chọn bởi kẻ tấn công và anh ta không hề biết chính các khoá đó. Các tấn công bản rõ lựa chọn rút gọn độ phức tạp của tìm kiếm vét cạn và độ phức tạp này nhanh hơn tấn công bản rõ lựa chọn trên cơ sở tính chất bù tới 3 lần. Các tấn công khoá lựa chọn có độ phức tạp rất thấp, tuy nhiên chúng chỉ có thể được sử dụng khi kẻ tấn

công có thể chọn được mối quan hệ giữa các khoá chưa biết và mong muốn biết chính các khoá đó.

Các tấn công này dựa trên cơ sở quan sát thấy rằng trong nhiều hệ mã khối, chúng ta có thể xem lược đồ tạo khoá như là một tập các thuật toán, mỗi một trong chúng trích ra một khoá con cụ thể từ các khoá con của một số ít vòng trước đó. Nếu tất cả các thuật toán trích khoá con của các vòng khác nhau là như nhau, thì cho trước một khoá chúng ta có thể dịch toàn bộ các khoá con ngược trở lại một vòng và có một tập hợp mới các khoá con có giá trị mà chúng có thể được thiết lập từ một vài khoá khác. Chúng ta gọi các khoá này là các khoá quan hệ.

Một đặc trưng rất hay của các tấn công dựa trên khoá quan hệ là chúng là độc lập với số vòng của hệ mã đang bị tấn công. Các tấn công này có thể áp dụng vào cả hai biến thể của LOKI và cả đối với hệ Lucifer. Tuy nhiên, chúng không thể áp dụng được vào DES, vì quan sát thấy rằng số các bước dịch của các thanh ghi khoá (C và D) trong lược đồ tạo khoá là không như nhau. Đương nhiên, nếu các phép dịch 1-bit trong lược đồ khoá của DES được thay thể bởi phép dịch 2-bít, thì khi đó DES cũng bị tổn thương đối với kiểu tấn công này.

Áp dụng tiềm tàng khác của khoá quan hệ là việc phân tích các hàm HASH (hoạc là hàm hash dựa trên các mã khối, hoặc các hàm hash tổng quan). Trong các hàm hash như thế, có thể chọn thông báo theo cách sao cho tính chất khoá quan hệ gợi ý một thông báo thêm vào với cùng một giá trị hash. Hiện tại, chúng tôi chưa ghi nhận một áp dụng cụ thể như thế đối với các hàm hash, nhưng người thiết kế hàm hash nên cẩn thận để thiết kế các hàm hash miễn dịch đối với điểm yếu này.

Kết quả cụ thể của tấn công này như sau. Độ phức tạp của tấn công bản rõ lựa chọn trên LOKI89 là khoảng1.5. 2^{54}, gần như nhanh hơn 3 lần so với các tấn công bản rõ lựa chọn đã báo cáo trước đây. Tấn công bản rõ lựa chọn khoá lựa chọn mất chừng khoảng vài giây trên một máy tính cá nhân và độ phức tạp của nó là 2^{17}, và độ phức tạp tấn công bản rõ đã biết khoá lựa chọn là khoảng 2^{32}. Độ phức tạp tương ứng trên LOKI91 là 1.375. 2^{61}, 2^{32}, và 2^{48}. Độ phức tạp của tấn công bản rõ lựa chọn khoá lựa chọn trên Lucifer là khoảng 2^{33}. Các hệ DES, IDEA và FEAL không bị tổn thương bởi tấn công này.

Tấn công khoá lựa chọn.

Trong tấn công khoá lựa chọn, hai khoá có quan hệ với quan hệ nào đó là được sử dụng và một vài bản rõ là được mã hoá dưới mỗi một trong chúng. Kẻ tấn công chỉ biết quan hệ giữa hai khoá, nhưng không hề biết chính chúng. Anh ta nhận được các bản mã và sử dụng chúng để tìm ra cả hai khoá. Hai loại tấn công khoá lựa chọn được nghiên cứu là: tấn công bản rõ đã biết khoá lựa chọn trong đó chỉ có quan hệ giữa các khoá là được chọn bởi kẻ tấn công, và tấn công bản rõ lựa chọn khoá lựa chọn trong đó kẻ tấn công chọn quan hệ giữa các khoá cũng như các bản rõ để mã hoá. Các tấn công này là độc lập đối với số vòng của hệ mã bị tấn công, ngay cả khi số vòng là tăng lên (đặc biệt nếu là gấp đôi số vòng ban đầu) thì hệ mã vẫn bị tổn thương với cùng một tấn công.

LOKI89

Trong LOKI89 mỗi phép chọn hai khoá con, một từ vòng lẻ và một từ vòng chẵn, ta có một khoá 64-bit tương ứng. Từ chỗ tất cả các thuật toán thiết kế các khóa con từ hai khoá con đứng trước (preceding) là như nhau, vị trí của các vòng trong đó hai khoá con có mặt không ảnh hưởng tới việc thiết kế các khoá con tiếp theo. Nếu chúng ta chỉ cố định hai khoá con K2 và K3 của khoá K, và định nghĩa K* bằng cách chọn K1* = K2 và K2* =K3, thì các giá trị của khoá con Ki* của K* là bằng các khoá con K(i+1) của K. Trong trường hợp này, K* = (K2, K3) = (K_R, ROL12(K_L)). Do đó tính chất sau đây là đúng với bất kỳ hai khoá quan hệ như thế: Nếu dữ liệu trước vòng thứ hai trong phép mã hoá với khoá K bằng dữ liệu trước vòng đầu tiên trong phép mã hoá với khoá K*, thì dữ liệu và các đầu vào của hàm F là như nhau trong cả hai cách thực hiện với sự sai khác một vòng. Trong trường hợp này, nếu bản rõ P được mã hoá dưới khoá K, thì dữ liệu trước vòng thứ hai sẽ là ($P_R \oplus K_R$, $P_L \oplus K_L \oplus F(P_R \oplus K_R, K_L)$). Dữ liệu này là bằng dữ liệu trước vòng đầu tiên trong phép mã hoá với khoá K*, giá trị của chúng là $P^* \oplus K^* = (P_L^* \oplus K_R, P_R^* \oplus ROL12(K_L))$ và như vậy trong một cặp

$$P^* = (P_R, P_L \oplus K_L \oplus ROL12(K_L) \oplus F(P_R \oplus K_R, K_L)) \quad (2.48)$$

Chúng ta thấy rằng vế phải của P bằng vế trái của P* và quan hệ giữa hai nửa còn lại là phụ thuộc vào khoá. Trong một cặp như thế, tưng tự cũng có mối quan hệ giữa các bản mã

$$C^* = (C_R \oplus K_L \oplus ROL12(K_L) \oplus F(C_L \oplus K_R, K_L), C_L) \quad (2.49)$$

Tấn công bản rõ lựa chọn khoá lựa chọn, dựa trên tính chất này thực hiện bằng cách chọn một giá trị 32-bit P_R, 2^{16} bản rõ P_0, ..., P_{65535} mà

vế phải của chúng bằng P_R và các vế trái 32-bit là được chọn ngẫu nhiên, và 2^{16} bản rõ P_0*, ..., P^*_{65535} mà vế trái của chúng bằng P_R và các vế phải 32-bit là được chọn ngẫu nhiên. Hai khoá quan hệ chưa biết được sử dụng để mã hoá các bản rõ này trên một máy mục tiêu: khoá K được dùng để mã 2^{16} bản rõ đầu tiên và khoá $K^*=(K_R, ROL12(K_L))$ dùng để mã 2^{16} bản rõ còn lại. Trong mỗi cặp bản rõ P_i và P_j*, chúng ta đảm bảo rằng $P^*_{jR} = P_i$ và theo nghịch lý ngày sinh với xác suất cao tồn tại hai bản rõ P_i và P_j* sao cho

$$P^*_{jR} = P_{iL} \oplus K_L \oplus ROL12(K_L) \oplus F(P_{iR} \oplus K_R, K_L).$$

Trong một cặp như thế dữ liệu là như nhau trong cả hai cách thực hiện được dịch đi một vòng. Dễ dàng nhận ra được cặp này, bằng cách kiểm tra xem $C^*_R = C_L$ hay không. Phép kiểm tra này có xác suất thành công là 2^{-32}, và như vậy chỉ có ít cặp qua được kiểm tra này.

Một cặp với tính chất này liên quan tới các giá trị của hai bản rõ và hai bản mã tới khoá bởi các phương trình 1 và 2. Như vậy, một cặp như thế sẽ tiết lộ ra giá trị của

$$F(P_R \oplus K_R, K_L) \oplus F(C_L \oplus K_R, K_L) = P^*_R \oplus P_L \oplus C^*_L \oplus C_R$$

trong đó chỉ có giá trị chưa biết là $K_L \oplus K_R$. Thực hiện duyệt cả 2^{32} giá trị có thể của $K_L \oplus K_R$, chỉ có vài giá trị là thoả mãn phương trình. Sau khi tìm được $K_L \oplus K_R$, dễ dàng tính toán giá trị $K_L \oplus ROL12(K_L)$ bởi phương trình (2.48) và (2.49) từ đó thiết lập được các khoá K và K*.

Tương tự tấn công bản rõ đã biết khoá lựa chọn sử dụng 2^{32} bản rõ đã biết P_i được mã hoá dưới khoá K và dụng 2^{32} bản rõ đã biết P^*_j được mã hoá dưới khoá $K^* = K_R \oplus ROL12(K_L)$. Theo nghịch lý ngày sinh với xác suất cao tồn tại một cặp thoả mãn tính chất đã nêu. Dễ dàng nhận ra được cặp này bằng 32 bít chung của các bản rõ và 32 bít chung của các bản mã. Cặp này có thể sử dụng để bóc ra khoá theo cùng cách như trong tấn công bản rõ lựa chọn khoá lựa chọn.

Tấn công bản rõ lựa chọn

Trong phần này chúng ta mô tả tấn công bản rõ lựa chọn chúng rút gọn độ phức tạp của tấn công vét cạn bằng cách sử dụng khoá quan hệ. Tấn công này có thể được kết hợp với các tấn công trên cơ sở tính chất bù và phiên bản nhanh nhất của nó hầu như gấp 3 lần nhanh hơn tấn công tương ứng chỉ dựa trên tính chất bù. Khi tấn công này được áp dụng trên các mã khối

64-bit, số bản rõ đòi hỏi khoảng từ 2^{32} đến 2^{37}, các bản mã tương ứng được lưu trong bộ nhớ truy nhập ngẫu nhiên.

Ý tưởng ở đây là tương tự với tấn công dựa trên tính chất bù của DES. Với tính chất bù của DES gợi ra rằng khi một bản rõ P được mã hoá dưới khoá K thành bản mã C = DES(P, K), thì phần bù của P được mã hoá bởi phàn bù của K cũng sẽ được bản mã là phần bù của nó C* = DES(P*, K*). Tấn công này chọn một cặp bù nhau của các bản rõ P_1 và $P_2 = P_1$*. Cho trước các bản mã của chúng là C_1 = DES(P_1, K) và C_2 = DES(P_2, K) dưới cùng một khoá K, kẻ tấn công tìm kiếm khoá k *bằng cách thử tất cả các khoá K′ với bít có nghĩa lớn nhất của nó bằng 0,* (tức là một nửa không gian khoá). Với mỗi khoá như thế, anh ta mã hoá P_1 thu được bản mã C' = DES(P_1, K'). Nếu C' =C_1, thì K'= K. Thêm vào đó, kẻ tấn công có thể dự đoán được (có được luôn) bản mã của P_1 dưới khoá K'* là C'$_2$ mà không cần thực hiện mã hoá. Nếu C' = C_2* thì ta có ngay K = K'*, do tính chất bù C_2* = DES(P_1, K*). Ngược lại, cả K' và K'* đều không phải là khoá K. Tấn công này có thể thực hiện dưới một kiểu tấn công bản rõ đã biết, cho trước khoảng 2^{33} bản rõ đã biết, theo nghịch lý ngày sinh với xác suất cao sẽ tồn tại hai bản rõ bù nhau trong số 2^{33} bản rõ ngẫu nhiên. Và khi đó ta sẽ lặp lại thủ tục tấn công như trên đã mô tả.

LOKI89 có một vài tính chất bù. Một tính chất bù của khoá gây ra là một khoá bất kỳ đều có 15 khoá tương đương, chúng mã bản rõ bất kỳ thành cùng bản mã. 15 khoá đó là khoá ban đầu (nguyên thuỷ) XOR với 15 số hexadecimal 64-bit mà các chữ số của chúng là như nhau. Mã hoá với các các khoá này kết quả là đều cho cùng các đầu vào tới hàm F trong tất cả 16 phép mã thực hiện. Do vậy hầu hết các khoá đều có độ dư và tấn công bản rõ đã biết có thể thực hiện với độ phức tạp 2^{60} thay cho 2^{64}.

Một tính chất bù khác của LOKI89 là do quan sát thấy rằng phép XOR khoá với giá trị hexadecimal $gggggggghhhhhhhh_x$ và XOR bản rõ với số $iiiiiiiiiiiiiiii_x$ ở đây g, h $\in \{0_x, ..., F_x\}$ và i =g $\oplus$ h, kết quả là bằng bản mã XOR với giá trị $iiiiiiiiiiiiiiii_x$. Tính chất này có thể sử dụng để rút gọn độ phức tạp của tấn công bản rõ lựa chọn bởi một thừa số lớn hơn là 16 để độ phức tạp còn là 2^{56}.

Tấn công chúng tôi trình bày trong phần này là có thể đoán được các giá trị của bản mã thêm vào được tạo từ các bản rõ thêm vào dưới các khoá quan hệ. Giả sử P là bản rõ bất kỳ, K là khoá bất kỳ và C là bản mã tương ứng C= LOKI89(P, K). Giả sử K*=(K_R, ROL12(K_L)) và P* là bản rõ mà dữ liệu của nó trước vòng đầu tiên của phép mã hoá dưới khoá K*

là giống như dữ liệu trước vòng thứ hai của phép mã hoá nguên thuỷ của P dưới khoá K. Khi đó, 15 vòng đầu tiên của phép mã C* = LOKI89(P*, K*) có chính xác cùng dữ liệu và các khoá con như 15 vòng cuối cùng của phép mã nguyên thuỷ của P, và nửa phải của C* là bằng với nửa trái của C (tức là $C_R^* = C_L$).

Với mỗi một khoá, có một khoá tương đương với 4 bít có nghĩa nhất của nó là bằng 0, và một khoá bù với 4 bít có nghĩa nhất của cả hai nửa đều là 0. Trong định nghĩa sau đây, phép toán next thực hiện phép quay một nửa khoá đi 12 bit (như trong lược đồ khoá trên mỗi vòng), và tìm được giá trị tương đương của kết quả.

Định nghiã 2.23: Toán tử next tác động trên giá trị 32-bit, quay nó đi 12 bít sang trái (ROL12) và XOR nó với một số hexadecimal 32-bit mà tất cả các chữ số của nó là bằng nhau, sao cho 4 bít có nghĩa nhất của kết quả là bằng 0.

Trong tấn công của chúng ta, ta sử dụng quan sát rằng dịch đi một vòng của dữ liệu và các khoá con có thể được thực hiện trong cả hai hướng xuôi và ngược. Do đó, mỗi khoá đem thử có thể dự đoán các bản mã dưới 3 khoá quan hệ, và như vậy tấn công này chỉ đòi hỏi thử khoảng 1/3 số khoá đòi hỏi bởi tấn công dựa trên tính chất bù. Thường không thể tìm ra một tập con các khoá thoả mãn điều kiện quan hệ mà lại chứa đúng 1/3 số khoá toàn bộ. Phép chọn tốt nhất cho LOKI89 là thử 3/8 số khoá. Chúng ta thực hiện trước một danh sách của các nửa khoá $\{L_i\}$ với tính chất: (1) 4 bít có nghĩa nhất của tất cả các khoá trong danh sách đều bằng 0, (2) Danh sách chứa đúng một giá trị từ mỗi một chu kỳ của toán tử next. Danh sách này chứa khoảng 2^{25} nửa khoá. Danh sách này có thể lưu trong một bộ nhớ khoảng $4.2^{25} = 2^{27}$ bytes, hoặc như một bitmap sử dụng $2^{28}/8 = 2^{25}$ bytes.

Tấn công đòi hỏi 2^{37} bản rõ lựa chọn mà bản mã của chúng được lưu trong bộ nhớ truy nhập ngẫu nhiên RAM (cỡ khoảng 2^{40} bytes). Tấn công cụ thể như sau:

1. Chọn bản rõ bất kỳ P_0 , tính toán 15 bãn rõ P_i, $i \in \{1_x, ..., F_x\}$ bởi công thức $P_i = P_0 \oplus iiiiiiiiiiiiiii_x$.
2. Với mỗi bản rõ P_i, chọn 2^{32} bản rõ thêm vào $P_{i,k} = (P_{iR}, P_{iL} \oplus k)$ với nửa trái của chúng bằng nửa phải của P_i còn nửa phải của chúng nhận được tất cả các giá trị có thể bởi phép XOR tất cả các giá trị 32-bit có thể k với nửa trái của P_i.

3. Với mỗi bản rõ P_i, chọn 2^{32} bản rõ thêm vào $P^*_{i,k} = (P_{iR} \oplus k, P_{iL})$ với nửa phải của chúng bằng nửa trái của P_i còn nửa trái của chúng nhận được tất cả các giá trị có thể bởi phép XOR tất cả các giá trị 32-bit có thể k với nửa phải của P_i.
4. Cho trước các bản mã $\{C_i\}$, $\{C_{i,k}\}$,$\{C^*_{i,k}\}$,thử với mỗi cặp nửa khoá (l_i, L_j) tất cả 24 khoá K' có dạng K' = (RORm(L_i), ROLn(L_j)), ở đây m là bội của 4 và n =m, n=m+4, hoặc n=m+8.
5. Mã hoá bản rõ P_0 dưới mỗi khoá thử K' thành bản mã C' = LOKI89(P_0, K').
6. Nếu C' bằng một trong những giá trị $C_i \oplus$ iiiiiiiiiiiiiiii$_x$, thì khoá ban đầu là K = K'$\oplus$ 00000000iiiiiiii$_x$.hoặc là một trong 15 khoá tương đương của chúng.
7. Cố định k là đầu ra của hàm F trong vòng đầu tiên của phép mã hoá P_0 dưới khoá K'. Nếu C'_L bằng một trong 16 giá trị $C_{i,kR}\oplus$ iiiiiiii$_x$, tiếp tục mã hoá P_0 với một vòng thứ 17 (Có thể tính toán thêm một vòng từ C' sử dụng khoá con K17' chúng có thể dễ dàng được thiết kế từ khoá K'), và nếu kết quả C'' bằng $C_{i,k} \oplus$ iiiiiiiiiiiiiiii$_x$, khi đó khoá bí mật có thể là K = (K'_R, ROL12 (K'_L) $\oplus$ iiiiiiii$_x$) hoặc là bất kỳ một trong 15 khoá tương đương.
8. Tính toán ngược trở lại một vòng từ P_0 sử dụng khoá con K'0 chúng có thể dễ dàng được thiết kế từ khoá K', và cố định k là đầu ra của hàm F trong vòng đó. Nếu dữ liệu sau vòng thứ 15 trong phép mã P_0 dưới khoá K' bằng một trong 16 giá trị $C_{i,k} \oplus$ iiiiiiiiiiiiiiii$_x$ (với i nào đó), khoá bí mật có thể là K = (ROL12(K'_R), $K'_L\oplus$ iiiiiiii$_x$) hoặc là bất kỳ một trong 15 khoá tương đương.

Phép mã hoá khoá K phải được nhận dạng hoặc ở bước 6, bước 7, hoặc bước 8. Từ chỗ phép so sánh trong các bước 6, 7 và 8 là nhanh hơn nhiều phép mã thử, nên độ phức tạp của tấn công này tập trung chủ yếu ở bước 5. Nếu số các phần tử trong mỗi chu trình được chia cho 3, độ phức tạp của tấn công giảm tương đối so với tấn công dựa trên tính chất bù khoảng 1/3. Trong trường hợp của LOKI89, nhân tử này là 3/8, và do đó độ phức tạp của tấn công trên là $(3/8).(2^{28})^2 = 1{,}5.\ 2^{54}$ phép mã thử.

VII. CÁC ĐẶC TRƯNG AN TOÀN CƠ BẢN CỦA MỘT HỆ MÃ KHỐI

Qua các khảo sát trên đây và qua các nghiên cứu của thế giới, chúng ta có thể rút ra các đặc trưng an toàn cơ bản sau đây đối với một hệ mã khối bất kỳ.

- Hệ mã phải có độ dài khối rõ, khối khoá đủ lớn (không gian rõ và khoá lớn) để tránh tấn công vét kiệt trên không gian rõ cũng như không gian khoá (thường độ dài cỡ khối lớn hơn hoặc bằng 128).

- Hệ mã phải có độ đo vi sai và độ đo độ lệch tuyến tính tối thiểu để tránh được hai kiểu tấn công nguy hiểm nhất là tấn công vi sai và tấn công tuyến tính theo các nguyên lý như đã trình bày trên.

- Các hộp thế, các phép biến đổi phi tuyến cần phải có bậc đại số cao tránh tấn công nội suy, tấn công vi sai bậc cao.

- Tầng tuyến tính trong các hàm vòng cần phải được lựa chọn cẩn thận để khi phối hợp với tầng phi tuyến phải tạo ra hệ mã có tính khuyếch tán tốt theo các nguyên lý của chương 1, để tránh các tấn công địa phương trên các khối mã nhỏ.

- Các phép biến đổi đầu vào đầu ra của một hệ mã khối cũng không được quá đơn giản (như DES) mà cần phải là tầng che dấu, ngăn cản việc thiết lập các vi sai hay các mảng đánh dấu tuyến tính các vòng đầu cuối đã biết trước đối với thám mã.

- Lược đồ tạo khoá cần phải tránh được các lớp khoá yếu, và nói chung nên dùng kiểu khoá phiên độc lập (nếu có thể được). Đặc biệt lược đồ khoá không tồn tại những quan hệ khoá đơn giản do tính đều, hay cân xứng trong lược đồ gây nên, nhưng lại phải đảm bảo các khoá là tốt như nhau để tránh các kiểu tấn công khoá quan hệ, tấn công trượt khối dựa trên tính giống nhau trong các phân đoạn tạo khoá con (không phụ thuộc số vòng của hệ mã).

CHƯƠNG 3: KHẢO SÁT HỆ MÃ KHỐI AN TOÀN THEO CÁC ĐẶC TRƯNG ĐỘ ĐO GIẢI TÍCH

Xuất phát từ các tấn công phân tích mã thực tế đã trình bày trong chương 2, đồng thời căn cứ vào nguyên lý thiết kế và các yêu cầu cơ bản đối với một hệ mã khối an toàn, hiệu quả, trong chương này và hai chương tiếp theo sẽ dành cho việc trình bày các nghiên cứu xây dựng các hệ mã khối an toàn chống lại được các tấn công cơ bản đã nêu. Cụ thể là, chương 3 dành cho việc khảo sát hệ mã khối dựa trên các đặc trưng đo được bằng các công cụ giải tích; chương 4 sẽ khảo sát hệ mã khối theo đại số nhóm sinh các hàm mã hoá; và chương 5 sẽ khảo sát hệ mã khối an toàn theo quan điểm xích Markov. Thông qua ba chương này chúng ta sẽ có được cái nhìn tương đối tổng thể về một hệ mã khối an toàn trên các quan điểm của toán học. Khía cạnh an toàn mã khối trên quan điểm các nhà công nghệ-kỹ thuật sẽ không được bàn tới ở đây.

Như chúng ta đã biết mô hình chung phổ biến của một hệ mã khối gồm hai phần: phần ngẫu nhiên hoá dữ liệu và phần lược đồ tạo khoá cho hệ mã. Phần ngẫu nhiên hoá dữ liệu gồm các cấu trúc cơ bản đã giới thiệu trong chương 1, có thể thấy nó thường chứa ba lớp: các hộp thế (lớp trong cùng), hàm vòng (lớp giữa) và cấu trúc mã-dịch (lớp ngoài cùng). Phần lược đồ khoá cũng sẽ được giới thiệu ở cuối chương, nó có thể gồm lược đồ on-line (tính cùng quá trình mã-dịch), hay off-line (tính trước quá trình mã-dịch), hoặc là lược đồ khoá độc lập với phần ngẫu nhiên hoá dữ liệu hay phụ thuộc phần ngẫu nhiên hoá dữ liệu. Để cho hệ mã là an toàn chống được các tấn công đã nêu, cần phải thiết kế xây dựng các hộp thế, hàm vòng và nghiên cứu lựa chọn cấu trúc mã-dịch sao cho hạn chế tối đa các tấn công phân tích mã hoặc vô hiệu hoá các phương pháp thám mã cụ thể. Đồng thời lược đồ khoá phải tránh được các quan hệ khoá đơn giản hoặc tránh các sự tương tự giữa các công đoạn tạo khoá...Muốn vậy chúng ta phải xây dựng và theo dõi được sự ảnh hưởng lẫn nhau giữa các độ đo an toàn của các thành phần cấu tạo nên hệ mã. Vì thế, nội dung chính của Chương 3 sẽ gồm các nghiên cứu khảo sát và xây dựng các thành phần cơ bản sau đây của hệ mã khối:

1. Nghiên cứu về các hộp thế của mã khối
2. Nghiên cứu về các dạng hàm vòng an toàn
3. Nghiên cứu độ an toàn thực tế của cấu trúc mã-dịch kiểu Feistel

4. Nghiên cứu về các lược đồ tạo khoá của mã khối.

Các khảo sát hệ mã khối an toàn trên quan điểm đại số hay xác suất sẽ được trình bày trong hai chương tiếp theo. Một chú ý nữa là, khi xây dựng phân tích một hệ mã khối cụ thể, chúng tôi không hàm ý là phải tách bạch các công cụ toán học đã nêu ở đây.

I. HỘP THẾ TRONG MÃ KHỐI

I.1. MỘT SỐ ĐỘ ĐO PHI TUYẾN CỦA HỘP THẾ

Ký hiệu V_n là không gian các bộ n-phần tử của trường GF(2). Giả sử $\alpha = (a_1, a_2,..., a_n)$ và $\beta = (b_1, b_2,..., b_n)$ là các phần tử của V_n. Ta ký hiệu tích vô hướng của α và β là $<\alpha, \beta> = a_1 b_1 \oplus a_2 b_2 \oplus ... \oplus a_n b_n$, ở đây tổng và tích thực hiện trong trường GF(2).

Giả sử f: $V_n \rightarrow$ GF(2) là một hàm boolean bất kỳ. Khi đó gọi dãy

$$((-1)^{f(\alpha_0)},(-1)^{f(\alpha_1)},....,(-1)^{f(\alpha_{2^n-1})})$$

là dãy của f, và dãy

$$(f(\alpha_0),f(\alpha_1),....,f(\alpha_{2^n-1}))$$

là bảng chân lý của f, ở đây: α_0=(0,..,0,0), α_1=(0,...,0,1),..., α_{2^n-1}= (1,...,1,1).

Định nghĩa 3.1: Hàm f được gọi là cân đối nếu bảng chân lý của nó có đúng 2^{n-1} số không (tức là số bít 0 và bít 1 bằng nhau).

Định nghĩa 3.2: Hàm f trên V_n được gọi là hàm bent nếu n chẵn và:

$$2^{-\frac{n}{2}}\sum_{x\in V_n}(-1)^{f(x)\oplus<\beta,x>} = \pm 1$$

với mọi $\beta \in V_n$. ở đây $x = (x_1,x_2, ..., x_n)$ và $f(x) \oplus <\beta, x>$ xem như là hàm giá trị thực.

Từ định nghĩa trên ta thấy hàm bent không cân đối, nhưng nó có tính chất sau:

Mệnh đề 3.3. Các phát biểu sau là tương đương:

(i) f là hàm bent.

(ii) $<\xi, l> = \pm 2^{n/2}$ với mọi dãy affine bất kỳ l có độ dài 2^n, ở đây ξ là dãy của f.

(iii) $f(x) \oplus f(x \oplus \alpha)$ là cân đối với bất kỳ véc tơ khác không $\alpha \in V_n$, ở đây $x = (x_1,x_2, ..., x_n)$.

Tiêu chuẩn thác chặt (SAC) đã được đề xuất khi nghiên cứu tạo các hộp nén mạnh về mật mã (S-hộp). Ta nhắc lại định nghĩa của khái niệm này.

Định nghĩa 3.4: Hàm f trên V_n được gọi là thoả mãn tiêu chuẩn thác chặt (SAC) nếu $f(x) \oplus f(x \oplus \alpha)$ là cân đối với bất kỳ véc tơ $\alpha \in V_n$ có $W(\alpha) = 1$, ở đây $x = (x_1, x_2, ..., x_n)$.

Có một thực tế chung đã chấp nhận trong mật mã là các S-hộp sử dụng trong mã khối là phải thoả mãn tiêu chuẩn SAC. Hiển nhiên các hàm bent thoả mãn tiêu chuẩn SAC tốt nhất.

Một S-hộp n x s (n-bit đầu vào, s-bit đầu ra) có thể xem là một ánh xạ từ V_n vào V_s ($n \geq s$). Để tránh các tấn công thống kê tầm thường thì S-hộp F phải có ***tính đều***, tức là F(x) phải lấy tất cả các giá trị trong V_s mỗi cái đúng 2^{n-s} lần khi x chạy qua đúng một lần tất cả các véc tơ trong V_n.

Mệnh đề 3.5. Giả sử $F = (f_1, f_2, ..., f_s)$, ở đây f_i là hàm từ V_n vào GF(2), ($n \geq s$). Khi đó F là ánh xạ đều nếu và chỉ nếu mọi tổ hợp tuyến tính khác không của
$f_1, f_2, ..., f_s$ là cân đối.

Định nghĩa 3.6: Cho f và g là hai hàm trên V_n, khoảng cách Hamming giữa f và g, ký hiệu là d(f, g) là trọng số hamming của bảng chân lý của hàm $f(x) \oplus g(x)$, ở đây $x = (x_1, x_2, ..., x_n)$. Độ phi tuyến của hàm f , ký hiện là N_f là số bé nhất của khoảng cách Hamming giữa f với tất cả các hàm affine trên V_n, tức là

$$N_f = min_{i=1,2,...,2^{n+1}} d(f, \varphi_i),$$

ở đây $\varphi_1, \varphi_2, ..., \varphi_{2^{n+1}}$ ký hiệu là các hàm affine trên V_n.

Với ánh xạ F từ V_n vào V_s ($s > 1$) ta có định nghĩa sau về độ phi tuyến

Định nghĩa 3.7: Cho ánh xạ $F = (f_1, f_2, ..., f_s) : V_n \rightarrow V_s$, ở đây $f_1, f_2, ..., f_s$ là các hàm boolean trên V_n. Khi đó độ phi tuyến của F được xác định bởi công thức sau đây:

$$N_F = min \quad N_{L(f_1, f_2, ..., f_s) \setminus \{0\}},$$

ở đây $L(f_1, f_2, ..., f_s) \setminus \{0\}$ là tập tất cả các tổ hợp tuyến tính khác không của $f_1, f_2, ..., f_s$.

Mệnh đề 3.8. Với hàm boolean bất kỳ trên V_n ta có ước lượng sau về độ phi tuyến: $N_f \leq 2^{n-1} - 2^{(n/2)-1}$. Với n chẵn, dấu bằng xảy ra khi f là hàm bent.

Định nghĩa 3.9 (K. Nyberg): Hàm F: $V_n \rightarrow V_m$, được gọi là hàm bent nếu và chỉ nếu với mọi $c \in V_n$, hàm boolean $x \rightarrow c.F(x)$ là hàm bent.
Chú ý rằng hàm bent véc tơ cũng co độ phi tuyến cực đại so với các hàm cùng số biến.

Độ phi tuyến của ánh xạ liên quan chặt chẽ với tấn công tuyến tính. Một S-hộp sẽ là miễn dịch đối với tấn công tuyến tính nếu độ phi tuyến của mỗi tổ hợp tuyến tính khác không của các hàm thành phần của nó là cao, tức là không được lệch quá xa so với số 2^{n-1}.

Khái niệm độ đo vi sai đều liên quan đến tính miễn dịch đối với tấn công vi sai.

Đối với một S-hộp n x s bất kỳ, bảng phân bố vi sai của nó là một ma trận 2^n x 2^s. Mỗi phần tử (α, β) của bảng cho ta số các véc tơ đầu vào khi được thay đổi một lượng là α (theo phép XOR), thì kết quả đầu ra sẽ thay đổi một lượng là β (cũng theo phép XOR).

Định nghĩa 3.10: Giả sử F là một S-hộp (n x s). Ký hiệu δ là giá trị lớn nhất trong bảng phân bố vi saicủa S-hộp (không kể dòng đầu tiên, ứng với XOR đầu vào bằng 0), tức là:

$$\delta = \max_{\alpha \in V_n, \alpha \neq 0} \max_{\beta \in V_s} \left| \{ x : F(x) \oplus F(x \oplus \alpha) = \beta \} \right|.$$

Khi đó F được gọi là δ-vi sai đều, và δ- được gọi là độ đều vi sai của F.

Với một S-hộp (n x s) bất kỳ ta luôn có $2^{n-s} \leq \delta \leq 2^n$.

Mệnh đề 3.11. Độ đo vi sai đều của một S-hộp đều là lớn hơn 2^{n-s}.

Dưới đây là các tính chất bất biến qua phép biến đổi không suy biến.

Mệnh đề 3.12 . Giả sử f là hàm trên V_n, A là ma trận không suy biến cấp n trên GF(2), và giả sử g(x) = f(x.A). Khi đó f và g có cùng bậc đại số, độ phi tuyến và số chiều tuyến tính.

Mệnh đề 3.13. Giả sử F là ánh xạ từ V_n vào V_s, ở đây $n \geq s$, A là ma trận không suy biến cấp n trên GF(2), B là là ma trận không suy biến cấp n trên GF(2). giả sử G(x) = F(x.A) và H(x) = F(x).B. Khi đó F, G và H có cùng tính đều và độ đo vi sai đều.

Định nghĩa 3.14: Giả sử f :$K^p \to R$, là một hàm tuỳ ý, chúng ta ký hiệu f* là biến đổi Hadamard-Walsh của hàm f, được xác định bởi công thức:

$$f^*(w) = \sum_{x \in K^p} f(x).(-1)^{x.w}, \forall w \in K^p ,$$

ở đây x.w là tích vô hướng trên trường K và giá trị tổng được tính trên trường số thực.

Bây giờ giả sử F: $K^p \to K^q$ là hàm mà chúng ta muốn phân tích thám mã. Đối với tấn công vi sai chúng ta cần có các tập hợp dạng sau:

$$D_F(a,b) = \{x \in K^p / F(x \oplus a) \oplus F(x) = b\},$$

ở đây $a \in K^p \backslash \{0\}$, và $b \in K^q$. Độ hiệu quả của thám mã vi sai dựa trên lực lượng của tập $D_F(a, b)$ ký hiệu là

$$\delta_F(a, b) = \# D_F(a, b).$$

Tương tự đối với tấn công tuyến tính chúng ta cần các tập có dạng

$$L_F(a,b) = \{x \in K^p / a.x \oplus b.F(x) = 0\},$$

ở đây $a \in K^p \backslash \{0\}$, và $b \in K^q$, sao cho $\# L_F(a, b) \neq (1/2) | K^p |$. Độ hiệu quả của thám mã tuyến tính phụ thuộc vào khoảng cách giữa lực lượng của tập $L_F(a, b)$ và lực lượng trung bình: $\lambda_F(a, b) = \# L_F(a, b) - (1/2) | K^p |$. Từ đó độ miễn dịch của hàm F có thể được đo bởi các thông số $\Delta_F = \sup_{a \neq 0, b} \delta_F(a,b)$ đối với tấn công vi sai, và $\Lambda_F = \sup_{b \neq 0, a} |\lambda_F(a,b)|$ đối với tấn công tuyến tính. Nếu các giá trị này càng nhỏ thì khả năng chống lại các tấn công trên càng cao. Chú ý rằng nếu $\Delta_F = \delta$, thì hàm F được gọi là δ-vi sai đều.

Định nghĩa 3.15: Cho trước tập $\mathcal{F}$ của các hàm, chúng ta sẽ nói rằng một hàm $F \in \mathcal{F}$, là miễn dịch đối với thám mã vi sai nếu Δ_F là đạt cực tiểu. Cũng vậy, ta gọi hàm F là miễn dịch đối với thám mã tuyến tính nếu như Λ_F đạt cực tiểu.

**** Với lớp hàm bent véc tơ ta có các kết quả sau.***

Định lý 3.16. Đối với ánh xạ F bất kỳ ta luôn có: $\Delta_F \geq 2^{p-q}$. Từ đó một hàm F được gọi là hàm phi tuyến hoàn thiện nếu và chỉ nếu $\Delta_F = 2^{p-q}$.

Định lý 3.17. (K. Nyberg):

(i) Một hàm là phi tuyến hoàn thiện khi và chỉ khi nó là hàm bent.

(ii) Hàm véc tơ bent chỉ tồn tại khi và chỉ khi $p \geq q$ và p chẵn.

Trong [35] cũng chỉ ra rằng Λ_F đạt cực tiểu với hàm bent F. Do vậy, khi p là chẵn và lớn hơn hoặc bằng 2.q, tính miễn dịch đối với tấn công vi sai là tương đương với miễn dịch tấn công tuyến tính và đạt được đối với lớp hàm bent.

*** Hàm Phi tuyến gần hoàn thiện, hàm gần bent**

Định nghĩa 3.18: Chúng ta có $\Delta_F \geq 2$. Nếu hàm F có $\Delta_F = 2$, thì ta gọi nó là hàm phi tuyến gần hoàn thiện (Almost Perfect Nonlinear).

Nhận xét: Do $\Delta_F \geq 2^{p-q}$, nên hàm phi tuyến gần hoàn thiện chỉ tồn tại khi $q \geq p$.

Trong [35] đã chứng minh cận sau đây của Λ_F.

Định lý 3.19. Với mọi ánh xạ F ta có:

$$\Lambda_F \geq \frac{1}{2}\left(3.2^p - 2 - 2.\frac{(2^p-1)(2^{p-1}-1)}{2^q-1}\right)^{1/2}.$$

Khi dấu bằng xảy ra ta nói hàm đó là hàm gần bent. Ngoài ra, một hàm gần bent luôn là hàm phi tuyến gần hoàn thiện.

Định lý 3.20. Nếu F: $K^p \rightarrow K^q$ là hàm gần Bent và không phải là hàm Bent, khi đó p = q, và p phải là số lẻ. Cận trên đây sẽ trở thành

$$\Lambda_F = \frac{1}{2}2^{\frac{p+1}{2}}. \qquad (3.1)$$

Ví dụ 1: (K. Nyberg) Giả sử $F(x) = x^{2^k+1}$ là đa thức luỹ thừa trên trường $GF(2^n)$. Nếu n lẻ, $1 < k < n$ và $GCD(n, k) = 1$, thì F là một hoán vị gần bent. Điều này sẽ được chứng minh trực tiếp ở phần sau.

**** Tóm lại các kết quả đã thu được là***

- Khi p>=2q, và p chẵn, tính miễn dịch tấn công vi sai là tương đương với miễn dịch tấn công tuyến tính và đạt được đối với các hàm bent véc tơ. Trong trường hợp này ta có

$$\Lambda_F = (1/2)2^{p/2} \text{ và } \Delta_F = 2^{p-q}.$$

- Khi p = q và p lẻ, tính miễn dịch tấn công vi sai là tương đương với tính phi tuyến gần hoàn thiện (ở đây $\Delta_F = 2$), tính miễn dịch tấn công tuyến tính là tương đương với tính gần bent (ở đây $\Lambda_F = (1/2)2^{(p+1)/2}$) và tính miễn dịch tấn công tuyến tính suy ra tính miễn dịch tấn công vi sai.

I.2. KHẢO SÁT MỘT SỐ LỚP HÀM CỤ THỂ

Trước hết ta nhắc lại định nghĩa về độ đo vi sai đều để phát biểu một kết quả liên quan đến mô hình mã tựa DES (DES-like).

Định nghĩa 3.21: Giả sử G_1 và G_2 là các nhóm Abelian hữu hạn. Một ánh xạ F: $G_1 \to G_2$ được gọi là δ -vi sai đều, nếu với mọi $\alpha \in G_1$, $\alpha \neq 0$ và $\beta \in G_2$ ta có: $\#\{z \in G_1 \mid F(z + \alpha) - F(z) = \beta\} \leq \delta$.

Đối với hệ mã DES-like (tựa DES) ta có kết quả sau [25].

Định lý 3.22. Nếu các hàm vòng của hệ mã DES -like trên nhóm Abelian G là δ-vi sai đều và các khoá vòng là độc lập ngẫu nhiên đều thì với mỗi cặp đầu vào cho trước $(x+\alpha, x)$, $\alpha \neq 0$, xác suất trung bình trên các khoá để đạt được một vi sai đầu ra $\beta \neq 0$ tại vòng thứ s, với $s \geq 4$ sẽ nhỏ hơn hoặc bằng $2(\delta/|G|)^2$.
Nếu tất cả các hàm vòng là như nhau và là một phép hoán vị thì khẳng điịnh còn đúng với $s \geq 3$.

Bây giờ ta sẽ khảo sát một số lớp hàm cụ thể có các tính chất mật mã tốt. Trước hết ta nhắc lại hai kết quả sau để sử dụng

Mệnh đề 3.2. Giả sử A: $G_1 \to G_1$ và B: $G_2 \to G_2$ là các phép đồng cấu nhóm và F: $G_1 \to G_2$ là ánh xạ δ-vi sai đều. Khi đó $B \circ F \circ A$ cùng là ánh xạ δ-vi sai đều.

Mệnh đề 3.24. Giả sử F: $G_1 \to G_2$ là một song ánh δ-vi sai đều. Khi đó ánh xạ nghich đảo của F cũng là δ-vi sai đều.

I..2.1. CÁC ĐA THỨC LUỸ THỪA $F(x) = x^{2^k} + 1$ TRONG $GF(2^n)$.

Định lý 3.25. Giả sử $F(x) = x^{2^k} + 1$ là một đa thức luỹ thừa trong $GF(2^n)$ và giả sử $s = \gcd(k, n)$. Khi đó F là ánh xạ 2^s - vi sai đều. Nếu n/s là số lẻ, tức F là một hoán vị, thì khoảng cách Hamming của hàm Boolean $f_\omega(x) = tr(\omega F(x))$ đối với tập các hàm Boolean tuyến tính sẽ bằng $2^{n-1} - 2^{(n+s)/2-1}$ với mọi $\omega \in GF(2^n)$, $\omega \neq 0$.
Chứng minh: Cho trước $\alpha, \beta \in GF(2^n)$, $\alpha \neq 0$, phương trình sau đây

$$F(x+\alpha) + F(x) = (x+\alpha)^{2^k+1} + x^{2^k+1} = \beta \qquad (3.2)$$

hoặc không có nghiệm hoặc có ít nhất hai nghiệm (do trường đặc số 2). Gọi x_1 và x_2 là hai nghiệm khác nhau. Khi đó ta có:

$$(x_1 + x_2)^{2^k} \alpha + (x_1 + x_2)\alpha^{2^k} = 0$$

hoặc tương đương

$$(x_1 + x_2)^{2^k - 1} = \alpha^{2^k - 1}$$

Từ đó suy ra rằng

$$x_1 + x_2 \in \alpha(G \setminus \{0\})$$

ở đây G là trường con của $GF(2^n)$ với bậc là 2^s.
Từ đó cho trước một lời giải x_0 của (3.2) thì sẽ thiết lập được tập tất cả các lời giải tương ứng là $x_0 + \alpha G$ và có lực lượng tương ứng là 2^s.

Bây giờ giả sử $\omega \in GF(2^n)$, $\omega \neq 0$ và ký hiệu F_ω^* là biến đổi Walsh của f_ω. Khi đó để chứng minh ý thứ hai ta chỉ cần chứng minh rằng

$$\max_{t \in GF(2^n)} \left| F_\omega^*(t) \right| = 2^{\frac{n+s}{2}}$$

Giả sử $t \in GF(2^n)$. Khi đó

$$(F_\omega^*(t))^2 = \sum_{x \in GF(2^n)} (-1)^{f_\omega(x)+t.x} \sum_{y \in GF(2^n)} (-1)^{f_\omega(x+y)+t.(x+y)}$$

$$= \sum_{y \in GF(2^n)} (-1)^{t.y} \sum_{x \in GF(2^n)} (-1)^{f_\omega(x+y)+f_\omega(x)}$$

Giả sử $y \neq 0$ và ký hiệu E_y là miền ảnh của ánh xạ tuyến tính

$$x \rightarrow F(x+y) + F(x) + F(y) = x^{2^k} y + y^{2^k} x$$

Tương tự như chứng minh của phần đầu ta thấy rằng hạt nhân của ánh xạ tuyến tính này là yG. Như vậy số chiều của không gian tuyến tính E_y sẽ là n-s. Với mỗi $y \neq 0$, hoặc là $tr(\omega\beta) = 0$ với mọi $\beta \in E_y$, hoặc $\sum_{\beta \in E_y} (-1)^{tr(\omega\beta)} = 0$.

Các véc tơ y làm cho $tr(\omega\beta) = 0$ với mọi $\beta \in E_y$, tức là

$$f_\omega(x+y) + f_\omega(x) + f_\omega(y) = tr(\omega(x^{2^k} y + y^{2^k} x)) = 0$$

với mọi $x \in GF(2^n)$, chúng sẽ lập thành một không gian con tuyến tính của $GF(2^n)$. Từ đó để chứng minh ta chỉ cần cho thấy rằng Y có 2^s phần tử.

Giả sử $y \in Y$. Khi đó

$$tr(\omega y x^{2^k}) = tr(\omega y^{2^k} x) = tr(\omega^{2^k} y^{2^k} x^{2^k})$$

với mọi $x \in GF(2^n)$, điều này tương đương với $\omega y = \omega^{2^k} y^{2^k}$ hoặc, nếu $y \neq 0$, $(\omega F(y))^{2^k-1} = 1$, từ đây ta có đúng 2^s -1 lời giải khác không của y, do F được giả thiết là một hoán vị.

Điều này hoàn thành việc chứng minh định lý.

Nhận xét:

Nếu n lẻ, $1 < k < n$, và $gcd(n, k) = 1$, thì đa thức luỹ thừa $F(x) = x^{2^k+1}$ trong $GF(2^n)$ sẽ là một hoán vị 2-vi sai đều.

K. Nyberg [30] cũng đã thiết lập mối quan hệ giữa độ đo thám mã tuyến tính với độ phi tuyến của một ánh xạ $f: K^n \rightarrow K^n$ như sau:

$L(f) := 2.\max_{a,b \neq 0} \left| \Pr(a.X \oplus b.f(X) = 0) - 1/2 \right| = 1 - 2^{1-n}.N(f)$

Sử dụng công thức trên đây, nếu lấy k=1, thì ta có đa thức luỹ thừa $F(x) = x^3$ trên trường $GF(2^n)$ với n-lẻ sẽ đạt cả các cận dưới đối với cả hai chỉ số độ đo tấn công vi sai và tấn công tuyến tính:

$\Delta_F = 2$ và $\Lambda_F = (1/2)2^{(n+1)/2}$

tức nó là hoán vị gần Bent và Phi tuyến gần hoàn thiện sẽ được dùng để thiết lập các hộp thế sau này.

I.2.2. HÀM NGHICH ĐẢO F(x) = x^{-1} TRONG GF(2^n).

Xét ánh xạ F: GF(2^n) $\rightarrow$ GF(2^n) thiết lập bởi công thức

$$F(x) = \begin{cases} x^{-1}, & x \neq 0 \\ 0, & x = 0 \end{cases}. \qquad (3.3)$$

Khi đó ta có định lý sau.

Định lý 3.26. Ánh xạ nghich đảo F xác định bởi (3.3) Là ánh xạ 4-vi sai đều trong nhóm (GF(2^n), +).

Chứng minh:

Cho trước $\alpha, \beta \in GF(2^n)$, $\alpha \neq 0$, xét phương trình sau đây

$$F(x+\alpha) + F(x) = (x+\alpha)^{-1} - x^{-1} = \beta \qquad (3.4)$$

Giả sử rằng $x \neq 0$, và $x \neq -\alpha$. Khi đó (3.4) tương đương với

$$\beta x^2 + \beta\alpha x - \alpha = 0, \qquad (3.5)$$

chúng có nhiều nhất hai nghiệm trong GF(2^n). Nếu hoặc x =0, hoặc x =- α là nghiệm của (3.4) thì cả hai chúng đều là lời giải và $\beta = \alpha^{-1}$. Trong trường hợp đó (3.5) tương đương với

$$x^2 + \alpha x - \alpha^2 = 0, \qquad (3.6)$$

chúng có thể cho 2 hay nhiều hơn lời giải của (3.4).

Bây giờ bình phương (3.6) và thế $x^2 = \alpha x + \alpha^2$ chúng ta nhận được

$$x(x^3 + \alpha^3) = 0,$$

chúng không có lời giả nào khác ngoài x = 0 hoặc α nếu gcd (3, 2^n -1) = 1, hoặc tương đương n -là số lẻ. Nếu n-chẵn thì 3 là ước của 2^n -1. Đặt d = (1/3)(2^n -1), khi đó có 2 hay nhiều hơn hai lời giải có dạng x =α^{1+d} và x = α^{1+2d}. □

*Từ kết quả trên chúng ta liệt kê các tính chất sau của ánh xạ nghịch đảo trong trường GF(2^n).

(i) $N(F) = \min_{\omega \neq 0} \min_{L \text{ linear}} \min_{x \in GF(2^n)} d(tr(\omega x^{-1}, L(x)) \geq 2^{n-1} - 2^{n/2}$;

(ii) $\deg(tr(\omega x^{-1})) = w_2(2^n-2) = n-1$;

(iii) F là 2-vi sai đều nếu n-lẻ, và là 4-vi sai đều nếu n-chẵn;

(iv) Thuật toán Euclid để tính x^{-1} là trong thời gian đa thức theo n.

I.2.3. ÁNH XẠ THIẾT KẾ TỪ ÁNH XẠ MŨ TRONG TRƯỜNG NGUYÊN TỐ

Giả sử p -là số nguyên tố xét nhóm Abelian G = {0, 1, ..., p-1} với phép cộng modulo p. Giả sử u là phần tử bậc q trong trường hữu hạn GF(p). Chúng ta định nghĩa ánh xạ F: G $\rightarrow$ G thiết lập bởi công thức

$$F(x) = u^x \text{ , với } x \in G,$$

ở đây phép mũ là được tính toán trong trường GF(p).

Giả sử α , $\beta \in G$, $\alpha \neq 0$, khi đó phương trình

$$F(x+\alpha) + F(x) = u^{(x+\alpha) \bmod p} - u^x = \beta \qquad (3.7)$$

tương đương với

$$\begin{cases} u^{x+\alpha} - u^x = \beta, \ \ and \ \ 0 \le x \le p-\alpha-1 & (3.8) \\ or & \\ u^{x+\alpha-p} - u^x = \beta, \ \ and \ \ p-\alpha \le x \le p-1. & (3.9) \end{cases}$$

Từ chỗ lời giải của x theo modulo p là duy nhất nên từ (3.8) suy ra rằng có nhiều nhất $\left\lceil \frac{p-\alpha}{q} \right\rceil$ lời giải trong G. Tương tự phương trình (3.9) cũng có nhiều nhất là $\left\lceil \frac{\alpha}{q} \right\rceil$ lời giải trong G. Hệ quả là phương trình (3.7) cũng có nhiều nhất là $\left\lceil \frac{p-\alpha}{q} \right\rceil + \left\lceil \frac{\alpha}{q} \right\rceil = \frac{p-1}{q} + 1$ lời giải trong G.

Từ đó ta có khẳng định sau.

Mệnh đề 3.27. Giả sử F là ánh xạ từ tập các số nguyên modulo một số nguyên tố p vào chính nó như đã định nghĩa trên bằng cách sử dụng phép mũ hoá và một phần tử bậc q trong trương GF(p). Khi đó F là $(\frac{p-1}{q}+1)$-vi sai đều.

Chú ý: ánh xạ F định nghĩa trong phần này có vẻ đủ phức tạp để sử dụng như một hàm vòng của DES-like trên các số nguyên modulo một số nguyên tố với một số nhỏ các vòng lặp. Độ phức tạp tính toán của các mã khối như thế tăng theo bậc của phần tử cơ sở *u*. Mệnh đề trên cho thấy một trade-off giữa độ phức tạp của thuật toán mã hoá (và giải mã) và độ an toàn chống lại tấn công vi sai.

II. HÀM VÒNG TRONG CÁC MÃ KHỐI LẶP

Phần này sẽ tập trung trình bày về các kiểu hàm vòng tổ hợp trên các hộp thế và các hàm thế phụ thuộc khoá tạo ra các vòng lặp có độ đo vi sai-tuyến tính an toàn chứng minh được.

II.1. CÁC ĐỘ ĐO AN TOÀN CỦA HÀM VÒNG PHỤ THUỘC KHOÁ

Một trong những nguyên nhân dẫn đến các hộp nén của DES còn tồn tại những yếu điểm đó là do chúng là các "hộp nén", tức là do không gian đầu vào không bằng không gian đầu ra, do đó khó có thể tìm được các hộp nén vừa có phân bố vi sai đều vừa có độ lệch tuyến tính tối thiểu. Do đó, ở đây chúng ta chỉ làm việc với các hộp thế cố định với cỡ đầu vào và đầu ra bằng nhau (bằng n).

Định nghĩa 3.28:

Cho ánh xạ $S : X \to Y$, trong đó $X \equiv Y \equiv \{0, 1\}^n$, $n \in N$. Cho trước $\Delta x, \Gamma x \in X$ và $\Delta y, \Gamma y \in Y$.Ta định nghĩa:

$DP^S(\Delta x \to \Delta y) = (1/2^n).\ \#\{x \in X: S(x) \oplus S(x \oplus \Delta x) = \Delta y\};$ (3.10)

$$LP^S(\Gamma x \to \Gamma y) = \left(2\frac{\#\{x \in X : x \bullet \Gamma x = S(x) \bullet \Gamma y}{2^n} - 1\right)^2 \qquad (3.11)$$

ở đây, ký hiệu $a \bullet b$ có nghĩa là $\oplus^n_{i=1}(a_i.b_i)$, trong đó $a = (a_1, a_2,.., a_n)$ và $b = (b_1, b_2,.., b_n)$. Chú ý rằng các số DP^S và LP^S nằm trong khoảng [0,1].

Trên cơ sở các tấn công vi sai và tuyến tính đối với DES ta thấy rằng một hộp thế mạnh phải là hộp có các số DP^S và LP^S đủ nhỏ với mọi $\Delta x(\neq 0), \Gamma x \in X$ và $\Delta y, \Gamma y(\neq 0) \in Y$, nên các tham số sau đây có thể được xem là các yếu tố miễn dịch của một hộp thế S chống lại các tấn công vi sai và tấn công tuyến tính trên chúng.

Định nghĩa 3.29: Với các qui ước đã nêu, ta đặt:

$$DP^S_{max} = \max_{\Delta x \neq 0,\ \Delta y} DP^S(\Delta x \to \Delta y) \qquad (3.12)$$

$$LP^S_{max} = \max_{\Gamma x,\ \Gamma y \neq 0} LP^S(\Gamma x \to \Gamma y) \qquad (3.13)$$

Bổ đề 3.30.
Với ánh xạ S bất kỳ, ta luôn có:

$$\sum_{\Delta y \in Y} DP^S(\Delta x \to \Delta y) = 1, \qquad \sum_{\Gamma x \in X} LP^S(\Gamma x \to \Gamma y) = 1. \qquad (3.14)$$

Ngoài ra, nếu S là một phép song ánh thì ta có thêm các hệ thức:

$$\sum_{\Delta x \in X} DP^S(\Delta x \to \Delta y) = 1, \qquad \sum_{\Gamma y \in Y} LP^S(\Gamma x \to \Gamma y) = 1. \qquad (3.15)$$

Tiếp theo chúng ta sẽ xét về độ đo độ an toàn của các hàm phụ thuộc khoá như trong hình 3.2.

Giả sử K là tập tất cả các giá trị khoá có thể. Chúng ta sẽ định nghĩa độ mạnh của hàm F phụ thuộc không gian khoá K như là độ mạnh trung bình của F< k > khi k chạy trên toàn bộ không gian K, ở đây F< k > ký hiệu là hàm một biến ứng với khoá k cố định. Cụ thể, ta có các định nghĩa sau.

Định nghĩa 3.31:

$$DP^F(\Delta x \to \Delta y) = \Sigma'_{k \in K} DP^{F<k>}(\Delta x \to \Delta y) \qquad (3.16)$$

$$LP^F(\Gamma x \to \Gamma y) = \Sigma'_{k \in K} LP^{F<k>}(\Gamma x \to \Gamma y) \qquad (3.17)$$

Trong thực tế, khi F là một phép biến đổi mã hoá, và các giá trị DP^F , LP^F là đủ nhỏ với bất kỳ $\Delta x(\neq 0)$, $\Gamma x \in X$ và $\Delta y, \Gamma y(\neq 0) \in Y$, chúng ta nói rằng F có độ an toàn chứng minh được chống lại các tấn công vi sai và tấn công tuyến tính. Một cách tương đương đương, F được gọi là có độ an toàn chứng minh được, nếu các giá trị sau đây là đủ nhỏ.

Định nghĩa 3.32:

$$DP^F_{max} = \max_{\Delta x \neq 0,\, \Delta y} DP^F(\Delta x \to \Delta y) \qquad (3.18)$$

$$LP^F_{max} = \max_{\Gamma x, \Gamma y \neq 0} LP^F(\Gamma x \to \Gamma y) \qquad (3.19)$$

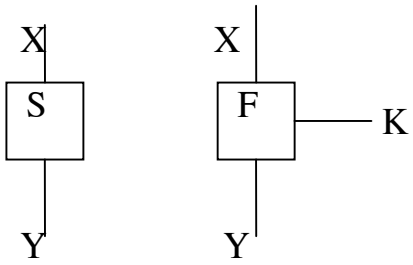
Đến đây, chúng ta thấy rằng vai trò và các yêu cầu hàm vòng được lựa chọn sử dụng trong mật mã là hết sức quan trọng. Ta có thể xem hàm phụ thuộc khoá như là một tập các phép biến đổi đã được cài đặt trong hệ thống mã dịch, và yêu cầu đặt ra là mỗi một phần tử của tập hợp này đều phải có các độ đo vi sai và độ đo tuyến tính đủ nhỏ; hoặc trong quá trình lựa chọn, ta cần phải loại bỏ được những khoá làm cho các độ đo của hàm

vòng tương ứng lớn, vì khi đó mới đảm bảo được tính an toàn chứng minh được của hàm mã hoá đang sử dụng. Sau đây chúng ta sẽ xét hai mô hình cơ bản của các hàm mã hoá phụ thuộc khoá, như đã chỉ ra trong các Hình 3.3 và 3. 4.

Định lý 3.33. [25]**.** Đối với hàm F cho trong Hình 3.3 ta có các đẳng thức sau:

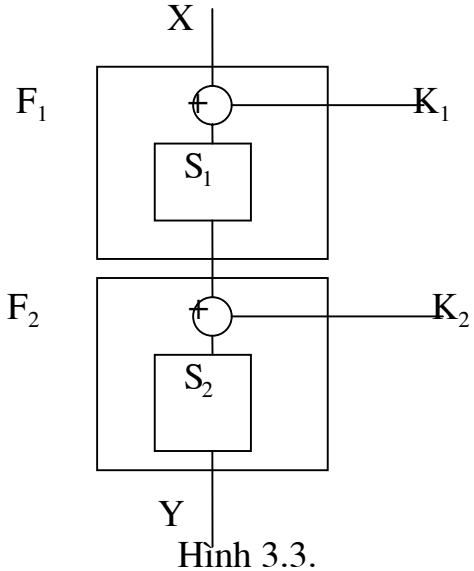
$$DP^F(\Delta x \to \Delta z) = \sum_{\Delta y \in Y} DP^{S1}(\Delta x \to \Delta y).\, DP^{S2}(\Delta y \to \Delta z) \qquad (3.20)$$

$$LP^F(\Gamma x \to \Gamma z) = \sum_{\Gamma y \in Y} LP^{S1}(\Gamma x \to \Gamma y).\, LP^{S2}(\Gamma y \to \Gamma z) \qquad (3.21)$$

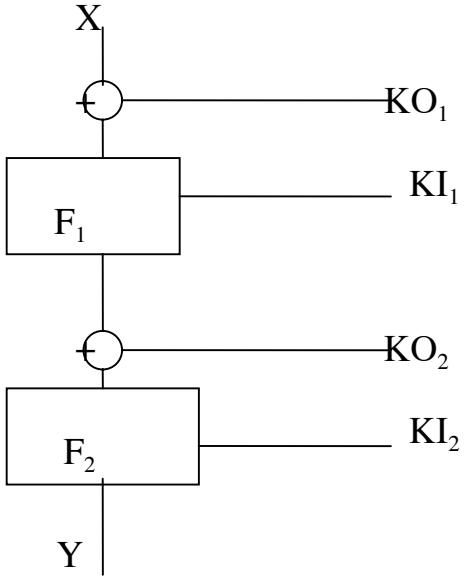


Hình 3.1.

Hình 3.2.



Hình 3.3.



Hình 3. 4.

Chứng minh:

Trước hết ta thấy khoá $k_1 \in K_1$ không ảnh hưởng tới kết luận của định lý, nên ta có

$$DP^F(\Delta x \to \Delta z) =$$

$$= \sum_{k_2 \in K_2}' \sum_{\Delta y \in Y} DP^{S1}(\Delta x \to \Delta y).\ D\tilde{P}^{F_2<k_2>}(\Delta y \to \Delta z \mid \Delta x \to \Delta y)$$

$$= \sum_{\Delta y \in Y} DP^{S1}(\Delta x \to \Delta y) \sum_{k_2 \in K_2}' D\tilde{P}^{F_2<k_2>}(\Delta y \to \Delta z \mid \Delta x \to \Delta y) \qquad (3.22)$$

ở đây số hạng cuối cùng biểu diễn xác suất có điều kiện sao cho Δy dẫn tới Δz khi Δx gây ra Δy. Nói cách khác, tồn tại một tập con $\tilde{Y}$ của không gian Y để ta có

$$D\tilde{P}^{F_2<k_2>}(\Delta y \to \Delta z \mid \Delta x \to \Delta y)$$

$$= \#\{\tilde{y} \in \tilde{Y} \mid F_2<k_2>(\tilde{y}) \oplus F_2<k_2>(\tilde{y} \oplus \Delta y) = \Delta z\}/\#\tilde{Y}.$$

Do đó tổng thứ hai của (3.22) sẽ là

$$\sum_{k_2 \in K_2}' D\tilde{P}^{F_2<k_2>}(\Delta y \to \Delta z \mid \Delta x \to \Delta y)$$

$$= \sum_{k_2 \in K_2}' \#\{\tilde{y} \in \tilde{Y} \mid F_2<k_2>(\tilde{y}) \oplus F_2<k_2>(\tilde{y} \oplus \Delta y) = \Delta z\}/\#\tilde{Y}.$$

$$= \sum_{k_2 \in K_2}' \#\{\tilde{y} \in \tilde{Y} \mid S_2(\tilde{y} \oplus k_2) \oplus S_2(\tilde{y} \oplus k_2 \oplus \Delta y) = \Delta z\}/\#\tilde{Y}.$$

$$= \sum_{\tilde{y} \in \tilde{Y}}' \#\{k_2 \in K_2 \mid S_2(\tilde{y} \oplus k_2) \oplus S_2(\tilde{y} \oplus k_2 \oplus \Delta y) = \Delta z\}/\#\tilde{K}_2$$

$$= \sum_{\tilde{y} \in \tilde{Y}}' DP^{S2}(\Delta y \to \Delta z)$$

$$= DP^{S2}(\Delta y \to \Delta z).$$

Tóm lại ta có

$$DP^F(\Delta x \to \Delta z) = \sum_{\Delta y \in Y} DP^{S1}(\Delta x \to \Delta y).\ DP^{S2}(\Delta y \to \Delta z).$$

tức là (3.20) được chứng minh.

Bây giờ ta chứng minh (3.21). Xuất phát từ định nghĩa và áp dụng Bổ đề 3.30 ta có

$$\sum_{\Gamma y \in Y} LP^{S1}(\Gamma x \to \Gamma y).\ LP^{S2}(\Gamma y \to \Gamma z)$$

$$= \sum_{\Gamma y \in Y} \sum_{x \in X}' \sum_{x' \in X}' \sum_{y \in Y}' \sum_{y' \in Y}' (-1)^{x \bullet \Gamma x \oplus S_1(x) \bullet \Gamma y \oplus x' \bullet \Gamma x \oplus S_1(x') \bullet \Gamma y \oplus y \bullet \Gamma y \oplus S_2(y) \bullet \Gamma z \oplus y' \bullet \Gamma y \oplus S_2(y') \bullet \Gamma z}$$

$$= \sum_{x\in X}' \sum_{x'\in X}' \sum_{y\in Y}' \sum_{y'\in Y}' (-1)^{x\bullet\Gamma x\oplus x'\bullet\Gamma x\oplus S_2(y)\bullet\Gamma z\oplus S_2(y')\bullet\Gamma z} \text{ x}$$

$$\sum_{\Gamma y\in Y} (-1)^{(S_1(x)\oplus S_1(x')\oplus y\oplus y')\bullet\Gamma y}$$

Trong biểu thức trên ta thấy tổng sau cùng chỉ khác không khi và chỉ khi $S_1(x) \oplus S_1(x') \oplus y \oplus y' = 0$. Từ đó, ta rút ra $y' = S_1(x) \oplus S_1(x') \oplus y$ và thay y bởi k_2 (vì vai trò của y và k_2 là như nhau do đặc tính của phép XOR), ta sẽ tính được

$$\sum_{\Gamma y\in Y} LP^{S1}(\Gamma x \to \Gamma y).\, LP^{S2}(\Gamma y \to \Gamma z)$$

$$= \sum_{x\in X}' \sum_{x'\in X}' \sum_{k_2\in K_2}' (-1)^{x\bullet\Gamma x\oplus x'\bullet\Gamma x\oplus S_2(k_2)\bullet\Gamma z\oplus S_2(k_2\oplus S_1(x)\oplus S_1(x'))\bullet\Gamma z}$$

$$= \sum_{x\in X}' \sum_{x'\in X}' \sum_{k_2\in K_2}' (-1)^{x\bullet\Gamma x\oplus x'\bullet\Gamma x\oplus S_2(k_2\oplus S_1(x))\bullet\Gamma z\oplus S_2(k_2\oplus S_1(x'))\bullet\Gamma z}$$

(ở đây ta đã thay k_2 bởi $k_2 \oplus S_1(x)$, nên biểu thức $k_2 \oplus S_1(x) \oplus S_1(x')$ trên mũ đã biến thành $k_2 \oplus S_1(x')$)

$$= \sum_{k_2\in K_2}' LP^{F<k_2>}(\Gamma x \to \Gamma z) \qquad \text{(do Bổ đề 3.30 và định nghĩa)}$$

$$= LP^F(\Gamma x \to \Gamma z)$$

Vậy đẳng thức (3.21) được chứng minh. Tóm lại định lý được chứng minh hoàn toàn. (ĐPCM)

Chú ý rằng định lý trên đây giả thiết S_1 và S_2 là các hộp thế cố định. Khi chúng là các hàm phụ thuộc khoá thì ta cũng có kết quả sau đây.

Định lý 3.34.
Đối với hàm mã hoá cho trên Hình 3.4, ta sẽ có các đẳng thức sau.

$$DP^F(\Delta x \to \Delta z) = \sum_{\Delta y\in Y} DP^{F_1}(\Delta x \to \Delta y).\, DP^{F_2}(\Delta y \to \Delta z) \quad (3.23)$$

$$LP^F(\Gamma x \to \Gamma z) = \sum_{\Gamma y\in Y} LP^{F_1}(\Gamma x \to \Gamma y).\, LP^{F_2}(\Gamma y \to \Gamma z) \quad (3.24)$$

Hệ quả 3.35.

Đối với hàm F cho trong Hình 3.4, ta sẽ có các đánh giá sau về độ đo vi sai và độ đo tuyến tính của chúng thoả mãn:

$$DP^F_{max} \le DP^{F_1}_{max} + DP^{F_2}_{max}\ , \quad LP^F_{max} \le LP^{F_1}_{max} + LP^{F_2}_{max} \quad (3.25)$$

Ngoài ra, nếu F là song ánh với bất kỳ giá trị khoá nào, thì ta sẽ có đánh giá:

$$DP^{F}_{max} \le \min\{ DP^{F_1}_{max}, DP^{F_2}_{max} \},\ LP^{F}_{max} \le \min\{ LP^{F_1}_{max}, LP^{F_2}_{max} \} \qquad (3.26)$$

II.2. MỘT SỐ DẠNG HÀM VÒNG AN TOÀN-CHỨNG MINH ĐƯỢC

Bây giờ vận dụng các kết quả cơ bản trên đây, trước hết chúng ta sẽ khảo sát hai kiểu phối ghép các hộp thế, hàm thế phụ thuộc khóa trong *một dạng* mô hình mã dịch được cho trong Hình 3.5 và Hình 3.6. Ta có định lý:

Định lý 3.36. (Matsui, 1996 [25]) Đối với hàm n-vòng F được cho trong Hình 3.5, giả thiết $n \ge 3$, và giả sử rằng mỗi một hộp S_i đều là song ánh có các độ đo vi sai (hoặc độ đo tuyến tính) thoả mãn $DP^{S_i}_{max} \le p$ (hoặc tương ứng $LP^{S_i}_{max} \le p$). Khi đó ta sẽ có các độ đo của hàm F sẽ được đánh giá bởi bất đẳng thức:

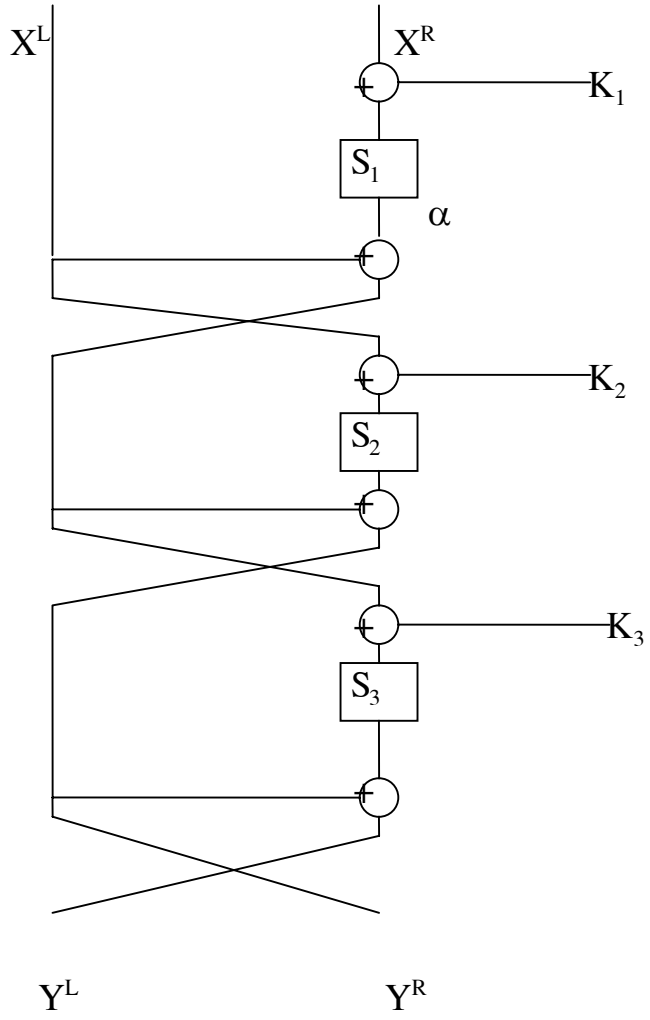
$$DP^{F}_{max} \le p^2 \text{ (hoặc tương ứng } LP^{F}_{max} \le p^2) \qquad (3.27)$$

Chứng minh:

Trước hết ta chứng minh cho n = 3, còn đối với n > 3 sẽ được suy trực tiếp từ Hệ quả 3.35. Ta xét bốn trường hợp sau đây:

Trường hợp 1: $\Delta x^R = 0$, khi đó $\Delta x^L \ne 0$. Từ đó theo dõi sơ đồ mã hoá ta có:

$$DP^{F}(\Delta x \rightarrow \Delta y) =$$
$$= DP^{S_2}(\Delta x^L \rightarrow \Delta x^L \oplus \Delta y^R)\ DP^{S_3}(\Delta x^L \rightarrow \Delta y^L \oplus \Delta y^R)$$
$$\le DP^{S_2}_{max}.DP^{S_3}_{max} = p^2 \text{ (theo định nghĩa).}$$



Hình 3.5.

Trường hợp 2: $\Delta x^L = 0$, khi đó $\Delta x^R \neq 0$. Khi đó vi sai đầu ra của S^2 là bằng 0 và vi sai đầu vào của S^3 phải bằng Δy^R; từ đó $\Delta y^R \neq 0$ và ta có:

$$DP^F(\Delta x \rightarrow \Delta y) =$$
$$= DP^{S_1}(\Delta x^R \rightarrow \Delta y^R)\ DP^{S_3}(\Delta y^R \rightarrow \Delta y^L \oplus \Delta y^R)$$
$$\leq DP^{S_1}_{max}.DP^{S_3}_{max} = p^2.$$

Trường hợp 3: $\Delta y^L \oplus \Delta y^R = 0$, khi đó vi sai đầu vào của S_3 bằng không và vi sai đầu ra của S_1 phải bằng Δx^L. Từ đó $\Delta x^L \neq 0$ và $\Delta x^R \neq 0$, và do vậy ta có:

$$DP^F(\Delta x \rightarrow \Delta y) =$$
$$= DP^{S_1}(\Delta x^R \rightarrow \Delta x^L)\ DP^{S_2}(\Delta x^L \rightarrow \Delta y^R)$$
$$\leq DP^{S_1}_{max}.DP^{S_2}_{max} = p^2.$$

Trường hợp 4: Xét các khả năng còn lại của Δx, Δy. Chúng ta giả sử rằng vi sai đầu ra của S_1 là Δα, nó không được xác định duy nhất. Bằng cách chứng minh tương tự như trong định lý trên, ta nhận được:

$$DP^F(\Delta x \to \Delta y) =$$

$$\sum_{\Delta\alpha} DP^{S_1}(\Delta x^R \to \Delta\alpha) DP^{S_2}(\Delta x^L \to \Delta x^L \oplus \Delta y^R \oplus \Delta\alpha) \times$$

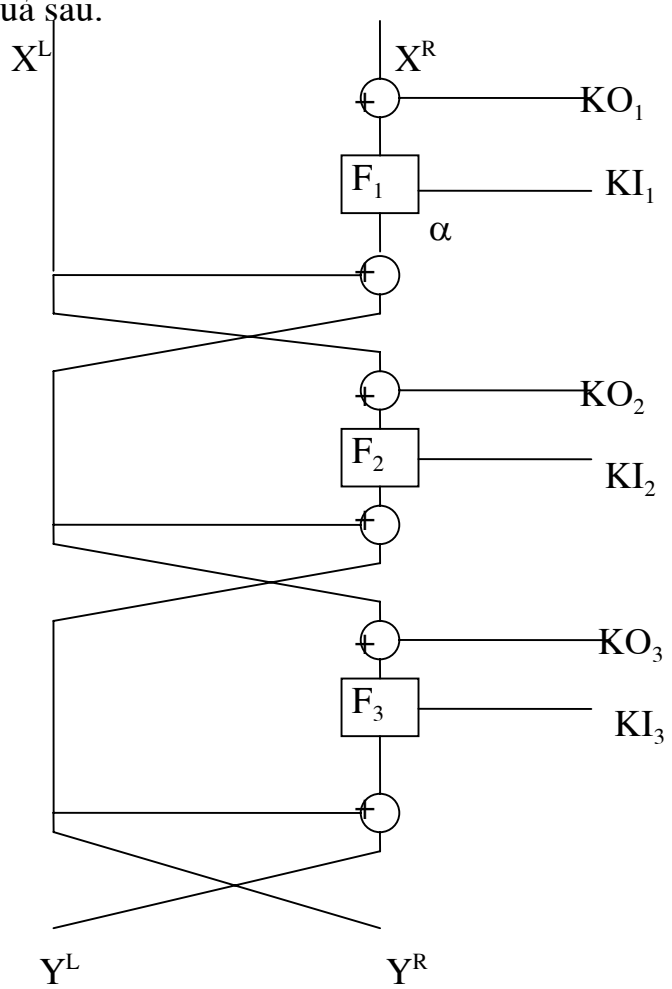
$$DP^{S_3}(\Delta x^L \oplus \Delta\alpha \to \Delta y^L \oplus \Delta y^R)$$

$$\leq DP^{S_1}_{max}.DP^{S_2}_{max} \sum_{\Delta\alpha} DP^{S_3}(\Delta x^L \oplus \Delta\alpha \to \Delta y^L \oplus \Delta y^R)$$

$$= DP^{S_1}_{max}.DP^{S_2}_{max} = p^2.$$

Tóm lại ta có Định lý 3.36 được chứng minh hoàn toàn. -(ĐPCM)-

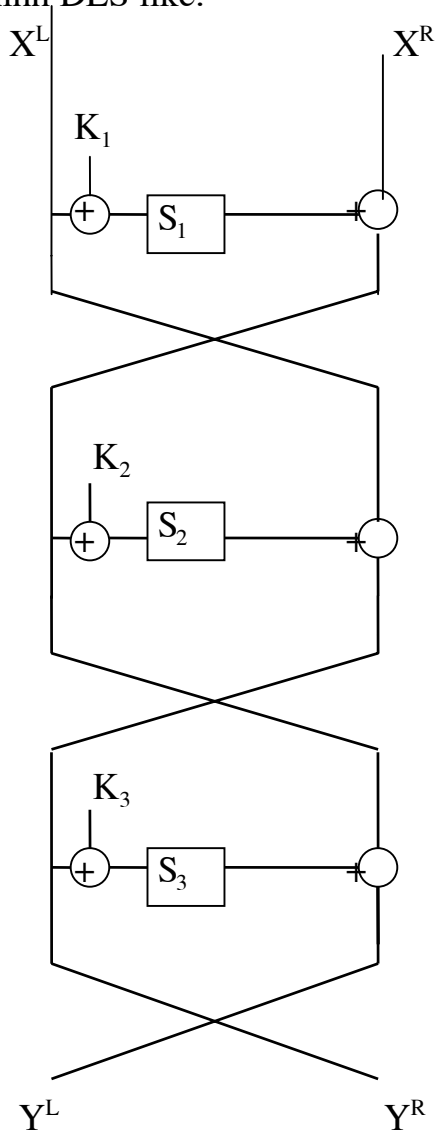
Nếu các hộp thế S_i trong Hình 3.5 được thay thế bằng các hàm phụ thuộc khoá thì bằng cách chứng minh tương tự phối hợp với các bổ đề đã nêu ta sẽ có kết quả sau.



Hình 3.6.

Định lý 3.37. Đối với hàm n-vòng F được cho trong Hình 3.6, giả thiết n ≥ 3, và giả sử rằng mỗi một hàm F_i đều là song ánh với bất kỳ khoá nào và có các độ đo vi sai (hoặc độ đo tuyến tính) thoả mãn $DP_{max}^{S_i} \le p$ (hoặc tương ứng $LP_{max}^{S_i} \le p$). Khi đó ta sẽ có các độ đo của hàm F sẽ được đánh giá bởi bất đẳng thức: $DP_{max}^{F} \le p^2$ (hoặc tương ứng $LP_{max}^{F} \le p^2$) (3.28)

<u>*Nhận xét*</u>: Trên cơ sở các Định lý 3.36, 3.37, chúng ta có thể xây dựng được các hàm vòng kiểu truy hồi đảm bảo có các độ đo tốt về tính an toàn và tốc độ nhanh. Cũng áp dụng cách chứng minh và các lập luận trên đây, chúng ta sẽ nhận được các đánh giá về độ an toàn của mô hình mã dịch quen thuộc- mô hình DES-like.



Hình 3.7.

Định lý 3.38. (Nyberg-Knudsen, 1995 [31]) Đối với hàm n-vòng F được cho trong Hình 3.7, giả thiết n ≥ 3, và giả sử rằng mỗi một hộp S_i đều là song ánh có các độ đo vi sai (hoặc độ đo tuyến tính) thoả mãn $DP_{max}^{S_i} \le p$ (hoặc tương ứng $LP_{max}^{S_i} \le p$). Khi đó ta sẽ có các độ đo của hàm F sẽ được đánh giá bởi bất đẳng thức:

$$DP_{max}^{F} \le 2p^2 \quad \text{(hoặc tương ứng } LP_{max}^{F} \le 2p^2) \qquad (2.29)$$

Các mô hình hàm vòng trên đây cùng với các kết quả của chương 3 cung cấp cho chúng ta những cơ sở lý thuyết quan trọng để thiết kế, xây dựng các hệ mã khối an toàn, hiệu quả.

***Chú ý*:** Để bảo toàn các độ đo an toàn đối với các hộp thế, các phép toán tác động trước khi dữ liệu đi vào hộp thế cần phải là các song ánh tuyến tính (hay đẳng cấu trên không gian khối dữ liệu). Một điều quan trọng nữa là khi chọn các phép toán tác động trước khi dữ liệu đi vào hộp thế cũng cần phải lưu ý tới khả năng, hay tác dụng khuyếch tán đều của phép toán đó. Rõ ràng là, chỉ với phép XOR với khóa K chưa đảm bảo đầy đủ tính khuyếch tán dữ liệu, dù rằng nó là song ánh tuyến tính. Thông thường, theo kinh nghiệm nghiên cứu, ta nên kết hợp giữa phép XOR với các phép dịch vòng (có bước nguyên tố với độ dài khối), cùng với các hoán vị có phân bố khoảng cách ngẫu nhiên. Đây là những công cụ ta đã có và có thể chủ động được.

Điều chú ý cuối cùng là số lượng các vòng lặp của mã khối cần cân đối giữa tốc độ mã dịch và độ đo độ an toàn của hệ mã, ngoài việc dựa vào các căn cứ lý thuyết cần có các căn cứ thông kê số liệu thực hành về các độ đo an toàn đã nêu để lựa chọn số vòng lặp cụ thể. Một mặt nữa là trong điều kiện cho phép thì lược đồ tạo khóa nên dùng kiểu khóa phiên độc lập, để vừa tránh việc dùng khóa quá lâu, hoặc vừa tránh sự phụ thuộc giữa các khóa con trong các hàm vòng, dù rằng về nguyên tắc nếu thám mã tấn công được khóa tại vòng cuối cùng thì hoàn toàn có thể tấn công tiếp các vòng trên để tìm khóa mà không cần dùng tới lược đồ tạo khóa đã biết trong khi vẫn không tăng độ phức tạp tính toán bao nhiêu.

III. ĐỘ AN TOÀN THỰC TẾ CỦA MÃ FEISTEL

III.1. Độ đo an toàn thực tế của cấu trúc Feistel (cấu trúc ngoài cùng)

Khi thiết kế các hệ mã Feistel an toàn thực tế, ta cần thiết lập được các đặc trưng vi sai và đặc trưng tuyến tính cực đại.

Như phần trước khái niệm đặc trưng vi sai đã được giới thiệu. Đó là một danh sách (phù hợp nhất) của các XOR đầu vào và XOR đầu ra của mỗi vòng trong hai phép mã hoá của một hệ mã khối. Masey-Lai-Murphy cũng đã giới thiệu khái niệm vi sai, ở đó các XOR đầu vào và đầu ra của các vòng trung gian là không cố định (tức là nó được tính tất cả các đường dẫn trung gian có thể nối giữa các XOR đầu vào và đầu ra của hai đầu của hệ mã khối đó). Nhìn chung xác suất của một *vi sai* là lớn hơn xác suất của một *đặc trưng* tương ứng. Tuy nhiên đối với các hệ mã Feistel DES-like, sẽ là cực khó để tìm một vi sai hữu ích s-vòng, khi s > 4 mà với chúng thì xác suất của *vi sai* chắc chắn lớn hơn xác suất của *đặc trưng* tương ứng.

Tuy nhiên trong [31] đã chỉ ra rằng trong một mã lặp DES-like R-vòng với các khoá vòng độc lập thì *xác suất của một vi sai s-vòng* bất kỳ đều bị chặn trên bởi số 2. $(p_{max})^2$, ở đây s = 4, 5, ..., R và p_{max} là *xác suất cực đại của đặc trưng một vòng* không tầm thường. Yếu tố này cũng đã được sử dụng để làm cận trên của của các xác suất của các vi sai tốt nhất (tương ứng với hệ mã khối an toàn-chứng minh được). *Từ chỗ đòi hỏi khoá vòng độc lập nói chung phi thực tế,* nên định nghĩa sau đây là hữu ích.

Định nghĩa 3.39: Một hệ mã khối với khoá vòng phụ thuộc được xem là *an toàn thực tế* chống lại tấn công vi sai nếu không tồn tại một *đặc trưng* nào với xác suất đủ lớn để tấn công vi sai dưới giả thiết khoá độc lập là thành công.

Chú ý: Độ phức tạp của tấn công vi sai là xấp xỉ với giá trị nghịch đảo của xác suất đặc trưng được sử dụng trong tấn công đó.

Mặt khác để tránh tấn công 2-vòng đầu cuối ta cần xem xét độ an toàn của hệ mã (R-2)-vòng, tức các đặc trưng (r-2) vòng cần được khảo sát.

Tiếp theo ta sẽ đưa ra khái niệm đặc trưng tuyến tính.

Định nghĩa 3.40 (Knudsen): Một đặc trưng tuyến tính một vòng là một danh sách của các bít đầu vào, khoá, và đầu ra của một mã khối và một xác suất p, sao cho giá trị boolean đạt được bằng cách cộng (mod 2) các bít đầu vào và khoá bằng giá trị boolean đạt được bằng cách cộng (mod 2) các bít đầu ra với xác suất p. *Một đặc trưng tuyến tính r-vòng là một sự ghép nối của r đặc trưng tuyến tính một vòng.*

Có thể trong các vòng nào đó các xấp xỉ đặc trưng tuyến tính là không cần thiết khi ghép nối thành đặc trưng toàn cục, nên ta gọi các vòng đó là các đặc trưng tầm thường. Tương tự với tấn công vi sai ta có định nghĩa sau.

Định nghĩa 3.41: Một hệ mã khối với khoá vòng phụ thuộc được xem là *an toàn thực tế* chống lại tấn công tuyến tính nếu không tồn tại một *đặc trưng* tuyến tính nào với xác suất đủ lớn để tấn công tuyến tính dưới giả thiết khoá độc lập là thành công.

Chú ý: Độ phức tạp của tấn công tuyến tính là xấp xỉ với giá trị bình phương nghịch đảo của độ lệch xác suất (so với 1/2) của đặc trưng tuyến tính được sử dụng trong tấn công đó. Và cũng vậy, để tránh tấn công 2-vòng đầu cuối ta cần xem xét độ an toàn của hệ mã (R-2)-vòng, tức các đặc trưng tuyến tính (R-2) vòng cần được khảo sát.

* Trong [18] Knudsen đã đưa ra phương pháp để đánh giá cận trên của xác suất đặc trưng cho hệ mã Feistel. ý tưởng cơ bản là tìm số tối thiểu các đặc trưng một vòng <u>không tầm thường</u> có thể phải có mặt trong một đặc trưng (r-2)-vòng. Khi đó xác suất cực đại của một đặc trưng một vòng <u>không tầm thường</u> sẽ cho một cận trên của xác suất cực đại của đặc trưng (r-2)-vòng tốt nhất có thể.

Giả sử rằng chỉ có cách tấn công mã Feistel bằng cách tìm các đặc trưng vi sai (hay đặc trưng tuyến tính) tốt nhất, tức là rất khó có thể tìm được các *vi sai (hay xấp xỉ tuyến tính)*, thì các số liệu sau:

+Xác suất của đặc trưng vi sai (tuyến tính) 1-vòng <u>không tầm thường</u> tốt nhất

+Số vòng trong đặc trưng đó

sẽ cho ta tính được cận dưới độ phức của hai tấn công này. Cận dưới này có thể đủ để khẳng định độ an toàn -chứng minh được đối với các ứng dụng thực tế trước hai tấn công trên, nếu như xác suất của đặc trưng vi sai (tuyến tính) 1-vòng không tầm thường tốt nhất là đủ nhỏ.

Bằng cách sử dụng các hộp thế có các độ đo tốt ta có thể thiết lập các hàm vòng có các đặc trưng 1-vòng đủ tốt để chống được hai tấn công cơ bản hiện nay.

Trong [18] Knudsen đã chỉ ra rằng với hệ mã Feistel, giả thiết các hàm vòng là song ánh thì số tối thiểu các đặc trưng vi sai (hay tuyến tính) 1-vòng không tầm thường cần 2 cho mỗi bộ 3 vòng liên tiếp tương ứng. Từ đó ta có công thức sau:

$$UDCP_{max}^{R} \le p^{2.r}, ULCP_{max}^{R} \le q^{2.r}, khi \quad R = 3r, 3r+1.$$

$$UDCP_{max}^{R} \le p^{2.r+1}, ULCP_{max}^{R} \le q^{2.r+1}, khi \quad R = 3r+2.$$

(3.40)

Khi xem xét đến tấn công-2R (2-vòng đầu cuối), cần đánh giá cận trên của xác suất đặc trưng vi sai/tuyến tính cực đại của mã Feistel (R-2)-vòng.

III.2. Một kiểu thiết kế hàm vòng 2-SPN (cấu trúc giữa)

Cấu trúc mạng thay thế hoán vị 2-tầng (2-SPN) là một cấu trúc tốt đã được các nhà khoa học Nhật bản sử dụng trong thiết kế hệ mã khối E2. Cấu trúc này có 2 tầng S-hộp được xen giữa bởi một tầng tuyến tính P. Với cấu trúc này số các S-hộp hoạt động trong một hàm vòng có thể tăng lên nhanh chóng. Tầng tuyến tính P cần được thiết kế sao cho số các S-hộp hoạt động trong một hàm vòng là lớn, và không quá phức tạp với các ứng dụng phần cứng. Các tác giả thiết kế E2 đã đưa ra các tiêu chí lựa chọn và đã tìm được phép biến đổi tuyến tính trong tầng tuyến tính đảm bảo số S-hộp hoạt động khá lớn. Chúng tôi cũng đã học tập ý tưởng này, nhưng do khả năng hạn chế nên không thể lặp lại những gì họ đã làm cộng với những ý định riêng của mình được. Từ đó chúng tôi quyết định theo phương án đơn giản hơn, dễ chứng minh hơn, nhưng không kém phần hiệu quả. Chúng tôi đã dùng phép dịch vòng đóng vai trò là tầng biến đổi tuyến tính, với bước dịch lựa chọn cẩn thận. Để tránh sự đơn giản của tầng

tuyến tính, trong tầng phi tuyến gồm các S-hộp chúng tôi đã dùng 2 S-hộp luân phiên thay vì chỉ sử dụng 1 S-hộp trong E2. Điều này hy vọng sẽ làm tăng hiệu quả khuyếch tán của tầng S-hộp.

Về mặt lý thuyết có thể thấy số tối thiểu các S-hộp hoạt động trong một hàm vòng với cấu trúc Feistel sẽ là 2. Nhưng qua thử nghiệm lập trình chúng tôi thấy số tối thiểu các S-hộp hoạt động qua hai vòng trong mỗi 3-vòng sẽ là .

Như vậy về mặt lý thuyết các lượng liên quan tới tấn công vi sai và tuyến tính sẽ được đánh giả bởi công thức:

$$\begin{aligned} &UDCP_{max}^{R} \le {p_s}^{2.x2r}, ULCP_{max}^{R} \le {q_s}^{2x2.r}, khi \quad R = 3r, 3r+1. \\ &UDCP_{max}^{R} \le {p_s}^{2x(2.r+1)}, ULCP_{max}^{R} \le {q_s}^{2x(2.r+1)}, khi \quad R = 3r+2. \end{aligned} \tag{3.41}$$

Nếu ta chọn các S-hộp sao cho $p_s = q_s = 2^{-6}$, R =16 thì độ phức tạp của tấn công vi sai và tuyến tính sẽ vào cỡ 2^{120} . Đây là con số an toàn chấp nhận được hiện nay.

IV. LƯỢC ĐỒ KHOÁ, CÁC PHÉP BIẾN ĐỔI ĐẦU VÀO ĐẦU RA CỦA HỆ MÃ KHỐI

I.1. PHÂN LOẠI LƯỢC ĐỒ KHOÁ CỦA CÁC HỆ MÃ KHỐI

Một vấn đề hết sức quan trọng trong thiết kế mã khối đó là xây dựng lược đồ tạo khoá cho hệ mã. Thông thường một hệ mã khối lặp thường có số vòng tương đối lớn. Khoá phiên không thể có dộ dài tuỳ ý, do đó từ khoá bí mật cần thiết phải xây dựng một thuật toán để tạo ra đủ số khoá con cần thiết để cung cấp cho các vòng lặp. Khoá chính thường dài từ 128 bít đến 512 bit, trong khi tổng số bít khoá con có thể lên tới hàng ngàn bít. Do vậy việc nghiên cứu lược đồ tạo khoá là không thể tránh khỏi. Lược đồ tạo khoá không chỉ đơn thuần cung cấp các khoá con cho các vòng lặp trong hệ mã khối mà nó còn đóng góp vai trò quan trọng trong độ an toàn của chính hệ mã đó.

Tuy nhiên chúng ta cũng đã thấy một số lược đồ khoá đã có những điểm sơ hở để thám mã có thể lợi dụng, như lược đồ quá đơn giản, lược đồ tạo ra các dạng khoá quan hệ, hay có sự tương tự lặp lại trong các giai đoạn

tạo khoá con. Để tránh các dạng tấn công đã xét, Knudsen đã đưa ra một số yêu cầu đối với một lược đồ tạo khoá mạnh đó là tất cả các khoá phải tốt như nhau, và không có các quan hệ đơn giản.

Định nghĩa 4.1: Xét một hệ mã khối lặp r-vòng, cỡ khối là 2m-bit với r khoá con vòng, mỗi khoá con có độ dài là n-bit. Một lược đồ khoá mạnh phải có các tính chất sau:

-Cho trước bất kỳ s-bit của r khoá con vòng được thiết kế từ một khoá chính chưa biết, khi đó khó có thể tìm ra được rn-s bít khoá còn lại từ s-bit khoá đã biết.

-Cho trước một quan hệ nào đó giữa hai khoá chính, khi đó khó có thể dự đoán được các quan hệ giữa bất kỳ các khoá con vòng nào được thiết kế từ các khoá chính đó.

Nói một cách đơn giản hơn là lược đồ khoá mạnh là lược đồ mà các hiểu biết về một khoá con nào đó không làm dò dỉ bất kỳ thông tin gì đối với các khoá con khác trong lược đồ đó. Trong phần này trước hết chúng ta đi phân loại các lược đồ khoá đã có, và sau đó đưa ra một số đề xuất liên quan đến việc xây dựng lược đồ khoá mạnh.

Các lược đồ khoá hiện tại có thể được chia thành hai kiểu.

*Kiểu 1 là kiểu ở đó tri thức về một khoá con vòng sẽ cung cấp một cách duy nhất các bít khoá của các khoá con vòng khác hay của khoá chính. Trong đó:

+Kiểu 1A là kiểu đơn giản nhất dùng khoá chính trong mỗi vòng mã hoá.

+Kiểu 1B, các khoá con vòng được tạo từ khoá chính theo cách sao cho hiểu biết về một khoá con vòng bất kỳ có thể xác định trực tiếp các bít khoá khác trong các khoá con vòng khác hay trong khoá chính. DES, IDEA, LOKI, GOST là các ví dụ về kiểu này.

+Kiểu 1C, tri thức về một khoá con vòng có thể giúp xác định một cách không trực tiếp các bít khoá khác trong các khoá con vòng khác hay trong khoá chính. Một vài thao tác cần thiết phải dược sử dụng giúp xác định tìm ra các bít khoá khác hay trong khoá chính. Ví dụ về kiểu này là lược đồ khoá của hệ CAST, SAFER.

Trong CAST, mỗi một vòng trong 4 vòng đầu tiên đều sử dụng 16 bít của khoá chính, chia nó thành 2 khối 8-bit, mỗi khối cho qua một S-hộp cố định. Các đầu ra của mỗi S-hộp là 32-bit, và kết quả được XOR với nhau tạo nên khoá con vòng đó. Nếu biết một khoá con này, chúng ta phải thử

2^{16} bít là đầu vào cho mỗi S-hộp để tìm ra xâu bít nào cho đầu ra phù hợp với khoá con đã biết. Chú ý rằng nếu biết bất kỳ khoá con nào từ vòng thứ 5 trở đi nó đều không thể áp dụng cách trên đây để thu được các thông tin khác về khoá.

Trong SAFER, nếu K = ($k_{1,1}$, ...$k_{1,8}$) là một khoá chính 8-byte, khi đó khoá con 8-byte vòng thứ i, $K_{i,j}$ sẽ được xác định như sau:

$$k_{i,j} = k_{i-1,j} <<3 \qquad (3.42)$$

$$K_{i,j} = k_{i,j} + bias[i, j] \bmod 256 \qquad (3.43)$$

Giả sử $K_{i,j}$ có thể được xác định theo một cách nào đó. Từ phương trình (3.43), $k_{i,j}$ có thể được xác định bởi $K_{i,j}$ - bias[i, j] mod 256, từ chỗ bias[i, j] là một hằng số đã biết khi biết j, j. Sử dụng quan hệ truy hồi $k_{i,j} = k_{i-1,j}$ <<3, $k_{i-1,j}$ có thể được xác định, từ đó $K_{i-1,j}$ có thể được xác định từ quan hệ truy hồi $K_{i-1,j} = k_{i-1,j}$ + bias[i-1, j] mod 256, ở đây bias[i-1, j] là hằng số đã biết. Như vậy các hiểu biết về $K_{i,j}$ có thể cho phép chúng ta xác định được duy nhất $K_{i-1,j}$ là khoá vòng trước đó. Rõ ràng thủ tục này có thể được tiếp tục để xác định ra được khoá chính.

*Kiểu 2, là các lược đồ tạo khoá của các hệ mã mà ở đó tri thức về một khoá con vòng bất kỳ đều không ta biết thêm bất kỳ bít nào của các khoá con vòng khác hay của khoá chính như trong trường hợp của Kiểu 1. Nhiều hệ mã hiện đại thuộc lớp này. Trong quá trình tạo khoá cụ thể, một số hệ mã sử dụng thông tin về các đại lượng đã dùng để tạo ra khoá con vòng trước đó. Trong nhiều trường hợp khoá con được tạo mới thường được sử dụng như là một phần của quá trình tạo của các khoá tiếp theo. Như vậy tri thức về một khoá vòng cụ thể không cung cấp khả năng có thể tìm được khoá ngay trước hoặc các khoá tiếp theo. RC5 là một ví dụ, trong hệ mã này khoá con vòng thứ i, S[i] được tính toán như sau: S[i] = (S_c[i] +S[i-1] + B_{i-1})<< 3, ở đây S_c[i] là một hằng số đã biết, S[i-1] là khoá con vòng thứ (i-1), và B_{i-1} phụ thuộc vào khoá chính. Như vậy, nếu biết S[i], thì S[i-1] + B_{i-1} cũng biết. Nhưng hiểu biết này không thể lợi dụng được nếu chúng ta không thể xác định được B_{i-1}, và nếu biết được B_{i-1} thì chúng ta sau đó mới tính ra được S[i-1]. Tương tự từ chỗ S[i+1] = (S_c[i+1] +S[i] + B_i)<< 3, các hiểu biết về S[i] cũng không giúp xác định được S[i+1]. Lược đồ như thế được gọi là thuộc kiểu 2A.

Các lược đồ trong đó tri thức về một khoá con vòng không cung cấp một tí gì về thông tin đối với các khoá con vòng khác hay khoá chính, cho tới khi nào có thể giải được các bài toán khó chẳng hạn như việc nghịch đảo hàm một chiều, chúng ta gọi nó thuộc kiểu 2B. Hệ mã REDOC II có

lược đồ khoá thoả mãn tính chất này. Một cách để tạo ra lược đồ khoá kiểu 2B đó là đảm bảo rằng mỗi khoá con vòng là một hàm một chiều của chỉ riêng một mình khoá chính. Theo cách đó, các khoá con vòng xuất hiện một cách độc lập dù rằng tất cả chúng dều phụ thuocc vào khoá chính. Sự khác nhau giữa các kiểu 2A và 2B là ở chỗ trong kiểu 2A, nếu biết thêm một chút thông tin thì nó có thể trở thành một bài toán đơn giản trong việc suy diễn ra các khoá con vòng khác; trong khi ở kiểu 2B, việc biết thêm một chút thông tin vẫn còn là bài toán khó đối với việc tìm ra thông tin về khoá con vòng khác hay khoá chính.

Kiểu 2C là kiểu trong đó việc tạo các khoá con vòng là hoàn toàn độc lập. Rất ít hệ mã có lược đồ khoá thuộc kiểu này, lý do là bài toán quản lý khoá sẽ trở nên không thể khi độ dài khoá chính quá lớn. Một ví dụ về kiểu 2C đó là DES với khoá con vòng độc lập (768 bít khoá chính-gọi là DESI).

Theo sự phân loại trên chúng ta thấy mỗi kiểu loại đều có những ưu điểm và nhược điểm, có những kiểu cho độ an toàn cao nhưng khong thuận tiện trong sử dụng, có kiểu thuận tiện trong sử dụng thực tế nhưng độ an toàn lại phụ thuộc vào phần ngẫu nhiên hoá dữ liệu, thậm chí phần lược đồ khoá có thể tạo ra các kẽ hở để thám mã có thể áp các tấn công trên chúng. Việc sử dụng kiểu loại nào phụ thuộc vào sự cân đối độ an toàn và tính tiện dụng trong thực tiễn cũng như tính khả thi của hệ mật trong các môi trường thực tế.

IV.2. MỘT SỐ LƯỢC ĐỒ KHOÁ MẠNH

Trong [6], các tác giả đã đề xuất một lược đồ khoá kiểu 2B sau đây. Trước hết giả thiết tồn tại một hàm một chiều mạnh OWF. Giả sử MK là khoá chính của một hệ mã khối r-vòng, quá trình tạo các khoá con vòng như sau:

+Bước 1: OWF(MK) = Khoá con vòng (1)

+Bước 2: Với i =1 đến r-1

Thực hiện hoán vị bit MK để tạo ra MK_i

$OWF(MK_i)$ = Khoá con vòng (i+1).

Một chú ý trong thuật toán trên là mỗi một khoá con vòng đều được tạo bởi toàn bộ các bit của khoá chính MK. Điều này có thể tạo ra hiệu ứng thác, tức là sự thay đổi một bít của khoá chính sẽ tạo ra sự thay đổi nhiều ở trong mỗi khoá con.

*Knudsen [17] cũng đã đề xuất một lược đồ khoá sau.

Giả sử $E_K(.)$ là một mã Feistel r-vòng với độ dài khối là 2m, sử dụng khoá chính K tạo ra r khoá con vòng độ dài n-bit, với $n \leq 2m$.

+Bước 1. Xác định một lược đồ khoá khởi động (ban đầu), với chúng đầu vào là khoá K cho đầu ra là r khoá vòng phụ thuộc là $\{K_i\} = K_1, ..., K_r$.

(a) $E_{K_i}(.)$ là an toàn chống lại tấn công bản rõ đã biết sử dụng phép mã hoá nhiều nhất r bản rõ đã biết, theo nghĩa rút gọn thông tin trong đó việc có được một thông tin không tầm thường là không thể.

(b) $E_{\{K_i\}}(.)$ không chứa các quan hệ đơn giản như $G_1(P, K) = P \oplus \alpha$, α là hằng số

+Bước 2. Xác định các khoá con vòng $\{RK_l\} = RK_1, ..., RK_r$ sử dụng cho phép mã hoá như sau

$$RK_l = nMSB(E_{\{K_i\}}(IV \oplus l))$$

ở đây IV là một giá trị cố định và ký hiệu nMSB(X) nghĩa là n-bít bên trái nhất của X.

Chú ý lược đồ khoá trên là mạnh theo nghĩa không chứa các khoá quan hệ và tri thức về một khoá con vòng nào đó không giúp gì trong xác định các thông tin về khoá con vòng khác cũng như khoá chính K.

IV.3. VIỆC SỬ DỤNG HOÁN VỊ TRONG CÁC HÀM VÒNG VÀ CÁC PHÉP BIẾN ĐỔI ĐẦU VÀO ĐẦU RA CỦA MỘT HỆ MÃ KHỐI

Để đảm bảo cho tính khuyếch tán dữ liệu hoàn toàn trong phần ngẫu nhiên hoá dữ liệu của một hệ mã khối, vấn đề xây dựng và lựa chọn hoán vị sử dụng trong các hàm vòng là hết sức quan trọng. Trên các tài liệu đã công khai, chúng tôi chưa thấy có tài liệu nào cho những chỉ dẫn cụ thể về vấn đề này. Tất nhiên, các hoán vị lựa chọn ở đây phải đảm bảo một số tính chất nào đó, như có khoảng cách trung bình là lớn, hay có khoảng cách đều hoặc khoảng cách ngẫu nhiên. Phép dịch bít trong chuẩn mã dữ liệu Sôviết GOST có thể xem như một hoán vị có khoảng cách đều nhưng nhỏ. Do đó theo một nghĩa nào đó sử dụng hoán vị này cũng sẽ cho độ khuyếch tán tốt nhưng chậm, tức là phải với số vòng khá lớn. Hoán vị trong DES có khoảng cách trung bình khá lớn nhưng lại là hoán vị theo bít nên tốc độ chậm ảnh hưởng trong thiết kế phần cứng. Có thể sử dụng

hoán vị ngẫu nhiên được lựa chọn theo các thuật toán cụ thể, nhưng đó là bài toán khó nếu số phần tử trong hoán vị là lớn. Một cách tích cực ở đây là dựa vào các kết quả đã nghiên cứu sâu về các dạng hoán vị trong mật mã ta có thể thiết kế và lựa chọn được các hoán vị đảm bảo các yêu cầu đặt ra, chẳng hạn các hoán vị giả ngẫu nhiên, hoán vị tạo bởi phép đồng dư tuyến tính hay phi tuyến. Tuy nhiên các lựa chọn cụ thể cần phải được tính toán cẩn thận để đảm bảo tối ưu yêu cầu khuyếch tán dữ liệu nhanh, đồng thời không ảnh hưởng lớn đến tốc độ mã hoá của hệ mã khối.

Đối với việc lựa chọn các phép biến đổi đầu vào đầu ra cho một hệ mã khối kiểu Feistel, hay bất kỳ hệ mã khối nào có sử dụng các hàm vòng lặp, người thiết kế thuật toán cũng cần phải hết sức chú ý. Chẳng hạn DES, sử dụng các hoán vị cố định cho trước, đối với thám mã hoán vị này hoàn toàn không có ý nghiã gì cả, nhưng nếu chọn một phép biến đổi nào đó phụ thuộc khoá, thì đương nhiên lược đồ khoá cần phải tính toán ra khoá đó và có nghĩa phần thiết kế lược đồ khoá cần phải đảm bảo an toàn cho cả khoá con vòng cùng với khoá cho phép biến đổi đầu vào đầu ra. Đây cũng là một yêu cầu không đơn giản. Một chú ý nữa là các phép biến đổi đầu vào đầu ra trong hệ Feistel phải đảm bảo tính đối xứng thuận nghịch, khi đó mới không ảnh hưởng tới qui trình mã dịch tương tự kiểu E/D như đã nói trong chương mở đầu. Có một số hệ mã khối đã đề cập đến vấn đề này như trong hệ E2, hệ Rijndael, Twofish...Chúng ta hoàn toàn có thể nghiên cứu học hỏi và vận dụng trong các thiết kế cụ thể. Tất nhiên, khi lựa chọn các phép biến đổi đầu vào đầu ra cần phải thiết kế ngay lược đồ khoá tương ứng đảm bảo tránh các tấn công đã khảo sát trong các phần trên.

CHƯƠNG 4: KHẢO SÁT MÃ KHỐI THEO NHÓM SINH CỦA CÁC HÀM MÃ HOÁ

Việc tìm các tính yếu của một hệ mã khối căn cứ vào những đặc tính cụ thể của nhóm sinh của các hàm mã hoá của hệ mã để trên cơ sở đó hình thành nên những tiêu chuẩn khi thiết kế xây dựng các hệ mã khối an toàn là một hướng đi được một số tác giả như Kennth G. Paterson (xem [33]), Ralph Wernsdorf (xem [38]), Sarval Patel, Zulfikar Ramran và Ganapathy (xem [32])...quan tâm và cũng đã đưa ra được những kết quả có ý nghĩa. Trong chương này chúng tôi bắt chước theo những ý tưởng của các tác giả nêu trên, trong đó có trình bày lại kết quả theo chúng tôi cho là có ý nghĩa nhất về mặt mật mã đó là khái niệm nguyên thuỷ của nhóm các phép thế của tác giả Kennth G. Paterson rồi lấy đó làm trong tâm phát triển. Công lao chủ yếu của chúng tôi đưa ra trong bài này là đưa ra các kết quả liên quan đến khái niệm t-phát tán và t-phát tán mạnh cùng với ý nghĩa mật mã của chúng. Qua các kết quả đã đưa ra cũng toát lên một vấn đề rất thực tế đó là mọi tính yếu về nhóm các phép thế có ảnh hưởng đến tính an toàn của hệ mật thì việc loại bỏ chúng chỉ là cần thiết vì rất dễ khắc phục các khuyết tật hình thức trên nhóm sinh (chỉ bằng cách bổ xung vào tập các hàm mã hoá cùng lắm là 2 hàm đơn giản) trong khi bản chất mật mà chỉ phụ thuộc vào chính tập các hàm mã hoá.

I. KHÁI NIỆM CƠ BẢN

I.1.MÃ KHỐI

Một hệ mã khối m-bit được định nghĩa là một bộ 3 $\{\mathcal{M}, \mathcal{C}, E, D\}$, trong đó $\mathcal{M}$ là không gian các bản rõ, $\mathcal{C}$ là không gian các bản mã, E là tập các hàm mã hoá còn D là tập các hàm giải mã với $\mathcal{M} = \mathcal{C}$ ={tập các sâu bit 0, 1}, E là tập con nào đó các phép thế trên tập $\{0;1\}^m$ và D là tập các phép thế ngược của E. Nếu ta ký hiệu $E=\{f_k: \{0;1\}^m \to \{0;1\}^m$ với $k \in K\}$ thì $D=\{f_k^{-1}: \{0;1\}^m \to \{0;1\}^m$ với $k \in K\}$ và tập K được gọi là tập khoá của hệ mật.

A muốn truyền thông báo M cho B theo phương thức sau.

Bước 1. A và B thoả thuận với nhau khoá $k \in K$ của phiên liên lạc.

Bước 2. (Quá trình mã hoá)

Thông báo M được chia thành các khối m bit $M=M_1M_2...M_r$ và được mã hoá thành $C=C_1C_2...C_t$ với $C_i=f_k(M_i)$ $(i=1\div t)$.

Bước 3. (Quá trình giải mã)

Khi nhận được $C=C_1C_2...C_t$ thì B thực hiện tìm lại $M=M_1M_2...M_r$ theo công thức $M_i=f_k^{-1}(C_i)$ $(i=1\div t)$.

I.2. NHÓM SINH CỦA CÁC HÀM MÃ HOÁ

Ta biết rằng tập tất cả các phép thế trên một tập hữu hạn với phép toán nhân ánh xạ lập thành một nhóm, theo lý thuyết nhóm với một tập các phép mã hoá E trên tập $X=\{0;1\}^m$ luôn tồn tại một nhóm con nhỏ nhất chứa E và được gọi là nhóm sinh bởi E. Từ nay chúng ta ký hiệu nhóm sinh bởi E của một mã khối là G, và trong chương này ta sẽ quan tâm đến các tính chất của G liên quan đến tính an toàn của hệ mật.

II. MỘT SỐ TÍNH CHẤT CƠ BẢN CỦA G

II.1. NHÓM CON BẤT ĐỘNG TRÊN MỘT TẬP

Định nghĩa 4.1. *Cho $\varnothing\neq Y\subseteq X$, ta ký hiệu $G_Y=\{g: g\big/_Y:Y\to Y\}$ và dễ dàng thấy rằng G_Y là nhóm con của G và được gọi là nhóm con bất động trên tập Y.*

Ta lại biết rằng khi này G được phân hoạch thành các lớp kề của G_Y và mỗi lớp kề của G_Y thoả mãn tính chất sau.

Tính chất 4.2. *Với mọi $g\in G$ ta có $G_Yg=\{h\in G: h:Y\to Yg\}$.*

Chứng minh.

Rõ ràng $\forall f\in G_Y$ ta có $Y\xrightarrow{f}Y$ mà $Y\xrightarrow{g}Yg$ vậy $Y\xrightarrow{fg}Yg$.

Ngược lại $\forall h:Y\to Yg$ thì do $g^{-1}: Yg\to Y$ nên $hg^{-1}: Y\to Y$ hay $hg^{-1}\in G_Y$. mà $h=hg^{-1}g$ nên $h\in G_Yg$.□

II.2. TÍNH PHÁT TÁN CỦA G

Kết quả 4.3. *Các tập $xG=\{y: y=xg, g\in G\}$ với $x\in X$ lập thành một phân hoạch của X.*

Chứng minh.

Hiển nhiên $X=\bigcup_{x\in X} xG$ cho nên ta chỉ cần chứng tỏ $\forall x,y\in X$ thì

$$\left[\begin{array}{l} xG = yG \\ xG \cap yG = \phi \end{array}\right.$$

Quả vậy nếu $z\in xG\cap yG$ tức là $z=xg_1=yg_2$ hay $x=yg_2g_1^{-1}$ do đó $\forall w\in xG$ ta có $w=xg= yg_2g_1^{-1}g\in yG$ hay $xG\subseteq yG$. Bao hàm thức ngược lại chứng minh tương tự.□

Kết quả 4.4. *Tập xG đóng kín với G theo nghĩa: $\forall w\in xG$, nếu $w=yg$ với g nào đó thì $y\in xG$.*

Chứng minh.

Từ $w\in xG$ ta có $w=xh$ với nào đó $h\in G$, lại từ $w=yg$ nên $y=wg^{-1}=xhg^{-1}$ hay $y\in xG$.□

Định nghĩa 4.5. *G được gọi là có tính phát tán nếu xG=X.*

Từ tính chất 4.2 thì G được phân hoạch thành các lớp kề của nhóm con $G_{\{x\}}$ mà số các lớp kề này chính bằng số các tập {xg} khác nhau hay là số phần tử của tập xG vậy ta có ngay hệ quả sau.

Hệ quả 4.6. *$\forall x\in X$ ta có $^{\#}G=^{\#}G_{\{x\}}.^{\#}(xG)$.*□

Chú ý rằng nếu G có tính phát tán $^{\#}(xG)=2^m$ nên $^{\#}G$ là bội của 2^m.

Ký hiệu $\mathcal{X}_t=\{Y\subseteq X: {}^{\#}Y=t\}$ khi đó $\forall g\in G$ ta có thể cảm sinh ánh xạ g* trên $\mathcal{X}_t$ như sau: Yg*=Yg. Nếu các g* vẫn là các phép thế thì tập các phép thế này là G* cũng là nhóm và theo kết quả 4.3 ta có các tập YG* tạo ra một phân hoạch của $\mathcal{X}_t$ và ta sẽ gọi G là t-phát tán nếu $YG^*=\mathcal{X}_t$.

Chú ý rằng nói chung lực lượng của G* không quá lực lượng của G. Trường hợp $t=^{\#}X$ thì G* chỉ còn đúng một ánh xạ và tính $^{\#}X$-phát tán luôn luôn đúng với mọi G.

II.3. TÍNH NGUYÊN THUỶ CỦA G

II.3.1. Block của nhóm các phép thế.

Định nghĩa 4.7. *Cho G là một nhóm các phép thế nào đó trên tập hữu hạn X. Tập $Y \subseteq X$ được gọi là một block của G nếu* $\forall g \in G$:

$$\begin{cases} Y = Yg \\ Y \cap Yg = \varnothing \end{cases} \qquad (4.1)$$

Chúng ta dễ dàng thấy rằng các tập X, ∅ và {y} là các block và được gọi là các bock tầm thường.

Tính chất 4.8. *Nếu Y_1 và Y_2 là các block thì $Y=Y_1 \cap Y_2$ cũng là block.*

Chứng minh.

$\forall g \in G$ ta có nếu $Y \cap Yg \neq \varnothing \Rightarrow \begin{cases} Y_1 \cap Y_1 g \neq \varnothing \\ Y_2 \cap Y_2 g \neq \varnothing \end{cases} \Rightarrow \begin{cases} Y_1 = Y_1 g \\ Y_2 = Y_2 g \end{cases} \Rightarrow Y=Yg.\square$

Tính chất 4.9. *Nếu Y là block thì* $\forall g \in G$ *ta có Yg cũng là block.*

Chứng minh.

Nếu $\exists h \in G$ để Yh không phải là block tức là $\exists g \in G$ sao cho

$$\begin{cases} Yh \setminus Yhg \neq \varnothing \\ Yh \cap Yhg \neq \varnothing \end{cases} \qquad (4.2)$$

như vậy ta có

$$\begin{cases} Y \setminus Yhgh^{-1} = Yhh^{-1} \setminus Yhgh^{-1} \neq \varnothing \\ Y \cap Yhgh^{-1} = Yhh^{-1} \cap Yhgh^{-1} \neq \varnothing \end{cases} \qquad (4.3)$$

tức là Y không phải block.□

Hệ quả 4.10. *Y là block của G khi và chỉ khi công thức (4.1) được thoả mãn* $\forall g \in R$ *với R là tập sinh của G.*

Tính chất 4.11. *Nếu* $Y \neq \varnothing$ *là block của nhóm phát tán G thì tập {Yg: g∈G} lập thành một phân hoạch trên X.*

Quả vậy, từ công thức (4.1) chúng ta đã có {Yg} là các tập con rời nhau của X cho nên để chứng minh tính chất này ta chỉ cần chỉ ra rằng

$X \subseteq \bigcup_{g \in G} Yg$ (4.4). Do $Y \neq \emptyset$, giả sử $x \in Y$, theo tính phát tán của G thì xG=X cho nên $\forall y \in X$ sẽ tồn tại $g \in G$ sao cho xg=y và như vậy $y \in Yg$ vậy tính chất đã được chứng minh.□

Định nghĩa 4.12. *Nhóm G phát tán trên X được gọi là nguyên thuỷ nếu chỉ tồn tại những block tầm thường.*

Tính chất 4.13. *Cho Y là một block không tầm thường của nhóm phát tán trên X và giả sử* $\{Y_i: i=1 \div r\}$ *là một phân hoạch X theo Y. Khi đó:*

(a).Với mỗi $g \in G$ *tồn tại hoán vị* π *trên tập {1,2,...r}sao cho* $\forall i=1 \div r$ *ta có:*

$$Y_i g = Y_{i\pi}. \qquad (4.5)$$

(b).ánh xạ ứng $g \in G$ *với* π *nêu trong (a) là đồng cấu nhóm từ G vào nhóm* Π *tất cả các hoán vị trên tập {1,2,...r}.*

III. QUAN HỆ GIỮA CÁC TÍNH CHẤT CƠ BẢN CỦA G VỚI TÍNH AN TOÀN CỦA HỆ MẬT

III.1.TÍNH PHÁT TÁN

Rõ ràng một yêu cầu về tính phát tán của G là hợp lý đối với một hệ mã khối bởi vì trong trường hợp tập các hàm mã hoá trùng với G thì hệ mật như vậy chứa đựng một thuộc tính rất tốt trong mật mã đó là mọi khối mã đều có thể được mã từ một khối rõ bất kỳ.

Ta có thể phân tích kỹ hơn tính yếu của hệ mật trong trường hợp tính phát tán của G không được thoả mãn. Chú ý rằng tính phát tán là tương đương với phân hoạch đưa ra trong kết quả 4.2 có đúng một tập do đó sự không phát tán của G dẫn đến X bị phân hoạch bởi các tập con xG, giả sử $X=X_1+X_2+...+X_t$ với $X_i=x_iG$. Như vậy đối với những khối mã thuộc tập X_i có lực lượng không lớn lắm thì do khối rõ tương ứng của nó chỉ thuộc tập này và có thể tìm được bằng cách vét cạn. Trong trường hợp các tập trong phân hoạch có lực lượng như nhau thì số lượng duyệt toàn bộ các khối rõ có thể đối với mỗi khối mã sẽ là $^{\#}M/t$.

Ngoài ra chúng ta cũng có thể chỉ ra yêu cầu về tính phát tán của một hệ mã khối là chưa đầy đủ qua ví dụ sau đây.

Ví dụ 4.14. Xét hệ mã có $E=\{f:X\rightarrow X\}$ với $xf=(x+1) \pmod{2^m}$, (ở đây ta đồng nhất phần tử x như một véc tơ trong không gian m chiều $\{0;1\}^m$ với giá trị nguyên có biểu diễn nhị phân là dãy m bit toạ độ tương ứng của nó). Hệ mật trên là rất tồi thế nhưng rõ ràng nhóm G của nó có tính phát tán do $\forall x,y\in X$ ta lấy $g=f^t\in G$ với $t=(y-x) \bmod 2^m$ là có ngay $xg=y$ hay $xG=X$.

III.2.TÍNH YẾU CỦA CÁC MÃ KHỐI CÓ G LÀ KHÔNG NGUYÊN THUỶ

Cho bản mã c được mã bằng $g\in G$ nào đó với giả thiết rằng chúng ta luôn thực hiện được việc mã hoá các thông báo m khác bằng mg.

Giả sử $\{Y_1,Y_2,...,Y_r\}$ là một phân hoạch X từ block không tầm thường Y nào đó của G. Với mỗi i lấy $m_i\in Y_i$ và tính $c_i=m_ig$, từ đó xác định $\pi\in\Pi$ được tương ứng bởi g.

Xác định j sao cho $c\in Y_j$. Khi này hiển nhiên $m\in Y_k=Y_j\pi^{-1}$ là một tập chỉ có $\frac{|X|}{r}$ phần tử.

Nếu như việc tìm $m\in Y_k$ chỉ theo phương thức vét cạn thì số bước thực tấn công trên chỉ là

$$r+\frac{|X|}{r}\geq 2\sqrt{|X|} \qquad (4.6)$$

Qua phân tích trên cho chúng ta một yêu cầu cần thiết đối với việc thiết kế một hệ mã khối đó là nhóm G của nó phải có tính nguyên thuỷ. Cũng như yêu cầu về tính phát tán chúng ta cũng có thể chỉ ra yêu cầu về tính nguyên thuỷ chũng không đầy đủ qua ví dụ sau.

Ví dụ 4.15. Xét hệ mã khối có tập các hàm mã hoá là E=G+{g*} với G là nhóm của hệ mã nêu trong ví dụ 4.14 còn g* là phép thế xác định như sau:

$$\forall x \in X \text{ thì } xg^* = \begin{cases} x & khi \quad x \le 1 \\ 2k+1 & khi \quad (x = 2k) \wedge (x > 1) \\ 2k & khi \quad (x = 2k+1) \wedge (x > 1) \end{cases} \qquad (4.7).$$

Theo ví dụ 4.14 thì nhóm các phép thế của hệ mật nêu trên cũng có tính phát tán, giả sử nhóm này là không nguyên thuỷ thì khi đó tồn tại block không tầm thường Y với hệ đầy đủ các block với dạng Yg. Không giảm tổng quát ta coi $0 \in Y$. Do Y không tầm thường nên tồn tại ít nhất $0 \neq x \in Y$.

Trước hết ta thấy rằng nếu $x,y \in Y$, từ tính phát tán ta có $\exists g$ sao cho xg=y lại do Y là block nên Y=Yg. Từ nhận định trên ta có nếu 0 và $x \in Y$ thì ix (mod 2^m) với mọi số nguyên i. Nếu x là số lẻ ta có ngay tập

$$Y \supseteq \{ix \pmod{2^m}: \text{ i nguyên}\} = X. \qquad (4.8).$$

Do Y là không tầm thường nên ta suy ra Y chỉ gồm những số chẵn.

Xét Yg*. Từ (4.7) rõ ràng 0g*=0 nên Yg*=Y, vậy $xg^* \in Y$. Nhưng cũng theo (4.7) thì xg* lại là một số lẻ nên mâu thuẫn với (4.8). Tóm lại Y không thể là block thực sự hay hệ mã của chúng ta có nhóm phép thế là nguyên thuỷ.

Với hệ mã nêu trong ví dụ trên ta dễ dàng nhận ra rằng rất không an toàn.

IV. MỘT SỐ ĐIỀU KIỆN ĐỦ ĐỂ NHÓM CÁC PHÉP THẾ CÓ TÍNH PHÁT TÁN VÀ NGUYÊN THUỶ

Điều kiện 4.16. *Nếu có hàm mã hoá $g \in E$ sao cho quỹ đạo với phần tử đầu là $x \in X$ là $\{x, xg, xg^2, ...\} = X$, và được gọi là phép thế có quỹ đạo cực đại, thì nhóm các phép mã hoá có tính phân tán.*

Chứng minh.

Từ giả thiết của điều kiện trên ta thấy $xG_g=X$ với G_g là nhóm cyclic sinh bởi g, mà $G_g \subseteq G$ nên ta có ngay $xG=X$.□

Mặc dù rằng chỉ cần tồn tại một hàm mã hoá có quỹ đạo cực đại là đảm bảo cho nhóm các phép thế có tính phát tán nhưng đặc tính này của các hàm mã hoá là một đặc tính tốt về mặt mật mã nên chúng ta nên đưa ra yêu cầu về các hàm mã hoá đó là:

Tiêu chuẩn 4.17. *Tập các hàm mã hoá E gồm các phép thế có quỹ đạo cực đại.*

Điều kiện 4.18. *Nếu G là 2-phát tán thì G cũng là nguyên thuỷ.*

Chứng minh.

Đầu tiên ta chứng tỏ G là phát tán.

Quả vậy, giả sử $x,y\neq 0$ bất kỳ, xét tập $\{0,x\}$. Do tính 2-phát tán nên $\exists g\in G$ sao cho

$$\{0,x\}g=\{x,y\} \qquad (4.9).$$

Từ đẳng thức (4.9) ta có $0g\neq 0$. Không giảm tổng quát coi có

$$0g=x \qquad (4.10).$$

Bây giờ $\forall z\in X$ bất kỳ, lại do tính 2-phát tán nên tồn tại $g_1\in G$ sao cho:

$$\{0,x\}g_1=\{x,z\} \qquad (4.11).$$

Nếu $0g_1=z$ thì $z\in 0G$, ngược lại ta có $0g_1=x$ và $xg_1=z$ vậy $0g_1^2=z$ nên $z\in 0G$. Tóm lại ta đã có $0G=X$.

Bây giờ nếu G không nguyên thuỷ tức là tồn tại block không tầm thường là Y, ta giả sử $x,y\in Y$. Xét $z\in X$ bất kỳ. Do tính 2-phát tán nên tồn tại $h\in G$ sao cho:

$$\{x,y\}h=\{x,z\} \qquad (4.12).$$

Đẳng thức (4.12) có nghĩa là $Yh\cap Y\neq\varnothing$ và do đó $Y=Yh$. Rõ ràng $[x,z\}\subseteq Yh=Y$ vậy ta có ngay $z\in Y$ hay $Y=X$. Mâu thuẫn này chứng tỏ rằng G là nguyên thuỷ.□

Kết quả trên chứng tỏ rằng tính 2-phát tán là mạnh hơn tính nguyên thuỷ. Do điều kiện thời gian và kiến thức chưa cho phép nên chúng tôi chưa có một khẳng định gì về điều ngược lại, tuy nhiên giữa tính phát tán và 2-phát tán thì chúng ta có được được khẳng định sau.

Kết quả 4.19. *Tính 2-phát tán mạnh hơn hẳn tính phát tán.*

Chứng minh.

Trong chứng minh điều kiện 4.18 chúng ta đã chỉ ra rằng nếu G 2-phát tán thì phát tán. Ngược lại, nhóm G nêu trong ví dụ 4.14 đã được khẳng định là phát tán tuy nhiên ta dễ dàng kiểm tra được rằng tập các số chẵn Y chính là một block không tầm thường của nhóm này và do điều kiện 4.18 ta có ngay G là không 2-phát tán.□

V. MỘT SỐ PHÂN TÍCH THÊM VỀ TÍNH T-PHÁT TÁN

Mặc dù chúng ta đã thu được một kết quả về tính mạnh hơn hẳn của tính 2-phát tán so với tính phát tán thế nhưng trong chú ý đưa ra về khái niệm này trong phần II chúng ta cũng thấy rằng mọi nhóm bất kỳ các phép thế đều có tính 2^m-phát tán điều này có nghĩa không thể có được một kết luận về tính mạnh hơn của tính (t+1)-phát tán so với tính t-phát tán một cách tổng quát. Có thể cố gắng để chứng minh thêm được một số kết quả tương tự cho một số giá trị t cụ thể nhưng có lẽ ý nghĩ của chúng cũng chưa được rõ ràng lắm nên trong phần này chúng tôi mở rộng thêm khái niệm t-phát tán để đưa ra được một số kết luận lý thuyết bổ ích hơn.

V.1. KHÁI NIỆM T-PHÁT TÁN MẠNH

Ký hiệu $\mathbf{X}_t$ là tập tất cả các chỉnh hợp chập t của 2^m phần tử của X và với mỗi $g \in G$ ta thấy rằng $\forall \{x_1,x_2,..,x_t\} \in \mathbf{X}_t$, do g là phép thế nên:

$$\{x_1g,x_2g,...,x_tg\} \in \mathbf{X}_t \quad (4.13).$$

Biểu thức (4.13) có nghĩa là từ $g:X \to X$ ta đã cảm sinh được $g^*:\mathbf{X}_t \to \mathbf{X}_t$ bởi công thức sau:

$$\{x_1,x_2,...,x_t\}g^*=\{x_1g,x_2g,...,x_tg\} \quad (4.14).$$

Hơn thế nữa, nếu $\{x_1,x_2,..,x_t\}\neq\{y_1,y_2,...,y_t\}$ thì hiển nhiên:

$$\{x_1g,x_2g,...,x_tg\}\neq\{y_1g,y_2g,...,y_tg\} \quad (4.15).$$

Biểu thức (4.15) có nghĩa là g* là phép thế trên $\mathbf{X}_t$.

Ký hiệu $G^*=\{g^*: g\in G\}$ ta dễ dàng chỉ ra rằng G* cũng là nhóm với phép nhân cảm sinh từ phép nhân trên G tức là:

$$g^*.h^*=(g.h)^* \quad (4.16).$$

Định nghĩa 4.20. *G được gọi là t-phát tán mạnh nếu $G^*=\{g^*: g\in G\}$ là phát tán trên $\mathbf{X}_t$.*

Qua định nghĩa trên ta thấy rằng nếu G là t-phát tán mạnh thì mỗi bộ t khối m bit đều là ảnh của mọi bộ t khối m bit như vậy nếu hệ mã có E=G là nhóm t-phát tán mạnh thì mỗi t khối m bit bất kỳ đều có thể là bản mã của một t khối m bit bất kỳ và đây chính là đặc trưng rất tốt cho tính an toàn của hệ mật. Chúng ta có thể đưa ra các thuộc tính t-phát tán mạnh để là các tiêu chuẩn cho một hên mã khối, nhưng trước hết ở đây ta sẽ chỉ ra các thuộc tính trên là tồn tại bằng kết quả sau.

Kết quả 4.21. *$\forall t: 0<t<2^m$ luôn tồn tại hệ mã khối có G là t-phát tán mạnh.*

Chứng minh.

Với mỗi bộ t phần tử có thứ tự $\{x_1,x_2,...,x_t\}\in\mathbf{X}_t$, ta ký hiệu $g_{(x_1x_2...x_t)}$ là phép thế trên X được xác định như sau:

$$\begin{aligned} x\, g_{(x_1x_2...x_t)} &= x_i \text{ với } x=1,2,...,t \\ x\, g_{(x_1x_2...x_t)} &= i \text{ với } x=x_i \\ x\, g_{(x_1x_2...x_t)} &= x \text{ trong các trường hợp còn lại.} \end{aligned} \quad (4.17).$$

xét hệ mã khối có $E=\{f_\alpha: \forall\alpha\in \mathbf{X}_t\}$.

Nếu ký hiệu $\alpha_0=\{1,2,...,t\}$ hiển nhiên ta có $\alpha_0 f_\alpha^*=\alpha$ do đó $\alpha_0E^*=\mathbf{X}_t$ vậy $\alpha_0G^*=\mathbf{X}_t$ với G là nhóm sinh bởi E tức là G là t-phát tán mạnh.□

Chú ý: Thực ra để có được kết luận nêu trong 4.21 chúng ta chỉ cần đưa ra hệ mã khối có E là tập toàn bộ các phép thế trên X. Tuy nhiên việc chứng minh của kết quả 4.21 thực chất còn bao gồm một minh hoạ cho sự tồn tại một hệ mã khối tồi có tính t-phát tán mạnh đó là hệ mã có E là tập các phép thế $g_{(x_1x_2\ldots x_t)}$.

V.2. MỘT SỐ TÍNH CHẤT

Từ các định nghĩa thì hiển nhiên ta có.

Tính chất 4.22. *Nếu G là t-phát tán mạnh thì nó cũng là t-phát tán.*□

Trong khi tại phần IV, với kết quả 4.19 chúng ta đã chỉ ra rằng tính 2-phát tán là mạnh hơn thực sự tính phát tán và đồng thời cũng cho thấy rằng mọi nhóm đều có tính 2^m-phát tán thì đối với tính t-phát tán mạnh chúng ta có tính chất sau.

Tính chất 4.23. *Nếu G là (t+1)-phát tán mạnh thì nó cũng t-phát tán mạnh.*

Chứng minh.

Xét bộ t phần tử có thứ tự $\{x_1,x_2,..,x_t\}$ bất kỳ. Lấy x_{t+1} khác với tất cả các x_i với i=1÷t, khi đó do G là (t+1)-phát tán mạnh nên tồn tại g∈G sao cho

$$\{1,2,...,t,t+1\}g^*=\{x_1,x_2,...,x_t,x_{t+1}\} \qquad (4.18)$$

Từ định nghĩa của g* ta có $ig=x_i$ với mọi i=1÷(t+1) và như vậy ta lại có

$$\{1,2,...,t\}g^*=\{x_1,x_2,...,x_t\} \qquad (4.19)$$

Hay G là t-phát tán mạnh.□.

Tính chất 4.24. *Nếu G là t-phát tán mạnh thì* $^{\#}G \geq {}^{\#}\mathbf{X}_t$.

Chứng minh.

Theo định nghĩa của t-phát tán mạnh thì với mỗi bộ có thứ tự $\{x_1,x_2,...,x_t\}\in\mathbf{X}_t$ luôn tồn tại g∈G sao cho có đẳng thức (4.19). Rõ ràng

nếu $\{x_1,x_2,...,x_t\}$ và $\{y_1,y_2,...,y_t\}$ là hai bộ t khác nhau thì các phép thế g và h tương ứng để thoả mãn đẳng thức (4.19) tức là.

$$\{1,2,...,t\}g^*=\{x_1,x_2,...,x_t\}$$
$$\{1,2,...,t\}h^*=\{y_1,y_2,...,y_t\}$$

cũng là khác nhau do đó ta đã có điều cần chứng minh.□

Một lần nữa cũng vì lý do thời gian và trình độ nên tại phần này chúng tôi vẫn còn để lại một số vấn đề như sau.

Thứ nhất ta thấy rằng một điều hiển nhiên là tính 1-phát tán và 1-phát tán mạnh chỉ là một, ngược lại với G chỉ gồm phép thế đơn vị là 2^m-phát tán nhưng không 2^m-phát tán mạnh. Với những $1<t<2^m$ còn một câu hỏi chưa giải quyết là>

Vấn đề 4.25. *Liệu có tồn tại $1<t<2^m$ sao cho tính t-phát tán mạnh là mạnh hơn hẳn tính t-phát tán?*.□

Lại xuất phát từ bất đẳng thức nêu trong tính chất 4.24. Đầu tiên với ví dụ 4.14 với hệ mật có tính phát tán đó là hệ có $E=\{f:X\rightarrow X\}$ với $xf=(x+1) \pmod{2^m}$, ta thấy rằng nhóm G sinh bởi E là nhóm cyclic có đúng 2^m phần tử khác nhau đó là f_i được xác định như sau.

$$xf_i=(x+i) \pmod{2^m}. \qquad (4.20)$$

Do đó tồn tại nhóm 1-phát tán mạnh có lực lương bằng lực lượng của $X=\mathbf{X}_t$. Mặt khác nhóm toàn bộ các phép thế là 2^m-phát tán mạnh mà nhóm này có $(2^m)!$ phần tử và cũng trùng với số phần tử của tập X_{2^m}. Đến đây một bài toán nữa có thể được đặt ra là.

Vấn đề 4.26. *Liệu có tồn tại những nhóm t-phát tán mạnh với $1<t<2^m$ để xảy ra dấu bằng trong bất đẳng thức nêu trong tính chất 4.24 hay không?*.□

Để có thể đưa ra những lời kết luận cuối cùng, sau đây ta cùng xem xét đến một kết quả liên quan đến vấn đề cần quan tâm nhất đó là tính chất

yêu cầu cao nhất về nhóm các hàm mã hoá mà ta đưa ra được trong bài này đó là 2^m-phát tán mạnh có đủ cho tính an toàn của hệ mật không.

Kết quả 4.27. *Tồn tại hệ mã khối không an toàn mà nhóm sinh bởi các hàm mã hoá của nó có tính 2^m-phát tán mạnh.*

Chứng minh.

Xét hệ mã khối có tập các hàm mã hoá $E=\{f_i: i=1\div(2^m-1)\}$ trong đó

$$xf_i=\begin{cases} i-1 & khi \quad x=i \\ i & khi \quad x=i-1 \\ x & (x\neq i)\wedge(x\neq i-1) \end{cases} \qquad (4.21)$$

(Trong công thức (4.21) cũng có sự đồng nhất dãy m bit với giá trị có biểu diễn nhị phân là dãy bit đó).

Hiển nhiên hệ mật trên là không an toàn vì chỉ có cùng lắm là hai khối rõ là có khối mã khác nó còn lại các khối mã đều chính là khối rõ.

Sau đây ta sẽ chỉ ra rằng nhóm G sinh bởi E là 2^m-phát tán mạnh.

Với i<j ta ký hiệu

$$g_{i,j}= f_jf_{j-1}..f_{i+1} \qquad (4.22)$$

Từ định nghĩa của các hàm f_i thì ta thấy rằng các hàm $g_{i,j}$ có tính chất sau:

$$jg_{i,j}=i \qquad (4.23)$$

$$kg_{i,j}=k \ \forall k<i \qquad (4.24)$$

Xét một hoán vị bất kỳ $\{x_0,x_1,..,x_{n-1}\}$ của tập n số $\{0,1,...,n-1\}$ (ở đây $n=2^m$).

Đầu tiên, giả sử $x_j=0$. Nếu j>0 ta xét ảnh của $\{0,1,...,n-1\}$ qua $g_{0,j}$ và giả sử là

$$\{0,1,...,n-1\}g^*_{0,j}=\{x_0^{\,1},x_1^{\,1},...,x_{n-1}^{\,1}\} \qquad (4.25)$$

Theo (4.23) thì $x_j^{\,1}=0$.

Tiếp theo, giả sử $x_k=1$ và tương ứng $x_k^1=r$. Hiển nhiên $r\geq 1$ và nếu $r>1$ ta xét ảnh của $\{x_0^1,x_1^1,...,x_{n-1}^1\}$ qua $g_{1,r}$ và giả sử là

$$\{x_0^1,x_1^1,...,x_{n-1}^1\}g^*_{1,r}=\{x_0^2,x_1^2,...,x_{n-1}^2\} \qquad (4.26)$$

Lại theo (4.23) ta có $x_k^1 g_{1,r}=1$ và theo (4.24) thì $x_j^1 g_{1,r}=0$. (4.27)

... Như vậy nếu lặp thuật toán trên đến bước thứ u thì dãy ảnh thu được là $\{x_0^u,x_1^u,...,x_{n-1}^u\}$ sẽ có tính chất sau: $x_i^u=x_i$ tại mọi vị trí i thoả mãn $x_i\leq i$. Như vậy dãy ảnh đến bước thứ n-1 là

$$\{x_0^{n-1},x_1^{n-1},...,x_{n-1}^{n-1}\}=\{x_0,x_1,...,x_{n-1}\} \qquad (4.28)$$

Đẳng thức (4.28) chứng tỏ $\{0,1,...,n-1\}g^*=\{x_0,x_1,...,x_{n-1}\}$ với g là tích của các $g_{i,j}$ được thực hiện trong n-1 bước trên và điều này có nghĩa G là 2^m-phát tán mạnh.□

Hệ quả 4.28. *Nhóm sinh của $\{f,f_1\}$* (trong đó f là phép thế đưa ra trong ví dụ 4.14 còn f_1 là phép thế đưa ra trong kết quả 4.27) *là 2^m-phát tán mạnh*.

Để chứng minh hệ quả này chúng ta chỉ cần chỉ ra rằng nhóm sinh của tập $\{f,f_1\}$ chứa E với E là tập các hàm mã hoá đưa ra trong kết quả 4.27 là đủ. Quả vậy ta luôn có $f_i=f^{n-i}f_1f^i$ $\forall i>0$ và cũng vẫn ký hiệu $n=2^m$.□

KẾT LUẬN

1. Mọi tính yếu của một hệ mã khối được xuất phát từ nguyên nhân nhóm sinh của tập các phép mã hoá của nó có một tính chất đặc biệt nào đó (chắc chắn là do nó không trùng với nhóm toàn bộ các phép thế) phải được đưa vào thành tiêu chuẩn để xây dựng thiết kế một hệ mã khối.

2. Trong nghiên cứu của chúng tôi, đặc biệt là hệ quả 4.28, cho thấy việc làm cho một hệ mã khối có nhóm sinh của tập các phép mã hoá E trùng với nhóm toàn bộ các phép thế là rất dễ dàng đó là chỉ cần bổ xung vào E hai hàm f và f_1 như đã trình bày trong hệ quả đó.

3. Có thể nhiều kết quả mà chúng tôi trình bày trong bài viết này chưa gắn bó gì với những điều chúng ta quan tâm nhất (thể hiện sự liên quan

đến tính an toàn của hệ mã khối) nhưng qua nó ta có thể thấy được những khai thác về mặt cấu trúc của nhóm các phép thế.

4. Cuối cùng chúng tôi cũng thừa nhận rằng các kết quả mới chỉ dừng ở mức độ lý thuyết, để có thể vươn tới các kết luận thực tế chúng ta cần phải đầu tư vào việc xem xét các thuộc tính đã được nghiên cứu vào những mô hình mã khối quen thuộc như DES, FEAL, IDEA, GOST,...

CHƯƠNG 5: KHẢO SÁT CÁC ĐẶC TRƯNG CỦA MÃ KHỐI THEO QUAN ĐIỂM XÍCH MARKOV

Rất nhiều mã khối, mà điển hình là DES và IDEA, có thuật toán mã hoá tiến hành lặp đi lặp lại một hàm (thường được gọi là hàm vòng). Hai phương pháp tấn công rất nổi tiếng đối với loại mã khối này là tấn công lượng sai và tấn công tuyến tính. Trong tấn công lượng sai, người ta đã liên hệ lượng sai của hai khối vào với lượng sai của các khối ra tương ứng của hàm vòng để thiết lập công thức lượng sai. Trên cơ sở các công thức lượng sai này người ta chọn một tập các cặp rõ, thu các cặp mã tương ứng rồi tiến hành tìm các bít khoá của người lập mã. Từ các bít khoá này, có thể dùng phương pháp thử và sai để tìm các bít khoá còn lại. Trong tấn công tuyến tính, các công thức tuyến tính liên hệ các bít khoá với các bít rõ và các bit mã tương ứng được thiết lập (đúng với một xác suất nào đó) từ mối liên hệ tuyến tính của các bít vào và các bít ra tương ứng của hàm vòng. Trên cơ sở các công thức tuyến tính này, khi thu được một tập các cặp rõ/ mã tương ứng, người ta tìm các bít khoá của người lập mã.

Hiệu quả của hai phương pháp này được thể hiện trên các phương diện sau đây: tập các cặp rõ, và các cặp mã tương ứng (trong tấn công lượng sai), tập các cặp rõ/ mã tương ứng (trong tấn công tuyến tính) có độ lớn là bao nhiêu thì xác suất thành công của người mã thám đủ cao? Khi có tập này rồi thì thời gian tiến hành có thực tế hay không? Khả năng thực tế trong việc thu thập tập hợp này?

Đối với người lập mã, các câu hỏi thường được đặt ra như sau: Hàm vòng phải được thiết kế như thế nào để các công thức ở trên đúng với xác suất bé? Số vòng lặp tối thiểu phải là bao nhiêu để khiến cho lực lương cần thiết của tập rõ/mã làm nản lòng các nhà mã thám?

Việc nghiên cứu mã khối trên quan điểm xích Markov đã giúp các nhà mật mã trả lời các câu hỏi đó trên những điểm lớn, khái quát.

I. MỘT SỐ CƠ SỞ TOÁN HỌC

I.1. XÍCH MARKOV HỮU HẠN

Một dãy các biến ngẫu nhiên rời rạc $v_0,v_1,\ldots,v_r$ được gọi là một *xích Markov* nếu với $0 \le i < r$ (có thể $r=\infty$), có

$$P\left(v_{i+1} = \beta_{i+1} \middle| v_i = \beta_i, v_{i-1}, \beta_{i-1}, \ldots, v_0 = \beta_0\right) = P\left(v_{i+1} = \beta_{i+1} \middle| v_i = \beta_i\right)$$

Một xích Markov được gọi là *thuần nhất* nếu $P(v_{i+1} = \beta | v_i = \alpha)$ không phụ thuộc vào i đối với tất cả α, β. Trong chương này, khi không nói gì thêm thì ta chỉ xét các xích Markov thuần nhất. Nếu các biến ngẫu nhiên v_i cùng nhận giá trị trên một tập hữu hạn (ký hiệu số phần tử của tập này là N) thì ta có xích Markov hữu hạn.

Đại lượng

$$p_{ij} = P(v_{n+1} = j | v_n = i)$$

được gọi là xác suất chuyển sau một bước. Đại lượng sau đây được gọi là xác suất chuyển sau k bước

$$p_{ij}^{(k)} = P(v_{n+k} = j | v_n = i) = P(v_{k+1} = j | v_1 = i)$$

Rõ ràng $p_{ij}^1 = p_{ij}$. Ta quy ước

$$p_{ij}^{(0)} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

Ma trận $P=(p_{ij})$, $P^{(k)} = \left(p_{ij}^{(k)}\right)$ lần lượt được gọi là ma trận xác suất chuyển sau 1 bước, ma trận xác suất chuyển sau k bước.

Phân phối của xích tại thời điểm n là:

$$p_j^n = P(v_n = j), \quad n = 0,1,...$$

Đặt $\Pi^{(n)} = \left(p_j^{(n)}\right)$. Khi đó $\Pi = \Pi^{(0)}$ được gọi là *phân phối ban đầu* của xích Markov hữu hạn, thuần nhất.

Phân phối ban đầu được gọi là *dừng* nếu $\Pi^{(n)}$ không phụ thuộc vào n, tức là $\Pi = \Pi^{(n)}$, hay $\Pi=\Pi P$.

Mô hình của một xích Markov rời rạc và thuần nhất có tập trạng thái hữu han $E =\{1,2,\ldots,N\}$ là bộ ba (v_n,Π,P), trong đó (v_n) là dãy các đại lượng ngẫu nhiên rời rạc nhận giá trị trên E, Π là phân phối ban đầu, P là ma trận xác suất chuyển.

Định lý 5.1. Giả sử (v_n,Π,P) là mô hình của một xích Markov rời rạc và thuần nhất có tập trạng thái hữu hạn $E =\{1,2,\ldots,N\}$.

(i) Nếu P chính quy theo nghĩa sau: tồn tại n_0 sao cho

$$\min\ p_{ij}^{(n_0)} > 0 \qquad (5.1)$$

thì tồn tại các số $\pi_{1,\ldots,}\pi_N$ sao cho

$$\pi_j > 0, \ \sum_{j \in E} \pi_j = 1 \qquad (5.2)$$

và với mỗi j∈E

$$\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j \ . \qquad (5.3)$$

(ii) Ngược lại, nếu tồn tại các số $\pi_{1,\ldots,}\pi_N$ thoả mãn các điều kiện (5.2) và (5.3) thì sẽ tồn tại n_0 thoả mãn (5.1).

(iii) Các số $\pi_{1,\ldots,}\pi_N$ là nghiệm của hệ phương trình

$$x_j = \sum x_k p_{kj}, j \in E \qquad (5.4)$$

và đó là nghiệm duy nhất thoả mãn điều kiện

$$x_j \geq 0, \forall j \in E \qquad ; \qquad \sum_{j\in E} x_j = 1$$

nếu (5.1) được thực hiện.

Nghiệm không âm ($\pi_{1,\ldots,}\pi_N$) của phương trình (5.4) sao cho $\sum \pi_j = 1$ được gọi là *phân phối dừng* (hay bất biến) của xích Markov với ma trận xác suất chuyển $P=(p_{i\,j})$.

Với phân phối dừng, ta có $\pi_j^n = P(v_n = j) = \pi_j$, tức là $v_1,\ldots,v_n,\ldots$ có phân phối xác suất như nhau.

Ta nói rằng xích Markov có phân phối giới hạn, nếu ∀j=1,2,…,N tồn tại các giới hạn $\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j$ không phụ thuộc vào i và thoả mãn các điều kiện $\pi_j>0$, $\sum_{j\in E} \pi_j = 1$. Trong trường hợp đó ta gọi ($\pi_{1,\ldots,}\pi_N$) là phân phối giới hạn.

Định lý 5.2. Nếu tồn tại phân phối giới hạn, thì đó là phân phối dừng duy nhất.

Ta nói rằng trạng thái i *liên thông* với trạng thái j nếu tồn tại các số nguyên r_1,r_2 sao cho $p_{ij}^{(r_1)} > 0$ và $p_{ji}^{(r_2)} > 0$, đồng thời ký hiệu i ↔j. Có thể thấy ↔ là quan hệ tương đương.

Nếu xích Markov chỉ có 1 lớp tương đương thì nó được gọi là *bất khả quy*. Trạng thái i có chu kỳ $d_i=\min\{ k: p_{ii}^{(k)} > 0 \}$ (Nếu $p_{ii}^{(k)}=0$ đối với tất cả n ≥1 thì đặt $d_i=0$) và được nói là *không có chu kỳ* nếu $d_i=1$. Rõ ràng nếu i liên thông với j thì $d_i=d_j$.

Chu kỳ d của xích Markov được định nghĩa là *ước số chung lớn nhất* của $d_1,d_2,\ldots,d_N$ và xích được nói là *không có chu kỳ* nếu d=1. Đối với xích bất khả quy thì $d= d_1=d_2=\ldots=d_N$.

Xích Markov hữu hạn, bất khả quy và không có chu kỳ được gọi là *ergodic*.

Định lý 5.3. Nếu xích Markov là ergodic, thì tồn tại phân phối duy nhất $\Pi = (\pi_{1,\ldots,}\pi_N)$ sao cho

$$\pi_j = \lim_{n\to\infty} p_{ij}^{(n)}. \qquad (5.5)$$

Ta có $\sum_j p_{ij} = 1$ theo định nghĩa, và ngoài ra nếu $\sum_i p_{ij} = 1$ thì P được nói là *ngẫu nhiên kép*.

Từ nay về sau, trong nhiều trường hợp, thay cho nói ' dãy … là ergodic' thì ta có thể nói 'P là ergodic' cho gọn.

Định lý 5.4. Nếu P (xích) là ergodic và ngẫu nhiên kép thì phân phối giới hạn cho nó là phân phối đều $\Pi = (1/N,\ldots,1/N)$.

Đặt

$$f_{ij}^{(n)} = P\{\nu_n = j, \nu_1 \neq j, \ldots, \nu_{n-1} \neq j \,|\, \nu_0 = i\}, j \in E.$$

Khi đó i được gọi là trạng thái hồi quy nếu $f_{ii}=1$; và được gọi là trạng thái không hồi quy nếu $f_{ii}<1$.

Giả sử i là trạng thái hồi quy. Ta nói i là trạng thái dương nếu $\mu_i < \infty$, i là trạng thái không nếu $\mu_i = \infty$, trong đó $\mu_i = \sum_{n=0}^{\infty} n f_{ii}^{(n)}$.

Định lý 5.5. Nếu (ν_n) là xích Markov bất khả quy, không có chu kỳ, có hữu hạn trạng thái thì tất cả các trạng thái là hồi quy dương.

Định lý 5.6. Giả sử (ν_n) là xích Markov có hữu hạn trạng thái. Khi đó các điều kiện sau là tương đương:

(i) (ν_n) là bất khả quy và không có chu kỳ.

(ii) (ν_n) là bất khả quy và không có chu kỳ, tất cả các trạng thái là hồi quy dương.

(iii) Tồn tại các giới hạn

$$\pi_j = \lim_{n\to\infty} p_{ij}^{(n)} \quad sao \quad cho \quad \pi_j > 0, \forall j \in E; \quad \sum_{j\in E} \pi_j = 1.$$

(iv) Tồn tại n_0 sao cho với mọi $n \geq n_0$ thì $\min_{i,j} p_{ij}^{(n)} > 0$.

II.2. ĐỒ THỊ NGẪU NHIÊN

Nhiều tác giả đã nhận xét rằng, có thể xem ma trận chuyển P như ma trận kề của đồ thị có hướng G=(V, E), trong đó $V=\{ v_1, v_2, \ldots, v_N\}$ là tập đỉnh và

có cạnh định hướng từ v_i tới v_j nếu và chỉ nếu $p_{i\,j} >0$. Người ta gọi G là đồ thị cơ sở của P.

Đồ thị có hướng G được gọi là có *liên kết mạnh* nếu với tất cả v_i , v_j luôn tồn tại đường dẫn có hướng từ v_i tới v_j. Khi đó P là bất khả quy nếu và chỉ nếu G là liên kết mạnh.

Chú ý rằng nếu G có *vòng* (có cạnh từ v_i tới chính nó, hay $p_{i\,i}>0$) thì P không có chu kỳ.

Ta sẽ nói rằng *hầu hết* các đồ thị N đỉnh và m cạnh có tính chất nhất định nếu tỷ lệ các đồ thị có tính chất này dần đến 1 khi $N\to\infty$. Người ta chỉ ra rằng hầu hết các đồ thị có hướng G với N đỉnh và m cạnh có vòng khi $m=N*\gamma_N$, $\gamma_N \to\infty$. Hơn nữa, Palásti [11] đã chỉ ra rằng hầu hết các đồ thị có hướng G với N đỉnh và m cạnh là liên kết mạnh khi $m=N(\log N+\gamma_N)$, $\gamma_N \to\infty$. Ở đây, γ_N là hàm bất kỳ tiến đến ∞ khi $N\to\infty$.

Định lý 5.7(Palásti)**.** Cho $m=N(\log N+\gamma_N)$, với $\gamma_N \to\infty$ Khi đó hầu hết các đồ thị có hướng G vừa là không có chu kỳ, vừa liên kết mạnh.

Chứng minh. Với hai biến cố bất kỳ α_1, α_2 ta có:

$$Pr(\alpha_1+ \alpha_2)=Pr(\alpha_1)+ Pr(\alpha_2)-Pr(\alpha_1\ \alpha_2)\le 1;$$

do đó

$$Pr(\alpha_1\ \alpha_2) \ge Pr(\alpha_1)+ Pr(\alpha_2) - 1.$$

Nếu α_1 =" G không có chu kỳ", α_2 = " G liên kết mạnh" thì với giả thiết nói trên, $Pr(\alpha_1)$, $Pr(\alpha_2)$ dần đến 1 khi $N\to\infty$. Từ đó $Pr(\alpha_1\ \alpha_2) \to 1$ khi $N\to\infty$.

Giả thiết rằng với mỗi N và m, đồ thị được chọn ngẫu nhiên đều. Ta có hệ quả của định lý trên.

Hệ quả 5.8. Cho G là đồ thị có hướng với m cạnh, N đỉnh. Cho

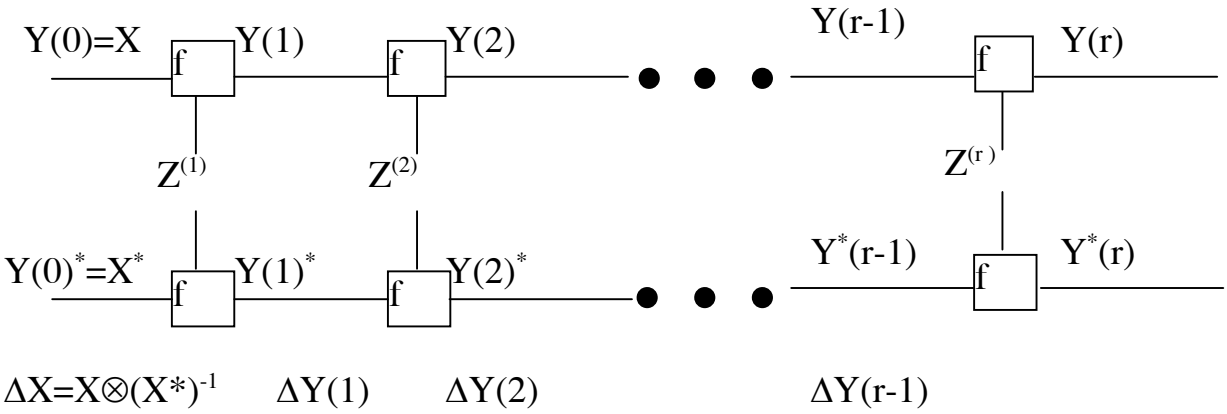
$$Pr\{\ m \ge N(\log N+\gamma_N)\ \} \to 1, \text{ trong đó } \gamma_N \to\infty. \text{ Thế thì khi } N\to\infty$$

$$Pr\{\ G \text{ không có chu kỳ và liên kết mạnh}\} \to 1.$$

II. MẬT MÃ MARKOV VÀ THÁM LƯỢNG SAI

II.1. MẬT MÃ MARKOV

Ta xét việc mã hoá một cặp các bản rõ khác nhau bởi mật mã lặp r vòng như được chỉ ra trong hình sau đây:



Trong hình này *lượng sai* $\Delta Y(i)$ giữa hai khối m bít Y(i) và Y*(i) được xác định như

$$\Delta Y(i) = Y(i) \otimes Y^*(i),$$

trong đó $\otimes$ ký hiệu phép toán nhóm đặc biệt trên tập các khối m bít và $Y^*(i)^{-1}$ chỉ phần tử nghịch đảo của Y*(i) trong nhóm.

Ta nhận được

dãy các lượng sai $\quad \Delta Y(0),\ \Delta Y(1), \ldots, \Delta Y(r)$

với $Y(0) = X$ và $Y^*(0) = X^*$ là cặp rõ sao cho $\Delta Y(0) = \Delta X$, và Y(i), Y*(i) ($0<i<r$) lả đầu ra của vòng thứ i, đồng thời cũng là đầu vào của vòng thứ (i+1). Các khoá con cho vòng thứ i được ký hiệu là $Z^{(i)}$ và f là hàm vòng sao cho $Y(i) = f(Y(i-1), Z^{(i)})$.

Sau này ta sẽ luôn luôn giả thiết rằng $X \neq X^*$ vì khi $X=X^*$, tất cả $\Delta Y(i)$ sẽ bằng phần tử trung hoà e của nhóm. Đây là trường hợp mà thám lượng sai không quan tâm. Như vậy, $\Delta Y(i) \in F_2^m - \{e\}$. Ta cũng giả thiết rằng các khoá con được dùng trong mỗi vòng của mật mã lặp là độc lập thống kê và có phân phối đều, tuy thế trong thực tế các kết quả này ứng dụng tốt cho các mật mã lặp dù rằng ở đây các khoá con vòng được sinh bởi thuật toán tác động lên khoá bí mật.

II.1.1. Định nghĩa.

Định nghĩa 5.9. Mật mã lặp với hàm vòng f là *mật mã Markov* *nếu tồn tại phép toán nhóm $\otimes$ đối với các lượng sai xác định sao cho, với mọi α ($\alpha \neq e$) và β ($\beta \neq e$), đại lượng*

$$P(\Delta Y = \beta \mid \Delta X = \alpha, X = \gamma),$$

trong đó Y = f(X, Z) và Y = f(X*, Z), là độc lập với γ khi khoá con Z là ngẫu nhiên phân phối đều; hoặc tương đương, nếu*

$$P(\Delta Y = \beta \mid \Delta X = \alpha, X = \gamma) = P(\Delta Y(1) = \beta \mid \Delta X = \alpha)$$

đối với mọi γ khi khoá con Z là ngẫu nhiên phân phối đều.

II.1.2. Các tính chất.

Định lý 5.10. *Nếu mật mã lặp r vòng là mật mã Markov và r khoá vòng là ngẫu nhiên, độc lập, phân phối đều, thì dãy các lượng sai ΔX = ΔY(0), ΔY(1), ..., ΔY(r) là xích Markov thuần nhất. Hơn nữa, xích này là dừng nếu ΔX phân phối đều trên các phần tử không trung hoà của nhóm.*

Chứng minh. Để chứng minh rằng dãy $\Delta X = \Delta Y(0), \Delta Y(1), \ldots, \Delta Y(r)$ là xích Markov ta sẽ chỉ ra rằng

$$P(\Delta Y(2) = \beta_2 \mid \Delta Y(1) = \beta_1, \Delta X = \alpha) = P(\Delta Y(2) = \beta_2 \mid \Delta Y(1) = \beta_1).$$

Muốn thế, chú ý rằng

$$\begin{aligned}
&P(\Delta Y(2) = \beta_2 \mid \Delta Y(1) = \beta_1, \Delta X = \alpha) \\
&\quad = \sum_\gamma P(Y(1)= \gamma, \Delta Y(2) = \beta_2 \mid \Delta Y(1) = \beta_1, \Delta X = \alpha) \\
&\quad = \sum_\gamma P(Y(1)= \gamma \mid \Delta Y(1) = \beta_1, \Delta X = \alpha) \\
&\qquad \times P(\Delta Y(2) = \beta_2 \mid \Delta Y(1) = \beta_1, Y(1)= \gamma, \Delta X = \alpha) \\
&\quad = \sum_\gamma P(Y(1)= \gamma \mid \Delta Y(1) = \beta_1, \Delta X = \alpha) P(\Delta Y(2) = \beta_2 \mid \Delta Y(1) = \beta_1, \\
&\quad Y(1)= \gamma) \\
&\quad = \sum_\gamma P(Y(1)= \gamma \mid \Delta Y(1) = \beta_1, \Delta X = \alpha) P(\Delta Y(2) = \beta_2 \mid \Delta Y(1) = \beta_1) \\
&\quad = P(\Delta Y(2) = \beta_2 \mid \Delta Y(1) = \beta_1),
\end{aligned}$$

trong đó đẳng thức thứ 3 có từ việc Y(1) cùng với ΔY(1) xác định cả Y(1) và Y(1)* sao cho ΔY(2) không còn phụ thuộc vào ΔX nữa khi Y(1) và ΔY(1) được chỉ ra. Vì cùng một hàm vòng được sử dụng trong mỗi vòng nên xích Markov là thuần nhất. Đối với khoá bất kỳ Z=z, hàm vòng f(. , z) là song ánh từ tập các bản rõ vào tập các bản mã. Song ánh này dẫn đến một song ánh từ cặp các bản rõ khác nhau (X,X*) đến cặp các bản mã khác nhau (Y,Y*) = (f(X,z), f(X*,z)). Sự kiện X và ΔX (≠e) độc lập và phân phối đều kéo theo rằng (X,X*) phân phối đều trên cặp các bản rõ khác nhau. Như vậy, (Y,Y*) cũng phân phối đều trên cặp các bản mã khác nhau và vì thế ΔY(≠e) cũng phân phối đều. Như vậy, phân phối đều là phân phối dừng đối với xích Markov này.

Ví dụ. 1) Mã khối DES là mật mã Markov khi định nghĩa lượng sai $\Delta X = X \otimes (X^*)^{-1} = X \oplus X^*$, trong đó ⊕ chỉ phép XOR bít.

2) Với định nghĩa lượng sai $\Delta X = X \oplus X^*$ người ta có thể chỉ ra rằng các mật mã khối LOKI, FEALvà các mật mã DES-like, REDOC cũng là mật mã Markov.

Định lý 5.11. Nếu hàm vòng của mật mã lặp có dạng

$$f(X, Z) = g(X \otimes Z_A, Z_B),$$

trong đó $\otimes$ là phép toán nhóm trên F_2^m và hàm $g(., Z_B)$ khả nghịch đối với mọi Z_B, thì mật mã lặp này là mật mã Markov với định nghĩa lượng sai là $\Delta X = X \otimes (X^*)^{-1}$

Chú ý. Từ định lý này suy ra rằng, mật mã khối IDEA là mật mã Markov.

Chứng minh định lý. Cho $S = X \otimes Z_A$ và $Y = g(S, Z_B)$, thì $S^* = X^* \otimes Z_A$ và $Y^* = g(S^*, Z_B)$. Như vậy,

$$\Delta S = S \otimes (S^*)^{-1} = (X \otimes Z_A) \otimes (Z_A^{-1} \otimes (X^*)^{-1}) = \Delta X.$$

Vì

$$\Delta Y = g(S, Z_B) \otimes (g(S, Z_B))^{-1} = g(S, Z_B) \otimes (g((\Delta S)^{-1} \otimes S, Z_B))^{-1},$$

suy ra rằng ΔY không phụ thuộc nữa vào X khi ΔS và S được chỉ ra. Như vậy, đại lượng

$$\begin{aligned}
&P(\Delta Y = \beta \mid \Delta X = \alpha, X = \gamma) \\
&= P(\Delta Y = \beta \mid \Delta S = \alpha, X = \gamma) \\
&= \sum_{\lambda} P(\Delta Y = \beta, S = \lambda \mid \Delta S = \alpha, X = \gamma) \\
&= \sum_{\lambda} P(\Delta Y = \beta \mid \Delta S = \alpha, X = \gamma, S = \lambda)\, P(S = \lambda \mid \Delta S = \alpha, X = \gamma) \\
&= \sum_{\lambda} P(\Delta Y = \beta \mid \Delta S = \alpha, S = \lambda)\, P(Z_A = \lambda \otimes \gamma^{-1}) \\
&= 2^{-m} \sum_{\lambda} P(\Delta Y = \beta \mid \Delta S = \alpha, S = \lambda),
\end{aligned}$$

độc lập với γ. Ở đây ta đã sử dụng điều kiện $Z = (Z_A, Z_B)$ ngẫu nhiên phân phối đều, độc lập với X và công thức

$$\begin{aligned}
&P(S = \lambda \mid \Delta S = \alpha, X = \gamma) = P(X \otimes Z_A = \lambda \mid \Delta X = \alpha, X = \gamma) \\
&= P(Z_A = \lambda \otimes X^{-1} \mid \Delta X = \alpha, X = \gamma) = P(Z_A = \lambda \otimes X^{-1} \mid X = \gamma) \\
&= P(Z_A = \lambda \otimes \gamma^{-1}).
\end{aligned}$$

Đối với mật mã Markov bất kỳ, cho Π là ma trận xác suất chuyển của xích Markov thuần nhất $\Delta X = \Delta Y(0), \Delta Y(1), \ldots, \Delta Y(r)$. Phần tử (i,j)

trong Π là $P(\Delta Y(1) = \alpha_j \mid \Delta X = \alpha_i)$, trong đó $\alpha_1,\alpha_2,\ldots, \alpha_M$ là một cách sắp thứ tự nào đó M giá trị của ΔX và $M = 2^m-1$ đối với mật mã m-bít. Khi đó, với mỗi $r \geq 1$,

$$\Pi^r = \left[p_{ij}^{(r)}\right] = [\ P(\Delta Y(r) = \alpha_j \mid \Delta X = \alpha_i). \qquad (5.6)$$

Chú ý rằng mỗi dòng của Π là một phân phối xác suất do đó tổng các phần tử trong mỗi dòng là 1. Từ định lý 5.10 suy ra rằng phân phối đều là phân phối dừng. Như vậy, với mỗi j ta có

$$\frac{1}{2^m-1} = \sum_{i=1}^{2^m-1} p_{ij}\ \frac{1}{2^m-1} = \frac{1}{2^m-1}\sum_{i=1}^{2^m-1} p_{ij}$$

do đó mỗi cột của Π cũng có tổng bằng 1. Vậy,

Định lý 5.12. *Ma trận chuyển của mật mã Markov là ngẫu nhiên kép, nghĩa là mỗi dòng có tổng là 1 và mỗi cột có tổng là 1.*

II.1.3. Ánh xạ lượng sai đều

Ánh xạ F được gọi là lượng sai δ-đều nếu mỗi phần tử ở bảng XOR của F (trừ phần tử ở dòng, cột đầu tiên) có giá trị lớn nhất là δ.

Vì mỗi phần tử trong bảng XOR là số chẵn nên ánh xạ lượng sai 2-đều có bảng XOR chỉ chứa các phần tử 0 và 2, và ma trận chuyển tương ứng có chính xác 2^{n-1} phần tử khác 0 trên mỗi dòng. Số phần tử khác 0 của nó là $(2^n-1)2^{n-1} = N(N+1)/2$.

Ví dụ. Ánh xạ $\rho : GF(2^3) \rightarrow GF(2^3)$ như sau: $\rho(x) = x^3 \bmod f(x)$, với $f(x) = x^3+x+1$ là đa thức bất khả quy trên GF(2). Khi đó ρ thực hiện tương ứng 0-0, 1-1, 2-3, 3-4, 4-5, 5-6, 6-7, 7-2. Đó là lượng sai 2-đều. Bảng XOR cho ρ và ma trận P tương ứng là

	0	1	2	3	4	5	6	7
0	8	0	0	0	0	0	0	0
1	0	2	0	2	0	2	0	2
2	0	0	2	2	2	2	0	0
3	0	2	2	0	2	0	0	2
4	0	0	0	0	2	2	2	2
5	0	2	0	2	2	0	2	0
6	0	0	2	2	0	0	2	2
7	0	2	2	0	0	2	2	0

$$P = (1/8)* \begin{bmatrix} 2 & 0 & 2 & 0 & 2 & 0 & 2 \\ 0 & 2 & 2 & 2 & 2 & 0 & 0 \\ 2 & 2 & 0 & 2 & 0 & 0 & 2 \\ 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 & 0 & 2 & 0 \\ 0 & 2 & 2 & 0 & 0 & 2 & 2 \\ 2 & 2 & 0 & 0 & 2 & 2 & 0 \end{bmatrix}$$

Chú ý rằng P nhận được từ bảng XOR bằng cách xoá dòng đầu và cột đầu rồi chia cho 8. P không có chu kỳ vì $p_{11}>0$.

Ma trận P kết hợp với ánh xạ lượng sai 2-đều có $(2^n-1)2^{n-1} = N(N+1)/2 > N^2/2$ phần tử khác 0. Định lý Palásti nói rằng ma trận chuyển với ít nhất N*log N cạnh khác 0 được phân phối ngẫu nhiên là ergodic vơí xác suất cao. Ta đưa ra giả thiết sau:

Mệnh đề 5.13. *Cho F: $Z_2^n \to Z_2^n$ là hàm song ánh, lượng sai 2-đều. Khi đó ma trận chuyển P rút ra từ F được giả thiết là ergodic.*

Kết quả thực nghiệm cho thấy: từ 40320 ánh xạ song ánh F: $Z_2^n \to Z_2^n$, có 10725 là lượng sai 2-đều, mỗi cái này có ma trận chuyển ergodic. Chỉ có 7 ma trận chuyển P khác nhau, với 1536 ánh xạ F thực hiện cùng ma trận chuyển P (chú ý rằng n=3 là số nhỏ nhất để ánh xạ song ánh lượng sai 2-đều tồn tại).

II.2. THÁM LƯỢNG SAI

II.2.1. Các lượng sai vòng

Thám lượng sai, do Biham và Shamir giới thiệu, là tấn công bản rõ lựa chọn đối với mật mã lặp. Đó là một trong những tấn công tốt nhất đối với mật mã lặp. Thám lượng sai khai thác chỗ yếu của mật mã lặp là hàm vòng f đơn giản và nó thường là yếu về mật mã theo nghĩa sau: đối với $Y(i) = f(Y(i-1),Z^{(i)})$ và $Y^*(i) = f(Y^*(i-1),Z^{(i)})$ nếu người ta biết một hoặc hơn các bộ ba $(\Delta Y(i-1), Y(i), Y^*(i))$ thì có thể xác định khoá vòng $Z^{(i)}$. Như vậy, nếu cặp bản mã đã biết và lượng sai của cặp đầu vào của vòng cuối cùng bằng cách nào đó cũng biết, thì thường là có thể xác định một phần chính khoá con của vòng cuối cùng. Trong thám lượng sai, điều này đạt được nhờ chọn các cặp rõ (X,X^*) có lượng sai cho trước α sao cho lượng sai $\Delta Y(i-1)$ của cặp đầu vào đối với vòng cuối cùng sẽ nhận một giá trị đặc biệt β với xác suất cao. Dựa trên ý tưởng này, ta có định nghĩa sau.

Định nghĩa 5.14. *Lượng sai i vòng là cặp (α,β), trong đó α là lượng sai của một cặp bản rõ khác nhau X và X*, còn β là lượng sai có thể đối với*

các đầu ra của vòng thứ i là Y(i) và Y(i). Xác suất của lượng sai i vòng (α,β) là xác suất có điều kiện để β là lượng sai $\Delta Y(i)$ của cặp bản mã sau i vòng đã cho và nếu cặp bản rõ (X , X*) có lượng sai $\Delta X = \alpha$ khi bản rõ X và các khoá con Z(1),...,Z(i) là độc lập và phân phối đều. Ta ký hiệu xác suất lượng sai này là $P(\Delta Y(i) = \beta / \Delta X = \alpha)$.*

Từ (5.6) ta thấy rằng đối với mật mã Markov , xác suất của lượng sai i vòng chính là phần tử (α,β) trong ma trận chuyển $\Pi^{(i)}$.

II.2.2. Tấn công phân tích lượng sai

Thủ tục cơ bản của tấn công phân tích lượng sai trên mật mã lặp r vòng có thể được tóm tắt như sau:

1) Tìm lượng sai (r-1) vòng (α,β) sao cho $P(\Delta Y(r-1) = \beta / \Delta X = \alpha)$ đạt cực đại hoặc gần cực đại.

2) Chọn bản rõ X ngẫu nhiên đều và tính X* sao cho $\Delta X = \alpha$. Làm sao người ta chấp nhận mã X và X* bởi khoá đúng Z. Từ các bản mã thu được Y(r) và Y*(r), tìm mỗi giá trị có thể của khoá con $Z^{(r)}$ của vòng cuối cùng tương ứng với lượng sai cho trước $\Delta Y(r-1) = \beta$. Cộng 1 vào bộ đếm số lần xuất hiện của mỗi giá trị có thể của khoá con $Z^{(r)}$.

3) Lặp lại 2) cho đến khi một hoặc hơn các giá trị có thể của khoá con $Z^{(r)}$ được đếm nhiều hơn một cách đáng kể so với các giá trị khác. Chấp nhận giá trị được đếm nhiều nhất này, hoặc một tập nhỏ các giá trị như thế, như quyết định của nhà mã thám đối với khoá đúng $Z^{(r)}$.

Chú ý. Nếu biết nhiều lượng sai r vòng có xác suất cao, thì phân tích lượng sai có thể được làm hiệu quả hơn nhiều theo nghĩa sau đây. Cho Ω là tập các lượng sai bản rõ đã biết trước đối với người tấn công sao cho với mỗi $\alpha \in \Omega$, tồn tại lượng sai (r-1) vòng (α,β_α) với xác suất cao. Tấn công lượng sai dựa trên hiểu biết về Ω có thể được thực hiện như sau:

Với mỗi cặp (đã biết hoặc lựa chọn) các cặp rõ/mã (X,Y) và (X*,Y*), nếu lượng sai $\Delta X = X \otimes (X^*)^{-1} = \alpha$ thuộc về tập Ω, thì tìm từng giá trị có thể của khoá con $Z^{(r)}$ của vòng cuối cùng từ hai bản mã tương ứng và lượng sai biết trước $\Delta Y(r-1) = \beta\alpha$. Cộng 1 vào bộ đếm số lần xuất hiện của mỗi giá trị như thế. Lặp lại bước trên cho mỗi cặp của các cặp rõ/mã như thế. Nếu một vài giá trị có thể của khoá con $Z^{(r)}$ được đếm nhiều hơn đáng kể các giá trị khác, thì những giá trị này được chọn làm ứng cử viên cho khoá con đúng $Z^{(r)}$ và có thể được kiểm tra bởi những test bổ sung khác để thu được quyết định mã thám đối với khoá con đúng $Z^{(r)}$.

Giả sử rằng tấn công bản rõ lựa chọn nhờ sử dụng một lượng sai cần T cặp mã hoá và giả sử tập Ω chứa N phần tử và N lượng sai tương ứng có gần

như cùng một xác suất. Khi đó tấn công bản rõ lựa chọn mô tả ở trên chỉ cần khoảng $T/\sqrt{N}$ bản rõ lựa chọn vì người ta có thể chọn $\sqrt{N}$ bản rõ theo cách sao cho mỗi cặp bản rõ này có lượng sai thuộc về tập Ω khiến cho $\sqrt{N}$ bản rõ sản sinh N lượng sai có ích cho tấn công.

II.2.3. Giả thiết tương đương ngẫu nhiên

Trong tấn công phân tích lượng sai, tất cả các khoá con là cố định, chỉ có bản rõ là có thể chọn ngẫu nhiên. Tuy nhiên, trong tính toán xác suất lượng sai, bản rõ và tất cả khoá con là độc lập và ngẫu nhiên đều. Trong khi chuẩn bị tấn công phân tích lượng sai, người ta dùng xác suất lượng sai tính được để xác định lượng sai sẽ dùng để tấn công, và người ta đã sử dụng giả thiết sau đây.

Giả thiết tương đương ngẫu nhiên

Đối với tất cả các lượng sai (r-1) vòng (α,β)*,công thức*

$$P(\Delta Y(r-1) = \beta / \Delta X = \alpha)$$

$$\approx P(\Delta Y(r-1) = \beta / \Delta X = \alpha, Z^{(1)}=z_1,\ldots,Z^{(r-1)} = z_{r-1}) \qquad (5.7)$$

đúng đối với phần căn bản của các giá trị khoá $(z_1,\ldots,z_{r-1})$.

Một lượng sai (r-1) vòng sẽ được gọi là *có ích trong tấn công phân tích lượng sai (DC-useful)* nếu (5.7) đúng đối với lượng sai này.

Từ mô tả tấn công lượng sai và từ việc có 2^m-1 giá trị có thể của $(\Delta Y(r-1)$ đối với mật mã khối m bít, người ta rút ra kết quả sau.

Định lý 5.15. *Giả sử giả thiết tương đương ngẫu nhiên là đúng, khi đó mật mã lặp r vòng với các khoá con độc lập sẽ bị tổn thương đối với phân tích lượng sai nếu và chỉ nếu hàm vòng là yếu và tồn tai một lượng sai có ích (r-1) vòng* (α,β) *sao cho*

$$P(\Delta Y(r-1) = \beta / \Delta X = \alpha) >> 1/(2^m-1),$$

trong đó m là độ dài khối của mật mã.

Từ những thảo luận trên, ta thấy rằng độ an toàn của mật mã lặp chống lại tấn công phân tích lượng sai phụ thuộc vào các xác suất lượng sai, đại lượng này lại phụ thuộc vào sự lựa chọn phép toán nhóm được dùng để xác định "lượng sai". Để phân tích lượng sai thành công, phép toán nhóm phải được chọn để *cực đại hoá xác suất lượng sai có ích*. Sự lựa chọn phép toán nhóm làm cho mật mã Markov tỏ ra là thích hợp nhất.

II.2.4. Độ phức tạp của tấn công phân tích lượng sai

Từ nghiên cứu về tấn công phân tích lượng sai, ta thấy rằng *độ phức tạp dữ liệu* của tấn công là *hai lần* số cặp bản rõ cần thiết được chọn cho mã kép. *Độ phức tạp dữ liệu* của tấn công phân tích lượng sai chủ yếu là *tổng số tính toán* được dùng để tìm giá trị có thể cho khoá con $Z^{(r)}$ từ các bộ ba ($\Delta Y(r-1)$, Y, Y*), trong thực tế chúng độc lập với r và trong hầu hết trường hợp tổng số này là tương đối nhỏ vì hàm vòng là yếu về mật mã. Độ phức tạp của tấn công phân tích lượng sai đối với mật mã r vòng đã được định nghĩa như *số các phép mã hoá* được sử dụng. Chú ý rằng ở đây *số này là độ đo độ phức tạp dữ liệu*.

Cho $C_d(r)$ là độ phức tạp dữ liệu của tấn công phân tích lượng sai. Bây giờ ta chỉ ra *cận dưới* độ phức tạp dữ liệu của tấn công phân tích lượng sai lên mật mã lặp r vòng (*phép chứng minh cận dưới này đã thực hiện trong Chương 2*).

Định lý 5.16. *Giả sử giả thiết tương đương ngẫu nhiên là đúng, thì trong tấn công lên mật mã lặp r vòng bởi tấn công phân tích lượng sai, có*

$$C_d(r) \geq 2/\left(p_{\max}^{(r-1)} - \frac{1}{2^m - 1} \right), \qquad (5.8)$$

trong đó $p_{\max}^{(r-1)} = \max_{\alpha} \max_{\beta} P\left(\Delta Y(r-1) = \beta \mid \Delta X = \alpha\right)$, *và* m *là độ dài khối của bản rõ. Đặc biệt, nếu* $p_{\max}^{(r-1)} \approx \frac{1}{2^m - 1}$, *thì tấn công phân tích lượng sai không thể thành công.*

Từ bất đẳng thức (5.8) ta thấy rằng khi $p_{\max}^{(r-1)} \leq 3 * 2^{-m}$ thì ít nhất 2^{m-1} cặp phép mã hoá cần cho tấn công. Suy ra rằng số các cặp rõ/mã (lựa chọn hoặc được biết) cần thiết là khoảng 2^m. Nhưng chỉ có 2^m các cặp khác nhau như thế đối với một khoá cố định khiến cho nếu đối với hầu hết các bản mã, các bản rõ tương ứng này được biết đối với người tấn công, thì anh ta không có nhu cầu xác định khoá vì anh ta biết bản rõ đối với mỗi bản mã. Như vậy, ta có thể nói rằng mật mã thực tế là an toàn chống lại tấn công phân tích lượng sai nếu $p_{\max}^{(r-1)} \leq 3 * 2^{-m}$ thậm chí nếu độ phức tạp dữ liệu(2^m) là nhỏ hơn nhiều độ phức tạp (xử lý) của tấn công tìm kiếm vét cạn tất cả khoá con đối với mật mã lặp.

Khi trình bày phương pháp thám lượng sai tấn công DES, Biham và Sharmir đưa ra khái niệm "các đặc trưng i vòng". Với quan niệm trong bài này của chúng ta, đặc trưng i vòng đó là một bộ $(\alpha, \beta_1, \ldots, \beta_i)$ đươc xem như một giá trị có thể của $(\Delta X, \Delta Y(1), \ldots, \Delta Y(i))$. Như vậy, đặc trưng một vòng trùng với lượng sai một vòng và đặc trưng i vòng xác định một dãy i lượng sai, $(\Delta X, \Delta Y(j)) = (\alpha, \beta_j)$. Hai tác giả trên đã định nghĩa xác suất của đặc trưng i vòng là

$$P(\Delta Y(1) = \beta_1, \Delta Y(2) = \beta_2, \ldots, \Delta Y(i) = \beta_i \mid \Delta X = \alpha)$$

trong đó bản rõ X và các khoá con $Z^{(1)}, \ldots, Z^{(i)}$ là độc lập và phân phối đều. Ở đây, ta sử dụng quan niệm lượng sai thay cho đặc trưng là vì trong phân tích lượng sai mật mã r vòng, duy nhất tri thức về $\Delta Y(r-1)$ là cần thiết để xác định khoá con $Z^{(r)}$, còn các lượng sai trung gian $\Delta Y(j)$, $1 \le j < r-1$, có thể không thành vấn đề. Như vậy, bằng cách sử dụng xác suất lượng sai thay cho xác suất đặc trưng, trong thực tế ta xét xác suất thật sự để thám lượng sai thành công mà không phải là cận dưới của xác suất này. Đây là lý do vì sao ta đã có thể rút ra *cận dưới* cho độ phức tạp của tấn công phân tích lượng sai từ xác suất các lượng sai. Mặt khác, xác suất của các đặc trưng cung cấp *cận trên* về độ phức tạp dữ liệu của tấn công. Đặc biệt, đối với mật mã Markov mà xác suất lượng sai của nó có thể được xấp xỉ tốt bởi xác suất đặc trưng (đó là trường hợp DES có số vòng nhỏ), việc sử dụng các đặc trưng là thực tế hơn vì xác suất của đặc trưng i vòng có thể dễ dàng tính được từ xác suất của các đặc trưng 1 vòng.

Đối với mật mã Markov với các khoá con ngẫu nhiên độc lập phân phối đều, xác suất của đặc trưng r vòng được cho bởi phương trình Chapman-Kolmogorov như

$$P\left(\Delta Y(1) = \beta_1, \ldots, \Delta Y(r) = \beta_r \mid \Delta X = \beta_0\right) = \prod_{i=1}^{r} P\left(\Delta Y(i) = \beta_i \mid \Delta X = \beta_{i-1}\right).$$

Từ phương trình trên suy ra rằng xác suất của lượng sai r vòng (β_0, β_r) là

$$P\left(\Delta Y(1) = \beta_1, \ldots, \Delta Y(r) = \beta_r \mid \Delta X = \beta_0\right) = \sum_{\beta_1} \sum_{\beta_2} \cdots \sum_{\beta_{r-1}} \prod_{i=1}^{r} P\left(\Delta Y(1) = \beta_i \mid \Delta X = \beta_{i-1}\right).$$

trong đó các tổng lấy trên tất cả các giá trị có thể của lượng sai giữa các phần tử khác nhau, nghĩa là trên tất cả các phần tử nhóm trừ phần tử trung hoà e.

II.2.5. Độ an toàn của cac mật mã Markov

Đối với phân tích lượng sai cho mật mã lặp, điều quan trọng là xác định xác suất lượng sai. Với mật mã Markov, các xác suất như thế được xác định duy nhất bởi ma trận chuyển của nó. Đoạn sau đây, ta thảo luận độ an toàn của mật mã Markov bằng cách xem xét ma trận chuyển P về tính bật khả quy, giá trị riêng và tính đối xứng của nó. Đôi khi, ta sẽ nói xích P thay cho việc nói xích Markov có ma trận chuyển P.

II.2.5.1. Khi nào thì mật mã Markov là an toàn?

Độ an toàn của mật mã lặp dựa trên niềm tin rằng, có thể nhận được hàm "mạnh" về mật mã bằng cách lặp một hàm "yếu" về mật mã đủ số lần. Các kết quả sau đây chỉ ra rằng đối với các mật mã Markov có các ma trận chuyển P nguyên thuỷ (nghĩa là tồn tại r sao cho ma trận P^r không có phần tử 0), phép lặp sẽ nâng cao độ an toàn chống lại phân tích lượng sai.

Định lý 5.17. *Cho* $F: Z_2^n \to Z_2^n$ *là hàm vòng với ma trận chuyển ergodic P. Khi đó với lượng sai bản rõ* ΔP *khác 0 bất kỳ và lượng sai bản mã* ΔC_r *khác 0 bất kỳ, khi số vòng* $r \to \infty$, *ta có*

$$P_r\{ \Delta C_r = j \mid \Delta P = i \} \to 1 / N. \qquad (5.9)$$

Như vậy, các mật mã có ma trận chuyển ergodic sẽ triệt tiêu tấn công phân tích lượng sai sau số vòng đủ lớn.

Định lý 5.18. *Đối với xích Markov có dạng nêu trong định lý 6.11, xích các lượng sai là bất khả quy nếu và chỉ nếu với mỗi cặp bản rõ (x,x*) và mỗi cặp bản mã (y,y*), tồn tại số nguyên* r_0 *và một sự lựa chọn các khoá con cho* r_0 *vòng đầu tiên sao cho, với* r_0 *vòng đầu tiên này và với các khoá con đã lựa chọn, thì x được mã thành y và x* được mã thành y*.*

Chứng minh. Cho $S = X \otimes Z_A$ và $S^* = X^* \otimes Z_A$, thì $\Delta S = \Delta X$.

Giả sử rằng xích lượng sai là bất khả quy. đối với cặp bản rõ (x,x*) và cặp bản mã (y,y*), cho α là lượng sai của x và x*, β là lượng sai của y và y*. Từ tính bất khả quy suy ra rằng có một r_0 sao cho

$$P(\Delta Y(r_0) = \beta \mid \Delta S = \alpha) = P(\Delta Y(r_0) = \beta \mid \Delta X = \alpha) > 0,$$

nó kéo theo rằng tồn tại các khoá con $z_B^{(1)}, z_A^{(2)}, z_B^{(2)}, ..., z_A^{(r_0)}, z_B^{(r_0)}$, và s, s* với $\Delta S = \alpha$, sao cho dưới các khoá này, s biến thành y, s* biến thành y*. Cho $z_A^{(1)} = x \otimes s^{-1}$, khi đó $s^* = x^* \otimes z_A^{(1)}$ vì $\Delta S = \Delta X$. Như vậy, dưới các khoá lựa

chọn $z_B^{(1)}, z_A^{(2)}, z_B^{(2)}, ..., z_A^{(r_0)}, z_B^{(r_0)}$, cặp bản rõ (x,x*) biến thành cặp bản mã (y,y*).

Điều ngược lại là rõ ràng và đúng đối với tất cả các mật mã Markov.

Từ định lý 5.18 ta thấy rằng đối với xích Markov có hàm vòng dạng này, nếu ma trận chuyển là nguyên thuỷ (*suy ra xích là bất khả quy*), thì tồn tại r_0 sao cho, với mỗi cặp bản rõ (x,x*) và mỗi cặp bản mã (y,y*) và với mỗi $r \geq r_0$, tồn tại sự lựa chọn các khoá con sao cho hàm mật mã r vòng với các khoá con này có thể ánh xạ cặp bản rõ (x,x*) thành cặp (y,y*). Nếu xích Markov bất khả quy nhưng không nguyên thuỷ, nghĩa là nếu xích có chu kỳ, thì với mỗi r, tồn tại cặp bản rõ (x,x*) và cặp bản mã (y,y*) sao cho, với tất cả các khoá con có thể, hàm mã hoá r vòng *không thể* ánh xạ cặp bản rõ (x,x*) thành cặp (y,y*).

Cho $\Lambda(P)$, hoặc đơn giản là Λ, là số phần tử khác 0 trong ma trận P. Nếu F được chọn ngẫu nhiên đều thì Λ là biến ngẫu nhiên nhận giá trị nguyên. Mục tiêu của chúng ta là chỉ ra rằng Λ vượt trội N*logN với xác suất dần đến 1 và như vậy kéo theo rằng P là ergodic với hầu hết F. Ta thừa nhận kết quả sau đây:

Định lý 5.19. *Nếu hàm F được chọn ngẫu nhiên phân phối đều thì*

$$P_r\left\{\Lambda \geq N^2/\left(2\log(N+1)\right)\right\} = 1 - 0\left(N^{-\log\log N + 5.5}\right). \qquad (5.10)$$

Như vậy, với số đỉnh N bất kỳ và số cạnh đã cho Λ bất kỳ,dưới giả thiết hợp lý, dựa trên lý thuyết đồ thị cơ sở ngẫu nhiên, ta có hệ quả:

Hệ quả 5.20. *Nếu hàm F được chọn ngẫu nhiên phân phối đều thì xác suất để "xích Markov các lượng sai là ergodic" sẽ dần đến 1 khi* $N \to \infty$.

II.2.5.2. Sự hội tụ và xáo trộn nhanh

Cho $P=[p_{i\ j}]$ là ma trận ergodic với $\Pi = (\pi_1, \ldots, \pi_N)$ là phân phối giới hạn của nó. Phép toán nhóm ở đây được hiểu là XOR. Cho $p_i^{(r)}$ là phân phối trạng thái sau r bước khi được bắt đầu ở trạng thái i. Độ lệch giữa $p_i^{(r)}$ và Π được xác định là

$$\left\| p_i^{(r)} - \Pi \right\| = \frac{1}{2}\sum_{j=1}^{N} \left| p_{ij}^{(r)} - \pi_j \right| = \frac{1}{2}\sum_{j=1}^{N} e_{ij}^{(r)} \qquad (5.11)$$

Một xích là *xáo trộn nhanh* nếu $e_{ij}^{(r)} \to 0$ như hàm nhanh của r. Ban đầu, các kết quả liên quan với xáo trộn nhanh chỉ được áp dụng vào những xích đặc biệt (như xích khả đảo theo thời gian). Xích P là khả đảo theo

thời gian nếu $\Pi_i p_{ij} = \Pi_j p_{ji}$ với tất cả các trạng thái i,j. Nếu P không thể đảo ngược theo thời gian thì xét $M(P) = P\tilde{P}$, với $\tilde{P} = \left[\tilde{p}_{ij}\right], \tilde{p}_{ij} = \Pi_j p_{ji} / \Pi_i$. Người ta chứng minh được rằng, M(P) khả đảo theo thời gian, là ergodic nếu P là ergodic, và có phân phối giới hạn Π nếu Π là phân phối giới hạn của cả P và $\tilde{P}$. Lợi ích của M(P) là có thể giới hạn $\left\|P_i^{(r)} - \Pi\right\|$ bằng cách sử dụng các giá trị riêng của nó.

Đối với xích ergodic P, định lý Perron-Frobenius phát biểu rằng giá trị riêng lớn nhất của nó là 1, còn tất cả các giá trị riêng khác bé hơn 1 theo môđun. Đặc biệt, N giá trị riêng của M(P) là thực và không âm. Hệ quả là, sự hội tụ của xích được xác định bởi độ lớn của giá trị riêng lớn thứ hai. Cho N giá trị riêng của M(P) là $1=\beta_1>\beta_2 \geq \ldots \geq \beta_N > 0$.

Định lý 5.21 *[Fill]*. Với trạng thái bất kỳ i, có $4\Pi_i \left\|P_i^{(r)} - \Pi\right\|^2 \leq (\beta_2)^r$.

Có vài phương pháp đánh giá β_2 khi M(P) khả đảo theo thời gian và phương pháp ta sử dụng ở đây dựa trên bất đẳng thức Poincaré và các đường dẫn chính tắc. Một kết quả từ nghiên cứu xáo trộn nhanh các xích Markov đã chỉ ra rằng sự hội tụ của xích P phụ thuộc vào các tính chất hình học của G_M (đồ thị cơ sở của M(P)). Cho $M(P) = [q_{ij}]$, từ tính khả đảo theo thời gian ta xác định

$$d_{ij} = \Pi_i q_{ij} = \Pi_j q_{ji} . \qquad (5.12)$$

Với $G_M = (V,E)$, cho $\delta(v_i,v_j)$ là đường dẫn có hướng giữa v_i và v_j với các cạnh không lặp lại. Cho Γ là tập các đường dẫn $\delta(v_i,v_j)$ chỉ chứa một đường dẫn đối với mỗi cặp véctơ (v_i,v_j) trong G_M. Tập Γ tồn tại vì P và M(P) là bất khả quy. Độ dài của đường dẫn $\delta(v_i,v_j)$ được xác định là

$$\left|\delta(v_i, v_j)\right| = \sum_{e \in \delta(v_i, v_j)} d(e)^{-1} \qquad (5.13)$$

với tổng được lấy trên tất cả các cạnh e trong $\delta(v_i,v_j)$ và $d(e) = d_{ij}$. Cuối cùng xác định R

$$R \overset{def}{=} R(\Gamma) = \max_e \sum_{e \in \delta(v_i, v_j) \in \Gamma} \Pi_i \Pi_j \left|\delta(v_i, v_j)\right| \qquad (5.14)$$

trong đó max được lấy trên tất cả các cạnh định hướng trong đồ thị và tổng lấy trên tất cả các đường dẫn chứa cạnh e.

Bất đẳng thức 5.22 *[Poincaré]*. Đối với xích khả đảo thời gian ergodic, có

$$\beta_2 \leq 1-(1/R) \qquad (5.15)$$

Nói chung, vì ma trận chuyển P mô tả phân phối lượng sai không thể đảo thời gian nên mục tiêu của ta là chỉ ra rằng sự hội tụ của P^r có thể được đánh giá nhờ định lý Fill và bất đẳng thức Poincaré.

II.2.5.3. Đánh giá các giá trị riêng

Trong mục này, ta xét ma trận P rut ra từ ánh xạ lượng sai 2-đều.

Bổ đề 5.23. $M(P) = PP^T$.

Dễ dàng chứng minh bổ đề này vì $\Pi_i = \Pi_j$ nên $\tilde{P} = P^T$.

Hệ quả 5.24. M(P) không có phần tử 0, hay đồ thị cơ sở G_M của M(P) là đầy đủ.

Chứng minh. Cho $M(P)=[q_{ij}]$ và chú ý rằng

$$q_{ij} = \sum_{k=1}^{N} P_{ik} \tilde{P}_{kj} = \sum_{k=1}^{N} P_{ik} P_{jk} \qquad (5.16)$$

Nhưng theo cấu trúc, mỗi dòng $P_{i1},P_{i2},\ldots,P_{iN}$ của P có $2^{n-1} = N/2+1/2$ phần tử khác 0, nghĩa là trên 2 dòng bất kỳ số phần tử khác 0 là N+1. Nếu với mọi k=1,…,N mà $P_{ik}*P_{jk} = 0$ (tức là $P_{ik} = 0$ hoặc $P_{jk} = 0$) thì trên 2 dòng i và j số phần tử 0 ít nhất là N, hay số phần tử khác 0 nhiều nhất là N. Ta gặp mâu thuẫn. Vậy, trên 2 dòng i và j, tồn tại ít nhất một k sao cho $P_{ik}*P_{jk} >0$, kéo theo $q_{ij} >0$ với mọi i,j =1,…,N. ĐPCM.

Bây giờ xét việc ứng dụng bất đẳng thức Poincaré để đánh giá độ lệch trong P, ở đó cần đến tập đường dẫn Γ cho G_M. Nhưng vì G_M là đầy đủ nên ta chỉ đơn giản lấy đường dẫn $\delta(v_i,v_j)$ là cạnh định hướng nối các đỉnh này, nghĩa là $\delta(v_i,v_j)=e_{ij}$. Vì mỗi cạnh chỉ được dùng 1 lần trong đường dẫn nên tổng trong (5.14) chỉ có 1 từ. Để xác định R, ta chỉ xác định $|\delta(v_i,v_j)|$ của mỗi đường dẫn, nghĩa là của mỗi cạnh. Độ dài của cạnh $e=e_{ij}$ là nghịch đảo của $d_{ij} = \Pi_i q_{ij}$ trong đó $1/\Pi_i =N$ và q_{ij} được cho trong (5.16). Vì $p_{ij} = 2/2^n$ nếu $p_{ij} >0$, hay $p_{ij} = 2/2^n.[p_{ij} >0]$ nên có thể viết

$$q_{ij} = \frac{4}{2^{2n}} \sum_{k=1}^{N} [p_{ik} > 0][p_{jk} > 0] \qquad (5.17)$$

Khi i=j thì tổng trong (5.17) là $2^{n-1} = N/2+1/2$. Tuy nhiên, khi $i \neq j$, ta sẽ mô hình hoá việc lấy tổng như biến ngẫu nhiên α_e có phân phối nhị thức

b(p,N) với $p = \left(\frac{1}{2} + \frac{1}{2n}\right)^2$ là xác suất để 2 dòng khác nhau của P là khác 0 trên cùng một vị trí. Có N^2-N biến ngẫu nhiên α_e để xem xét tương ứng với $q_{i\,j}$, $i \neq j$.

Khi cho những hạn chế này trên M(P), công thức (5.14) đối với R là

$$R = R(\Gamma) = \max_e \frac{1}{N^2} \frac{N 2^{2n}}{4\alpha_e} = \max_e \frac{(N+1)^2}{4N\alpha_e} \qquad (5.18)$$

Do đó R đạt cực đại khi α_e đạt cực tiểu. Để ý rằng $\beta_2 \leq 2^{-t}$ khi $R = \frac{2^t}{2^t - 1}$, kéo theo việc có α_e nào đó sao cho (e đạt max)

$$\alpha_e = \frac{2^t - 1}{2^t} \cdot \frac{(N+1)^2}{4N} = \left(1 - 2^{-t}\right) pN \qquad (5.19)$$

Thế thì α_e - pN = -2^{-t}pN. Từ đây suy ra rằng một cận tốt cho β_2 là có được nếu giá trị α_e nhỏ nhất nhỏ hơn chút ít giá trị trung bình pN của phân phối nhị thức b(p,N) của nó.

Ví dụ. Đối với ma trận P của ánh xạ lượng sai 2-đều ρ ở phần trên, có thể kiểm tra rằng M(P) là ma trận 7×7 với 1 trên đường chéo chính và 1/8 ở những vị trí khác. Khi đó pN=16/49 và $\alpha_e = (1-2^{-3})$pN với tất cả $q_{i\,j}$, $i \neq j$, kéo theo $R = \frac{2^3}{2^3 - 1}$, $\left\| P_i^{(r)} - \Pi \right\|$ có giá trị như nhau với tất cả i. Bất đẳng thức Poincaré phát biểu rằng $\beta_2 \leq 2^{-3}$, và các giá trị tính toán cụ thể ở bảng dưới đây chỉ ra rằng $(\beta_2)^r = 2^{-3r}$ là cận trên của (4/7). $\left\| P_i^{(r)} - \Pi \right\|$ như định lý Fill đã dự đoán.

r	$\left\| P_i^{(r)} - \Pi \right\|$	(4/7). $\left\| P_i^{(r)} - \Pi \right\|^2$	2^{-3r}
1	0.10714	0.45918	0.875
2	0.53571	0.11479×10^{-1}	0.10937
3	0.13392×10^{-1}	0.71747×10^{-3}	0.13671×10^{-1}
4	0.66964×10^{-2}	0.17936×10^{-3}	0.17089×10^{-2}
5	0.16741×10^{-2}	0.11210×10^{-4}	0.21362×10^{-3}

Đối với n lớn ta không thể tính M(P) chính xác như trong ví dụ trên và cần một vài phát biểu xác suất. Bổ đề sau cho một cận về xác suất để $|\alpha_e - pN| \geq 2^{-t}pN$.

Bổ đề 5.25. Nếu $0<p<1/2$ và $(pN)^{-1/2}<\varepsilon<1/6$ thì

$$P_r(|\alpha_e - pN| \geq \varepsilon pN) < \exp\{-\varepsilon^2 pN/(3(1-p))+\varepsilon/(1-p)\} \qquad (5.20)$$

trong đó e là cơ số của logarit tự nhiên.

Nếu $2^{2t} = 0(N)$ và cho p=1/4 thì xác suất để tất cả N^2-N đại lượng α_e lệch với pN một lượng bé hơn $2^{-t}pN$ là $\left(1-\exp\left\{\frac{N}{9.2^{2t}}+\frac{4}{3.2^t}\right\}\right)^{N^2-N}$ hay áp dụng công thức $(1+\gamma)^k \approx 1+k\gamma$, ta có xác suất

$$1-e^{\ln N+\ln(N-1)-\frac{N}{9.2^{2t}}+\frac{4}{3.2^t}}+O\left(e^{2\ln N-\frac{2N}{2^{2t}}}\right) \qquad (5.21)$$

Sử dụng (5.21) ta có thể lập luận các kết quả hội tụ cho những mật mã có tham số lớn như kích cỡ khối và số các vòng.

Ví dụ. Xét mật mã 16 vòng có cỡ khối n = 64 hoặc $N=2^{64}-1$ mà ta muốn chỉ ra rằng $\beta_2 \leq 1/32$. Trong trường hợp này t=5 và (5.21) giới hạn xác suất lệch lớn hơn $1-e^{128-2^{50}}$, đây là đại lượng mà trong thực hành được coi bằng 1. Khi đó nếu ít nhất một α_e bé hơn giá trị trung bình, thì định lý Fill tuyên bố rằng

$$4\Pi_i \left\| P_i^{(16)} - \Pi \right\|^2 \leq (\beta_2)^{16} < 2^{-5.16} = 2^{-80}$$

Vì độ lệch có N từ nên độ lệch trung bình phương xấp xỉ $\sqrt{\frac{N.2^{-80}}{N}} = 2^{-40}$.

II.2.5.4. Sự không đối xứng của ma trận chuyển

Mặc dù tính bất khả quy và các giá trị riêng của ma trận chuyển có tầm quan trọng đặc biệt đối vơi sự an toàn của mật mã Markov , nhưng khó xác định các đại lượng này đối với các mật mã có cỡ lớn (chẳng hạn m=64, ma trận chuyển có cỡ $(2^{64}-1)\times(2^{64}-1)$). Một yêu cầu thực tế quan trọng hơn là như sau: *Ma trận xác suất chuyển của mật mã Markov phải không đối xứng.*

Giả sử ma trận chuyển P là đối xứng. Khi đó, mỗi cặp các lượng sai 1-vòng dạng (a,b) và (b,a) sẽ có cùng xác suất $p_{ab}=p_{ba}$. Giả sử rằng (a,b) là công thức lượng sai 1-vòng có khả năng nhất. Khi đó (a,b,a,b,…,a,b,a) sẽ

là đặc trưng 2i-vòng có khả năng nhất. Đối với i nhỏ và đối với trường hợp p_{ba} lớn đáng kể, (a,a) sẽ là lượng sai 2i-vòng với xác suất cao. Xác suất cao này có thể sẽ được xấp xỉ bởi xác suất của đặc trưng 2i-vòng tương ứng, nghĩa là $p_{aa}^{(2i)} \approx p_{ba}^{2i}$. Tương tự, (a,b) sẽ là lượng sai (2i+1)-vòng với xác suất cao. Xác suất cao này có thể sẽ được xấp xỉ bởi xác suất của đặc trưng (2i+1)-vòng có khả năng nhất (a,b,a,b,…,a,b), nghĩa là $p_{ba}^{(2i+1)} \approx p_{ba}^{2i+1}$. Như vậy, việc nối công thức lượng sai 1-vòng có khả năng nhất với chính nó r-1 lần sẽ cho đặc trưng 2i-vòng có khả năng nhất, nó sẽ dẫn đến cung cấp lượng sai r-vòng với xác suất cao. Tính không đối xứng của ma trận chuyển ngăn chặn việc nối công thức lượng sai 1-vòng có khả năng cao.

III. THÁM TUYẾN TÍNH

Đây là tấn công không vét cạn vào DES nổi tiếng vào bậc nhất. Cơ sở của tấn công này là tìm quan hệ tuyến tính giữa những bít nhất định của bản rõ P, bản mã C và khoá K, được biểu diễn như $\sum P+\sum C=\sum K$. Hình thức hơn, tấn công này rút ra xấp xỉ dạng:

$$\sum_{i=1}^{n} a_{1i} p_i + \sum_{i=1}^{n} a_{2i} c_i = \sum_{i=1}^{m} a_{3i} k_i \,(\mathrm{mod}\, 2)\,. \qquad (5.22)$$

với $a_{1i} \in \{0,1\}$ và $P=p_1,\ldots,p_n$; $C=c_1,\ldots,c_n$; $K=k_1,\ldots,k_m$. Matsui đã chỉ ra rằng $\sum K$ có thể được xác định một cách chính xác nhờ sử dụng phương pháp hợp lý cực đại nếu có đủ tổng số N_L các bản mã đã biết. Đặc biệt, nếu xấp xỉ trong (5.22) đúng với xác suất q* thì tấn công này được hy vọng thành công 98% khi $N_L \approx |\, q^*-1/2\,|$. Kết quả của tấn công là tri thức về 1 bít thông tin liên quan với khoá, cụ thể là giá trị của $\sum K$.

Ta sẽ giả thiết rằng ở mỗi vòng, khoá con được XOR với bản mã hiện thời. Cho $Y_i = y_{i,1}y_{i,2}\ldots y_{i,n}$ là véc tơ nhị phân n-chiều với i=0,1,…,n. Bản mã trung gian ở vòng i được ký hiệu là $C_i = c_{i,1}c_{i,2}\ldots c_{i,n}$; khoá con $Ki = k_{i,1}k_{i,2}\ldots k_{i,n}$. *Xấp xỉ được sử dụng ở vòng i* sẽ là

$$\sum_{j=1}^{n} y_{ij}\left(c_{ij} + k_{ij}\right) = \sum_{j=1}^{n} y_{i+1,j} c_{i+1,j}\,, \qquad (5.23)$$

và nó *đúng với xác suất* q_i nào đó. Như vậy Y_i ký hiệu tổng của các biến vào được dùng trong xấp xỉ ở vòng i và cũng là tổng của các biến ra được dùng trong xấp xỉ ở vòng (i-1), với i>1. Khi r cái xấp xỉ này được cộng modulo 2, hai từ có Y_i sẽ được bỏ qua, để lại một xấp xỉ với xác suất q* theo kiểu một tổng các bít rõ $Y_0 = \sum P$, một tổng các bít mã $Y_r = \sum C$ và

một tổng các bít khoá ∑K, có dạng như (5.22). Tất cả các tuyến tính hoá của mật mã lặp đều có thể được rút ra theo kiểu này.

Khi các bít khoá được giả thiết là ngẫu nhiên và độc lập, những xấp xỉ trong (5.23) với các i khác nhau là độc lập. Khi đó, Gallager đã chứng tỏ rằng

$$q^* = \frac{1}{2} + 2^{r-1}\prod_{i=1}^{r}\left(q_i - \frac{1}{2}\right) \qquad (5.24)$$

nó vốn được đề cập đến như bổ đề Piling-Up của Matsui. Sẽ thuận lợi hơn nếu viết lại công thức q^* dạng các hệ số tương quan. Nếu f và g là hai hàm boolean n-bít thì hệ số tương quan, hoặc đơn giản hơn là tương quan của f và g, được ký hiệu c(f,g) với định nghĩa

$$c(f,g) = 2^{-n}\sum_{X}(-1)^{f(X)}(-1)^{g(X)} = \Pr(f = g) - \Pr(f \neq g).$$

Nếu với mỗi i ta xem Y_i như hàm boolean $\sum_{j=1}^{n} y_{ij}c_{ij}$, thì $q_i - \frac{1}{2} = \frac{c(Y_{i-1}, Y_i)}{2}$.

Bổ đề 5.26. Cho $Y_0 = \sum P$ và $Y_r = \sum C$ biểu diễn tổ hợp tuyến tính của bản rõ và bản mã tương ứng trong mật mã lặp r vòng dùng các khoá con độc lập $K_1,K_2,\ldots,K_r$. Cho q^* là xác suất của xấp xỉ $Y_0 = Y_r + \sum K$. Nếu ta xem Y_r như hàm của bản rõ và tất cả các khoá con thì

$$c\left(Y_0, Y_r + \sum K\right) = \prod_{i=1}^{r} c(Y_{i-1}, Y_i)$$

và vì vậy

$$q^* = \frac{1}{2} + \frac{1}{2}\prod_{i=1}^{r} c(Y_{i-1}, Y_i). \qquad (5.25)$$

III.1. XÍCH ĐỂ THÁM TUYẾN TÍNH

Cho hàm vòng $F : Z_2^n \to Z_2^n$ là song ánh và $f_i : Z_2^n \to Z_2$ là hàm boolean mô tả bít thứ i của F, nghĩa là $F=(f_1,\ldots,f_n)$. Ký hiệu $X=x_1x_2\ldots x_n$ là véc tơ biến vào.. Cho $N=2^n-1$, ta xác định ma trận N×N là $P=[p_{i\,j}]$, $1 \le i,j \le N$, trong đó $i=(i_1\ldots i_n)_2$, $j=(j_1\ldots j_n)_2$ là biểu diễn cơ số 2 và

$$p_{ij} = c^2\left(\sum_k i_k x_k, \sum_k j_k f_k\right). \qquad (5.26)$$

Nghĩa là, $p_{i\,j}$ là bình phương của hệ số tương quan giữa tổ hợp tuyến tính của x_i và một tổ hợp tuyến tính của f_i. Chú ý rằng xích này bao gồm các

giá trị tầm thường i=j=0. Người ta gọi P là ma trận tương quan. Ta sẽ chỉ ra rằng, P có tổng mỗi dòng và tổng mỗi cột bằng 1 (nghĩa là ngẫu nhiên kép). Điều này có ứng dụng mạnh mẽ đối với phân phối giới hạn của mũ của P (nếu nó tồn tại). Trước hết ta xét các cột của P. Tổng một cột cho trước tương ứng là tương quan tổng cộng giữa $f=\sum j_k f_k$ và tập tất cả các hàm tuyến tính không tầm thường. Vì f là cân bằng nên tương quan tổng cộng với tất cả các hàm tuyến tính là 1 đối với hàm boolean bất kỳ. Mặt khác, vì F là song ánh nên biến đầu vàocó thể được biểu diễn như hàm song ánh F^{-1} của các biến đầu ra. Bằng lập luận tương tự, người ta thấy rằng mỗi dòng của P cũng có tổng là 1. Vì vậy, ta có

Bổ đề 5.27. Đối với hàm F, ma trận tương quan của nó là ngẫu nhiên kép.

Cho $Y=(Y_0,Y_1,\ldots,Y_r)$ là ký hiệu xấp xỉ r-vòng của mật mã khối. Cho $P^r=\left[p_{ij}^{(r)}\right]$ là luỹ thừa bậc r của P. Theo định nghĩa của xích ta có

$$p_{ij}^{(r)} = \sum_{\substack{\sum i_k p_k \\ \sum j_k c_k}} \prod_{t=1}^{r} c^2\left(Y_{t-1}, Y_t\right). \qquad (5.27)$$

Như thế, bình phương của tương quan toàn thể bị chặn trên bởi $p_{ij}^{(r)}$ (Nghĩa là,

$p_{ij}^{(r)} \geq \prod_{t=1}^{r} c^2\left(Y_{t-1}, Y_t\right)$) và từ bổ đề…,định lý ….($\lim_{r\to\infty} p_{ij}^{(r)} = \frac{1}{N}$) ta suy ra

Định lý 5.28. *Đối với hàm vòng* $F: Z_2^n \to Z_2^n$ *có ma trận tương quan ergodic P, tương quan giữa hàm tuyến tính* Y_0 *bất kỳ của bản rõ P và hàm tuyến tính bất kỳ* Y_r *của bản mã C đối với số lớn r vòng sẽ thoả mãn tiệm cận*

$$\left|2q^* - 1\right| = \left|\prod_{i=1}^{r} c\left(Y_{i-1}, Y_i\right)\right| \leq \frac{1}{\sqrt{N}}. \qquad (5.28)$$

Như vậy, với giả thiết tương tự với giả thiết về sự tương đương ngẫu nhiên, định lý này đã chỉ ra rằng những mật mã có ma trận tương quan ergodic là triệt tiêu thám tuyến tính sau số vòng đủ lớn.

III.2. TÍNH ERGODIC ĐỐI VỚI CÁC HÀM VÒNG NGẪU NHIÊN

Cho Λ(P) (hoặc đơn giản là Λ) ký hiệu số các phần tử khác 0 trong ma trận tương quan P. Giả sử F được chọn ngẫu nhiên đều, khi đó Λ trở nên một biến ngẫu nhiên nguyên có thể được biểu diễn dạng tổng nguyên $\Lambda=\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_{ij}$ của các biến ngẫu nhiên nhị phân, trong đó $\lambda_{ij}=1$ nếu $p_{ij}>0$ và $\lambda_{ij}=0$ nếu $p_{ij}=0$. Mục tiêu của ta là tìm kỳ vọng và phương sai của Λ. Người ta có thể chứng minh được rằng

$$E[\Lambda] = \sum_{1\le i,j\le N}\left(1 - p_{ij}\right) \tag{5.29}$$

$$Var[\Lambda] = \sum_{1\le i,j<N}\left(p_{ij} - p_{ij}^2\right) + \sum_{1\le i,j<N}\sum_{1\le i',j'<N}\left(p_{ij,i'j'} - p_{ij}p_{i'j'}\right), \tag{5.30}$$

trong đó $p_{ij}=P_r(\lambda_{ij}=0)$, $p_{ij,i'j'}=P_r(\lambda_{ij}=0, \lambda'_{i'j'}=0)$.
Bây giờ ta xác định những xác suất này và hành vi tiệm cận của chúng khi 2^n lớn như trong thực hành.

Bổ đề 5.29. Đối với bất kỳ i,j

$$p_{ij} = P_1 \overset{def}{=} \frac{\binom{2^{n-1}}{2^{n-2}}^2}{\binom{2^n}{2^{n-1}}} \sim \sqrt{\frac{8}{\pi 2^n}}. \tag{5.31}$$

Việc rút ra các xác suất cặp $p_{ij,i'j'}$ còn khó khăn hơn nhiều. Có 3 trường hợp: (1) i'=i và j'≠j, (2) i'≠i và j'=j, (3) i'≠i và j'≠j. Chúng được hỗ trợ bởi 2 bổ đề sau:

Bổ đề 5.30. Đối với bất kỳ i'=i, j'≠j hoặc i'≠i và j'=j

$$p_{ij,i'j'} = P_2 \overset{def}{=} \frac{\sum_{k=0}^{2^{n-2}}\binom{2^{n-2}}{k}}{\binom{2^n}{2^{n-1}}} \sim \frac{8}{\pi 2^n}. \tag{5.32}$$

Bổ đề 5.31. Đối với bất kỳ i'≠i và j'=j

$$p_{ij,i'j'} = P_3 \overset{def}{=} \frac{1}{\binom{2^n}{2^{n-1}}^2\binom{2^{n-1}}{2^{n-2}}^4}\cdot\sum_{M}\frac{2^n!}{\prod_{l=0}^{15}m(l)} \sim \frac{8}{\pi 2^n} \tag{5.33}$$

trong đó M là tập tất cả các nghiệm nguyên không âm (m(0),…,m(15)) của hệ 16 phương trình tuyến tính dạng (với ký hiệu $l=(l_1,l_2,l_3,l_4)_2$)

$$\sum_{l_{s_3},l_{s_4}} m(l) = 2^{n-2}, \forall l_{s_1}, l_{s_2} \tag{5.34}$$

đối với $(s_1,s_2)=(1,2),(3,4),(1,3)$ và $(2,4)$ tương ứng, trong đó (s_1,s_2,s_3,s_4) là hoán vị của (1,2,3,4).
Từ các bổ đề vừa nêu, theo (5.29), (5.30) và bất đẳng thức Chêbưsép ta có

Định lý 5.32. Đối với hàm F được lấy phân phối đều, số Λ các phần tử khác 0 trong ma trận tương quan thoả mãn

$$P_r\{\Lambda \ge N(\log N + \gamma_N)\} = 1\text{-}0(1/N). \tag{5.35}$$

Từ định lý này và lý thuyết đồ thị ngẫu nhiên, ta suy ra

Hệ quả 5.33. Đối với hàm F được lấy phân phối đều, xác suất để xích Markov đối với thám tuyến tính là ergodic sẽ dần đến 1 khi $N \to \infty$.

IV. MẬT MÃ MARKOV VÀ CÁC NHÓM LUÂN PHIÊN

IV.1. CÁC ĐIỀU KIỆN LÝ THUYẾT NHÓM CHO HÀM MỘT VÒNG

Ta xét mật mã lặp r vòng trên tập hữu hạn χ và nó là mật mã Markov trong liên hệ với lượng sai đã cho. Ta tìm điều kiện cho xích Markov tương ứng là bất khả quy và không có chu kỳ. Những điều kiện này độc lập với lượng sai đã cho.

Ký hiệu

$G = < \{ f_z : z \in Z\} >$ (Z là tập các khoá vòng) là nhóm hoán vị trên χ được sinh bởi các hàm 1 vòng f_z và

$$\forall t \in N, H_t = \left\langle \left\{ f_{z_1} \circ f_{z_2} \circ ... \circ f_{z_t} \middle| (z_1, z_2, ..., z_t) \in Z^t \right\} \right\rangle$$

là nhóm hoán vị được sinh bởi các hàm t vòng.

Bổ đề 5.34. $\forall t \in N$, hoặc $G = H_t$ hoặc H_t là nhóm con chuẩn thật sự của G.

Kết quả sau đây được ứng dụng phân tích DES và IDEA(32):

Định lý 5.35. Nếu G là nhóm đối xứng hoặc nhóm luân phiên trên χ và $|\chi| \geq 5$ thì đối với tất cả các mật mã Markov tương ứng, các xích Markov của lượng sai là bất khả quy và không có chu kỳ.

IV.2. ỨNG DỤNG CHO DES

Ký hiệu

$<\Pi>$ là nhóm hoán vị được sinh bởi tập các hoán vị Π; $V_m = \{0,1\}^m$;

$A_{2^{64}}$ là nhóm luân phiên trên V_{64} (nhóm các hoán vị chẵn);

$S_{2^{64}}$ là nhóm đối xứng trên V_{64} (nhóm tất cả các hoán vị).

Trong [38] *Ralph Wernsdorf đã chứng minh các kết quả sau:*

Định lý 5.36. $G = A_{2^{64}}$

Hệ quả. $G(n) = A_{2^{64}}$, trong đó G(n) là nhóm sinh bởi các hàm n vòng của DES với các khoá con độc lập, n=2,3,…

Từ định lý 5.34 và các kết quả này, suy ra

Định lý 5.37. Cho f_z là hàm một vòng của DES thì đối với tất cả các mật mã Markov tương ứng, xích các lượng sai là bất khả quy và không có chu kỳ, nghĩa là sau đủ nhiều vòng các lượng sai sẽ hầu như có xác suất gần bằng nhau.

Như vậy, nếu giả thiết tương đương ngẫu nhiên đúng cho phần mật mã tương ứng, thì DES là an toàn chống lại thám lượng sai sau đủ nhiều vòng đối với tất cả các mật mã Markov này. Những kết quả này còn đúng cho tất cả các mật mã lặp r vòng, nếu các hàm một vòng là tương tự DES sinh ra nhóm luân phiên.

IV.3. ỨNG DỤNG CHO IDEA

Đối với các mật mã IDEA(8) và IDEA(16) người ta đã chứng minh rằng các xích Markov dựa trên các lượng sai đã cho là bất khả quy và không có chu kỳ [22]. *Điều này cho bằng chứng để phỏng đoán rằng IDEA(32) và IDEA(64) có cùng tính chất. Trong* [22] *đây còn là một vấn đề mở, nhưng* [13] *đã đưa ra câu trả lời cho IDEA(32).*

Định lý 5.38. Đối với IDEA(32), ta có $G = A_{2^{32}}$

Từ đây suy ra

Định lý 5.39. Cho f_z là hàm một vòng của IDEA(32) thì đối với tất cả các mật mã Markov tương ứng, xích các lượng sai là bất khả quy và không có chu kỳ, nghĩa là sau đủ nhiều vòng các lượng sai sẽ hầu như có xác suất gần bằng nhau.

Như vậy, nếu giả thiết tương đương ngẫu nhiên đúng cho phần mật mã tương ứng, thì IDEA(32) là an toàn chống lại thám lượng sai sau đủ nhiều vòng đối với tất cả các mật mã Markov này.

Chú ý. • *Định lý 5.38 kéo theo rằng một vài "việc làm ngắn" IDEA(32) để thám có thể bị loại trừ hoặc bị hạn chế. Mặc dù các kết quả nêu trên không đề cập đến IDEA(64) đầy đủ, nhưng chúng đóng vai trò những lý lẽ cho sức mạnh mật mã cho thuật toán này.*

• *Đối với mật mã lặp r vòng và đối với những khoá đặc biệt nào đó, các hàm một vòng ($f_{z1},...f_{zr}$) có thể chỉ sinh nhóm con $G' \subset G$ nghĩa là không bắc cầu đôi (một nhóm hoán vị trên χ được gọi là có tính bắc cầu nếu với mọi x,y trên χ luôn tồn tại một hoán vị chuyển x thành y. Nhóm được gọi là bắc cầu đôi nếu với 2 cặp có thứ tự bất kỳ trên χ, tồn tại một chuyển vị biến cặp này thành cặp kia...). Điều này cho thấy giả thiết tương đương ngẫu nhiên không luôn luôn đúng ([1]). Vì vậy, DES và IDEA(32) có thể không an toàn đối với tất cả các lượng sai thoả mãn định lý 5.38 và 5.39 tương ứng.*

• *Hành vi của các lượng sai không sinh ra các xích Markov vẫn chưa được biết. Sự phân tích nó vẫn còn là vấn đề mở, thú vị.*

V. KẾT LUẬN

• Khi nghiên cứu mã khối dưới góc độ mật mã Markov, người ta đã tìm cách chứng minh mật mã này có xích Markov tương ứng là bất khả quy và không có chu kỳ. Nếu làm được điều này thì có thể khẳng định mật mã là an toàn trước tấn công lượng sai và tấn công tuyến tính khi số vòng lặp đủ lớn.

• Đã có hai cách để chứng minh xích Markov là bất khả quy và không có chu kỳ. Một là dùng lý thuyết đồ thị ngẫu nhiên, và phương pháp thứ hai là sử dụng tính chất của nhóm luân phiên. Phương pháp thứ hai là khó song kết quả của nó là tất định.

• Nhìn chung ta vẫn chưa đưa ra được "số vòng đủ lớn" là bao nhiêu?

• Giả thiết tương đương ngẫu nhiên không phải luôn luôn đúng vì vậy để chứng minh một mã khối là an toàn trên quan điểm xích Markov cũng còn rất nhiều việc phải làm.

CHƯƠNG 6
XÂY DỰNG THUẬT TOÁN MÃ KHỐI MK_KC-01-01

Trong chương này vận dụng các cơ sở lý thuyết đã nghiên cứu khảo sát trong các chương trước, chúng tôi thiết kế một thuật toán mã khối cụ thể đảm bảo các thông số an toàn, hiệu quả phục vụ cho đề tài. Thuật toán lấy tên là MK_KC-01-01.

Trước hết, phần ngẫu nhiên hoá dữ liệu được xây dựng theo cấu trúc 3 lớp: trong, giữa và ngoài cùng. Lớp ngoài cùng chúng tôi chọn cấu trúc Feistel có thể đánh giá được các độ đo an toàn trước các tấn công mạnh nhất hiện nay. Lớp giữa là có cấu trúc kiểu mạng thay thế hoán vị 2-SPN (có 2 tầng phi tuyến được xen giữa bởi 1 tầng tuyến tính) như đã nêu trong chương 3. Lớp trong cùng là các hộp thế phi tuyến. Các hộp thế này được lựa chọn từ 2 hộp thế S1 và S2 đã được khảo sát trong chương 3 có các độ đo an toàn tốt tránh các kiểu tấn công đã khảo sát. Ngoài ra các phép hoán vị, phép dịch vòng được lựa chọn cẩn thận sao cho hệ mã có tính khuyếch tán ngẫu nhiên đều. Các phép biến đổi đầu vào và đầu ra đều lấy là phép XOR với khoá tương ứng.

Phần lược đồ khoá, dùng để ngẫu nhiên một mầm khoá có độ dài 128-bit thành các khoá con đủ cho các vòng lặp và các phép biến đổi đầu vào và đầu ra. Phần lược đồ khoá cũng đã chú ý để tránh tấn công kiểu trượt khối, đồng thời sử dụng tối đa các hộp thế phi tuyến của phần ngẫu nhiên hoá dữ liệu.

Dưới đây là mô tả cụ thể thuật toán MK_KC-01-01.

I. PHẦN NGẪU NHIÊN HOÁ DỮ LIỆU

I.1. Mô hình mã, giải mã (dựa trên cấu trúc Feistel)

a/ Các thành phần, phép toán cơ bản tham gia trong mô hình.

-Phép cộng (môđulô 2) giữa các dữ liệu 64-bit và 128-bit: ký hiệu là XOR

-Phép dịch vòng sang trái 11-vị trí: ký hiệu là LRO-11.

-Hoán vị giả ngẫu nhiên trên tập hợp 32 phần tử: ký hiệu là RP-32.

-Các hộp thế dựa trên hàm lũy thừa trên các trường GF(2^8): ký hiệu là S_1, S_2.

-Hàm vòng: cấu trúc 2-SPN (hai tầng hộp thế)

-Mô hình mã-dịch: (kiểu cấu trúc Feistel)

-Lược đồ tạo khoá: (tựa ngẫu nhiên)

b/ Mô tả cụ thể

MK-KC 01 là mô hình mã khối với độ dài khối rõ-mã là 128-bit. Khối rõ 128-bit được tác động bởi khóa có độ dài 128-bít theo 16-vòng lặp như sau.

+Trước hết khối rõ X được cộng XOR với khoá K_0-128 bit được X_0-128 bít.

+Sau đó ta viết khối $X_0 = L_0R_0$, trong đó L_0 là 64 bít bên trái, R_0 là 64-bít bên phải của X_0.

+Tiếp theo L_0R_0 biến đổi qua 16 vòng lặp theo một hàm xác định để được các xâu L_iR_i, $1 \le i \le 16$, theo qui tắc:

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

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i).$$

trong đó F là hàm sẽ được mô tả ngay dưới đây.

Sau 16 vòng lặp ta lấy ra khối $Y_0 = R_{16}L_{16}$.

+Thực hiện cộng XOR khối Y_0 với khoá K_{17}-128 bit ta sẽ nhận được bản mã:

$$Y = Y_0 \text{ XOR } K_{17}.$$

Khi dịch mã chỉ cần áp lược đồ mã hóa trên đây vào bản mã Y với khóa con có thứ tự ngược lại: $K_{17}, K_{16}, ..., K_1, K_0$.

*Hàm F(x, K) trong các vòng lặp được thực hiện như sau:

-Trước hết xâu x được cộng XOR với khóa K (64-bit), được xâu (64-bit)

x' = x XOR K;

-Tiếp theo x' được chia liên tiếp thành 8 xâu con độ dài 8-bit, và được đưa vào 8 hộp thế $S_1, S_2, S_1, S_2, S_1, S_2, S_1, S_2$;

-Qua 8 hộp thế ghép nối tiếp 8 xâu con độ dài 8-bit thành xâu x'' (64-bit);

-Sau đó xâu x" được dịch vòng sang trái 11-vị trí được xâu x"':

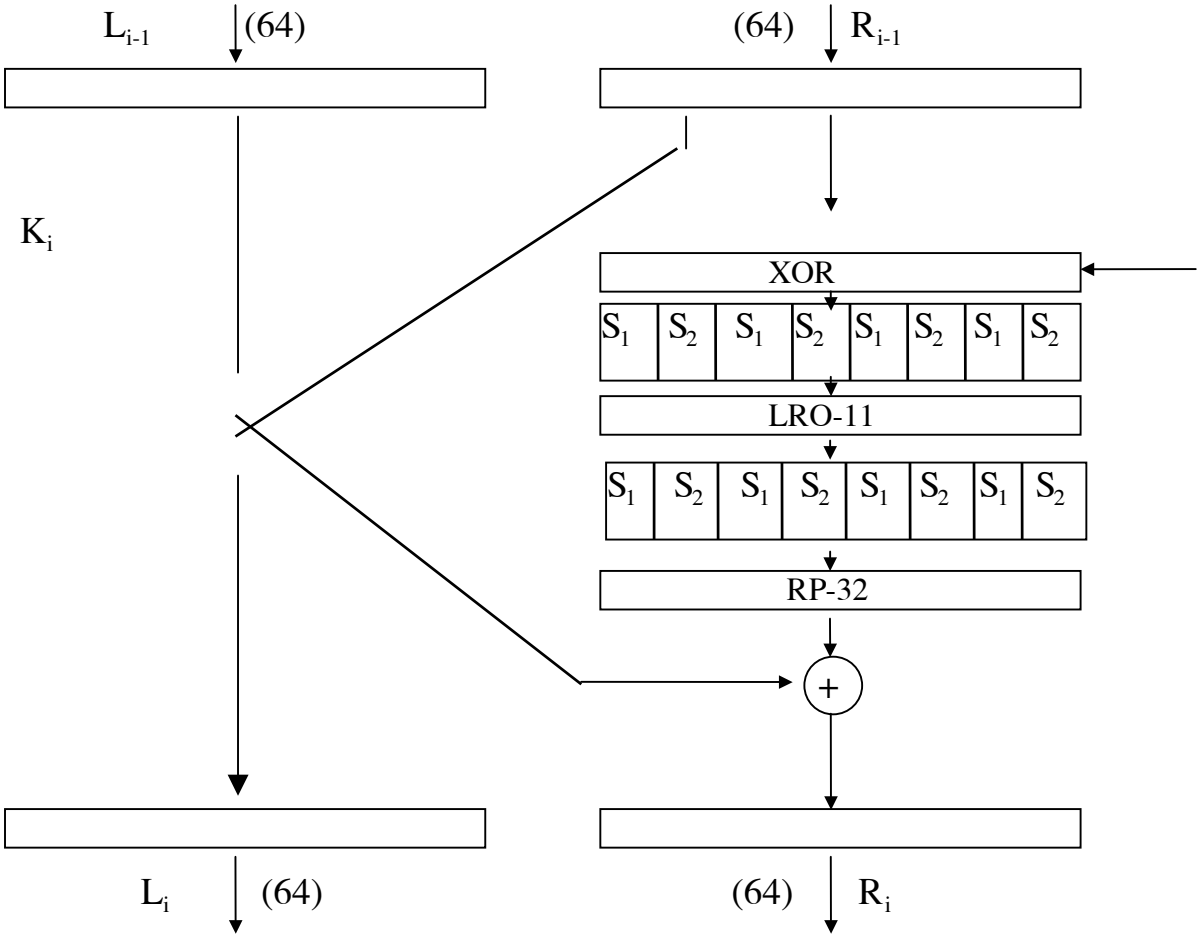
x"' = LRO-11 (x");

-Tiếp theo xâu x"' lại được chia liên tiếp thành 8 xâu con độ dài 8-bit, và được đưa vào 8 hộp thế $S_1, S_2, S_1, S_2, S_1, S_2, S_1, S_2$;

-Qua 8 hộp thế ghép nối tiếp 8 xâu con độ dài 8-bit thành xâu x"" (64-bit);

-Xâu x"" được chia liên tiếp thành 32 nhóm độ dài 2-bit, và được đưa vào hoán vị RP-32.;

-Xâu kết quả (64 -bit) sau khi ghép liên tiếp 32 nhóm 2-bit đã qua hoán vị chính là F(x, K).



Hình 5.1: Sơ đồ một vòng lặp của thuật toán.

I.2. Các tham số cụ thể

Các đa thức để tạo các hộp thế S_1, S_2, và hoán vị giả ngẫu nhiên RP-32 được lựa chọn là:

$f(x) = 1 + x + x^3 + x^4 + x^8$ (trên trường $GF(2^8)$; bất khả qui

$g(x) = 1 + x + x^5 + x^6 + x^8$ (trên trường $GF(2^8)$; nguyên thuỷ

$h(x) = 1 + x^2 + x^5$.

Trong đó, S_1 là ánh xạ affine của ánh xạ nghịch đảo α^{-1} mod f(x), với $\alpha \in GF(2^8)$ trong trường $GF(2^8)$, và S_2 là ánh xạ affine của ánh xạ nghịch đảo β^{-1} mod g(x), với $\beta \in GF(2^8)$ trong trường $GF(2^8)$, cụ thể:

$$S_1 = A(\alpha^{-1} \bmod f(x)) + B ,$$

$$S_2 = C(\beta^{-1} \bmod g(x)) + D.$$

Trong đó A và C là các ma trận có ngược 8x8, và B và D là các véc tơ nhị phân 8-bit sau đây:

A =[F8, 7C, 3E, 1F, 8F, C7, E3, F1];

$B = C6^T = (1\ 1\ 0\ 0\ 0\ 1\ 1\ 0)^T$.

C=[8F, E3, C7, F1, 1F, 3E, 7C, F8];

$D = DA^T = (1\ 1\ 0\ 1\ 1\ 0\ 1\ 0)^T$.

**Bảng tra hai S-hộp cụ thể là:*

<u>Hộp thế S_1</u>

```
198, 62,238,222, 79,214,246,163, 12,128,230,212,127,235,213,110,
 83, 65,147,190, 95,154,226, 15,181, 43, 69,245, 57, 37, 78,  3,
237,191,201,100,108,252,239, 51, 44,165,167,143,142, 27,140,168,
 32,227,196,195, 24,105,160, 89,224, 72,  1, 71,215,228, 77,174,
144,193, 52, 88,216,118, 90,  5, 74,220,107,205,148,199,244, 33,
202,139,  0,183,  4, 63,141,218, 86,211,125,156, 82, 50, 26,243,
 11,247, 85,223,194,178,204,161,162,159, 64,254, 10, 60,249, 21,
138,197,  2,241, 73,185, 28,175, 61,109, 91,132,  8,255,207, 75,
179, 48,200, 55,250,233, 34,232, 35,229,126,188, 38,186,152,206,
  6,129,242, 59, 68, 84,  9, 17, 98,119, 29, 40,123,122,208,219,
  7, 76, 92, 80,146, 96, 36, 58, 67,203, 53, 70,137,169, 39,158,
231, 19,236,182,177,171,114,149, 54,106, 47, 87,166, 94,117, 16,
 93, 30,164,116, 56,101, 45, 99, 23,187, 46,248,210,189,209, 81,
 14,124,173,102, 18,192,111,112,134,172,234,157, 97,131,184,121,
135, 31, 25,136,150,155,113, 41,217,120,225,151,115,170, 20,251,
```

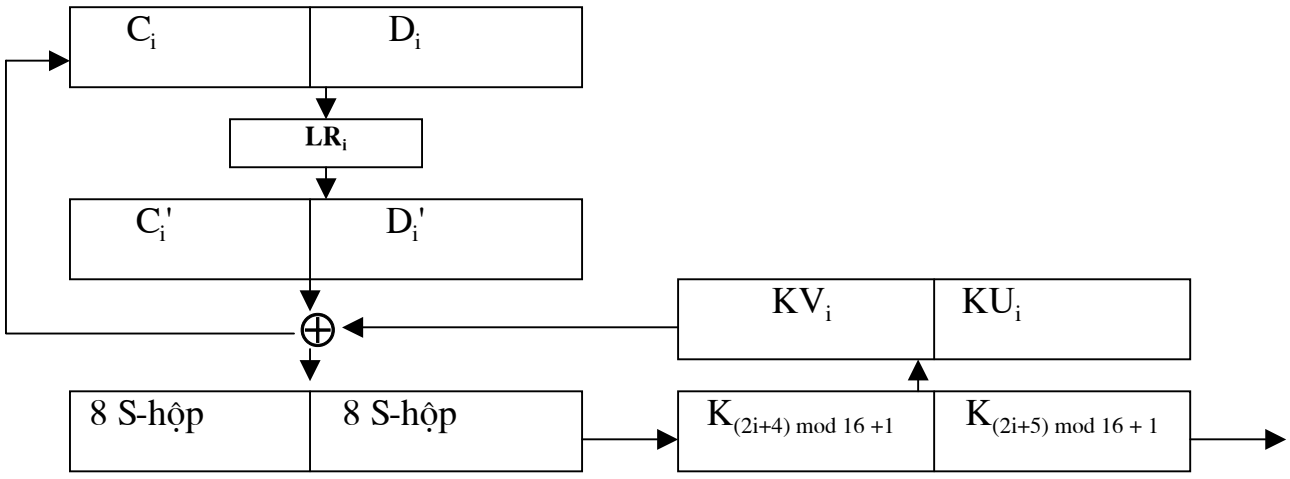
```
 49,133,145,176,253,103, 66, 22,130,153,180,240, 13, 42,221,104,
```

Hộp thế S_2

```
218, 85,155, 91,195, 90, 26,206,145,252,154, 94,  3,239, 40,245,
 64,196, 14, 59, 67,121,152,227, 72,161,198,104, 37,166,181,224,
233, 77,171,240,241, 31,237, 32,209,200,204,141,251, 97,255,189,
210,238,222,174, 19,133,188, 69,156,179,228,194, 44,158,199,187,
124,170, 80, 51, 93,116, 55,230,183, 95,129,169,126,172, 30,164,
217,143,146, 76,144, 35,137, 89, 52, 46,  7,127, 54, 86, 23,110,
225,108, 70, 45,216, 56,223,202,184, 13,178, 27,151, 81,107,102,
249,168,150,106,197, 75, 17,205, 39,135, 65,254,147,109,173,193,
 78, 82,221, 34, 25,235,214,157,160,232,117, 63,212, 57,125,219,
148,138, 24, 33,176, 48,229,100,246,  2,103,211,  1,119, 92, 47,
226,177, 49, 50,120,242,208, 87,192,175, 38,180,139,203,162,123,
236, 96,159, 58, 74,207,118,  8, 84,247,163, 66,186, 53,  6, 18,
 71, 21,190,112, 83,134,167,128, 98, 79,213, 29, 88, 73, 42, 68,
149,113,201,244, 22,220,131,114,250,191,153,  9,132,142, 61,  5,
140, 99,101,253,122, 15,  4,165, 43,115,234, 12,  0,185, 16,111,
 36,136, 10, 60,105,130,182, 20,248, 11, 62, 28,231,215, 41,243,
```

*Hoán vị giả ngẫu nhiên RP-32, được lấy từ kết quả đã nghiên cứu, chúng có khoảng cách phân bố đều góp phần tạo ra độ khuyếch tán tốt cho dữ liệu.

II. PHẦN LƯỢC ĐỒ KHOÁ



Hình 5.2: Sơ đồ một vòng lặp tạo khoá

MÔ TẢ

Cho khoá K độ dài 128-bit.

Trước hết chia K thành 2 xâu liên tiếp 64-bít, ký hiệu là $C_0 \| D_0$.

- $C_0 \| D_0$ được dịch vòng sang trái LR_0-vị trí được xâu $C_0' \| D_0'$.
- $C_0' \| D_0'$ được cộng XOR với $KV_0 \| KU_0$ và đưa vào hai dãy 8 S-hộp.
- Qua hai bộ 8 S-hộp ta thu được khoá $K_5 \| K_6$.
- Đặt $C_1 \| D_1 := C_0' \| D_0'$ (XOR) $KV_0 \| KU_0$.
- Đặt $KV_1 \| KU_1 := K_5 \| K_6$ và lặp lại quá trình trên.

Sau 8 vòng lặp trên đây ta được 16 khoá con K_i (i =1..16).

Hai khoá K_0 và K_{17} là kết quả của vòng lặp thứ 9 và thứ 10.

Véc tơ ban đầu $KV_0 \| KU_0 := 0123456789abcdef_{ASCII}$.

Các bước dịch vòng trái qui định như sau:

$LR_i = 23$ với i =1, 2, 4, 8.

$LR_i = 5$, với i =0, 3, 5, 6, 7, 9.

III. CÁC THÔNG SỐ AN TOÀN LÝ THUYẾT VÀ THỰC NGHIỆM

1. Trước hết, các hộp thế S_1, S_2 được xây dựng như trên sẽ có các độ đo an toàn là: $DP(S_i)_{max} = LP(S_i)_{max} = 2^{-6}$. Mỗi vòng có ít nhất 2 S-hộp hoạt động. Sau 16 vòng số các S-hộp hoạt động ít nhất là 2 x 10 = 20. Do đó theo các chú ý và kết quả đã nêu, với số vòng lặp đã chọn của mô hình MK-KC 01, thì các độ đo an toàn thực tế của mô hình sẽ là:

$$DP(\text{MK-KC 01})_{max} \leq 2^{-6 \text{ x2x10}} = 2^{-120}$$

$$LP(\text{MK-KC 01})_{max} \leq 2^{-6 \text{ x2x10}} = 2^{-120}.$$

Từ các thông số này, độ phức tạp của các tấn công vi sai hay tấn công tuyến tính tương ứng cũng sẽ vào cỡ 2^{120}. Đây là con số an toàn chấp nhận được với năng lực tính toán mạnh nhất hiện nay.

2. Để khảo sát độ an toàn thực hành, nhóm nghiên cứu đã tiến hành hai công việc:

- Thứ nhất, nhóm đã thực hiện sinh các mẫu khóa giả ngẫu nhiên từ mô hình theo kiểu tạo chu trình khóa dòng. Sau đó đã tiến hành kiểm tra các

mẫu khóa đó theo hệ tiêu chuẩn kiểm tra khóa ngẫu nhiên. Kết quả là nguồn khóa do sơ đồ MK-KC 01 sinh ra hoàn toàn đảm bảo được chất lượng về tiêu chuẩn khóa ngẫu nhiên như đã nói trên.

- Thứ hai, nhóm nghiên cứu đã thử chạy thống kê phân bố vi sai bằng cách lấy ΔX ngẫu nhiên, sau đó cố định lại. Cho bản rõ X chạy ngẫu nhiên, đặt $X' = X \oplus \Delta X$, sau đó chương trình sẽ tính phân bố số lượng các vi sai đầu ra ΔY tương ứng. Kết quả là với 10^6 cặp bản rõ ngẫu nhiên, ta thấy không có ΔY nào xuất hiện quá một lần. Điều này chứng tỏ độ đo vi sai thực tế của MK-KC 01 là rất nhỏ.

- Ngoài ra tốc độ mã hóa khối của thuật toán đạt cỡ 20 Mb/s trên máy tính 933 MHz.

Với các thông số an toàn lý thuyết và thực nghiệm trên đây, chúng tôi thấy mô hình thuật toán MK-KC- 01-01, hoàn toàn đáp ứng được các yêu cầu thực tế đặt ra.

TÀI LIỆU THAM KHẢO

STT	Tác giả	Tên bài	Tạp chí
1	AES (nhiều tác giả)	Tuyển tập 15 hệ mã khối dự tuyển chuẩn mã tiên tiến (AES)	Tài liệu từ Internet
2	E. Biham	New types of cryptanalytic attacks using related keys	EUROCRYPT' 93, pp 398-409
3	A. Biryukov, D. Wagner	Slide Attacks	Fast Software Encryption, 1999, pp 245-259
4	A. Biryukov, D. Wagner	Advanced Slide Attacks	EUROCRYPT' 2000, pp 589-606
5	S. Burton, Jr. Kaliski, M.J.B. Robshaw	Linear Cryptanalysis using Multiple Approximations	CRYPTO'94, pp 26-39
6	G.Carter, E. Dawson, and L. Nielsen	Key Schedules of Iterative Block Ciphers	Tài liệu từ Internet, (10 trang)
7	F. Chabaud and S. Vaudenay	Links between differential and linear cryptanalysis.	Eurocrypt' 94, pp 256-365.
8	C. Charnes, L. O'Connor, J. Pieprzyk, R. Safavi-Naimi, Y. Zeng	Comments on Soviet Encryption Algorithm GOST	EUROCRYPT'94, pp 433-438
9	L.J. O'Conner and J. Dj Golic'	A unified markov approach to differential and linear cryptanalysis.	Asiacrypt, November 1994.
10	L.J. O'Conner.	Design Product Ciphers Using Markov Chain	Selected Area in Cryptography 1994.
11	L.J. O'Conner.	Convergence in Differential Distributions,	Crypto'95, p.13-23.
12	I.I. Ghicman, A.V. Skorokhod.	Nhập môn về lý thuyết các quá trình ngẫu nhiên	NXB "HAYKA", Maxcova 1977.
13	G. Hornauer, W. Stephan, R.Wernsdorf.	Markov Ciphers and Alternating Groups,	Eurocrypt'93, p.453-460.
14	T. Jacobsen, L.R. Knudsen	Interpolation Attacks on the Block Cipher.	Fast Software Encryption, 1997, pp 28-40
15	Y. Kaneko, F. Sano, K. Sakurai	On Provable Security against Differential and Linear Cryptanalysis in Generalized Feistel Ciphers with Mutiple Random Functions	Tài liệu từ Internet, 15 trang
16	J. Kelsy, B. Schneier, and D. Wagner	Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SEFER,and Triple-DES	CRYPTO'96, pp 237-251
17	L.R. Knudsen	Block Ciphers-Analysis, Design and Applications	July, 1, 1994 (Ph. D Thesis)
18	L. Knudsen	Practically secure Feistel ciphers	Fast Software

			Encryption, 1993, pp 211-221.
19	L.R. Knudsen	New potentially "weak' keys for DES and LOKI	EUROCRYPT' 94, pp 419-424
20	L. Knudsen, M.J.B. Robshaw	Non-linear Approximations in Linear Cryptanalysis	EUROCRYPT' 96, pp 224-236
21	M. Kwan, J. Pieprzyk	A General purpose Technique for Locating Key Scheduling Weaknesses in DES-like Cryptosystems	ASIACRYPT'91, pp 237-246
22	X. Lai	On the Design and Security of Block Ciphers	Hartung-Gorre Verlag Konstanz, 1995
23	Lai, X, J.L. Massey and S. Murphy	Markov Ciphers and Differential cryptanalysis	Eurocrypt' 91, p.17-38.
24	M. Matsui	New Block Encryption Algorithm MISTY	Fast Software Encryption, 1997, FSE'97, pp 54-68
25	M. Matsui	New structure of block ciphers with provable security against differential and linear cryptanalysis	Fast software Encryption, 1996, pp 21-23.
26	M. Matsui	Linear Cryptanalysic Method for DES Cipher	EUROCRYPT' 93, pp 386-397
27	M. Matsui	The First Experimental Cryptanalysic of the Data Encryption Standard	CRYTO' 94, pp 1-11
28	S. Moriai, T. Shimoyama, T. Kaneko	Interpolation Attacks of the Block Cipher: SNACK.	Fast Software Encryption, 1999, pp 275-289
29	K. Nyberg	Differentially uniform mappings for cryptography.	EUROCRYPT'93, pp. 55-64, 1994.
30	K. Nyberg	Linear Approximation of Block Ciphers	Eurocrypt'94, pp 439-444.
31	K. Nyberg, L. R. Knudsen	Provable security against a differetial cryptanalysis	Journal of Cryptology, Vol. 8, pp. 27-37, 1995.
32	Savan Patel, Zulfikar Ramzan, and Ganapathy S. Sundaram,	Towards Making Luby-Rackoff Ciphers Optimal and Practical	Fast Software Encryption, 1999, pp 171-185
33	Kenneth G. Paterson	Imprimitive Permutation Groups and Trapdoor in Iterated Block Ciphers,	Fast Software Encryption, 1999, pp 201-214
34	T. Shimoyama, T. Kaneko	Quadratic Relation of S-box and Its Application to the Limear Attack of Full Round DES	CRYPTO'98, pp 200-211
35	J. Seberry, X. M. Zhang and Y. Zheng	Relationships Among Nonlinearity Criteria	EUROCRYPT'94, pp. 76-388, 1995.
36	D. R. Stinson	Cryptography: Theory and Practice	1995 by CRC Press,

			Inc.
37	Nguyễn Duy Tiến.	Các mô hình xác suất và ứng dụng, Phần I- Xích Markov và ứng dụng	NXB Đại học Quốc gia Hà Nội,2000.
38	R.Wernsdorf.	The One-Round Functions of the DES Generate the Alternating Group,	Proc. Eurocrypt' 92, LNCS 658, 1993, p. 99-112.

Phụ lục A:
Mã nguồn chương trình thám mã DES-8 vòng

Thư viện hàm mã/dịch DES, các khai báo chung được khai báo trong file glodes.h. Chương trình thể hiện thám mã DES 8 vòng cụ thể như sau:

```
#include <stdio.h>
#include <math.h>
#include "glodes.h"

/* Compute n random plaintext-ciphertext corresponding under the same
key */
void create_randomPC(void)
{
      long i;
      struct timeval timeval;
        struct timezone timezone;

      timezone.tz_dsttime=0;

      /* Tao khoa con */
      khoades8rt(kst,khoatham,1);

        if (gettimeofday(&timeval, &timezone)!=0) {
          printf("\n Can't get system time in sub function
part1_step2");
        }

      srand((unsigned long )((timeval.tv_usec <<10) ^
timeval.tv_usec));
      for (i=0;i<N;i++) {
            bro[i][0]=((unsigned long) rand()<<16)^rand();
            bro[i][1]=((unsigned long) rand()<<16)^rand();
            bma[i][0]=bro[i][0];
            bma[i][1]=bro[i][1];
            des8r(bma[i],kst);
            }
}


void create_randomKey(void)
{
      long i;
      struct timeval timeval;
      struct timezone timezone;
      timezone.tz_dsttime=0;

      if (gettimeofday(&timeval, &timezone)!=0) {
          printf("\n Can't get system time in sub function
part1_step2");
      }
```

```
      srand((unsigned long )((timeval.tv_usec <<10) ^
timeval.tv_usec));

      for (i=0;i<8;i++) {
            khoatham[i] = (unsigned char) (rand()^rand());
            }
}


/*          LINEAR ATTACK FUNCTIONS       */

/* 2 phuong trinh tuyen tinh: (theo quy uoc trai->phai la STT tu thap -
> cao)
   o
Pr[3,8,14,25]+Pl[17]+Cl[8,14,25]+F1(Pr,K1)[17]+F8(Cr,K8)[8,14,25] =
       K2[26]+K4[26]+K5[4]+K6[26]                    (1)

   o
Pl[8,14,25]+Cr[3,8,14,25]+Cl[17]+F1(Pr,K1)[8,14,25]+F8(Cr,K8)[17] =
       K3[26]+K4[4]+K5[26]+K7[26]                    (2)
 */

void effective_text()
{
    unsigned long eff_text1, eff_text2;
    unsigned long i;

    for (i=0; i<N; i++) {
      eff_text1 = eff_text2 = 0;

      /* Deal with Equation (1) */

      /* Pr[3,8,14,25]+Pl[17]+Cl[8,14,25] */
      eff_text1 |= (((bro[i][1]&0x20000000)
>>29)^((bro[i][1]&0x01000000)>>24)
                  ^((bro[i][1]&0x00040000)
>>18)^((bro[i][1]&0x00000080)>>7)
                  ^((bro[i][0]&0x00008000)
>>15)^((bma[i][0]&0x00000080)>>7)
                  ^((bma[i][0]&0x01000000)
>>24)^((bma[i][0]&0x00040000)>>18));

      /* Cr[16-Cr[21] a/h toi F8(Cr,K8)[8,14,25] - S5 */
      eff_text1 |= (bma[i][1]&0x0001f800)>>10;  /* eff_text =
0x0000007f*/

      /* Pr[32], Pr[1]-Pr[5] a/h toi F1(Pr,K1)[17] - S1 */
      eff_text1 |= (bro[i][1]&0xf8000000)>>20;  /* eff_text =
0x00000fff*/
      eff_text1 |= (bro[i][1]&0x00000001)<<12;   /* eff_text =
0x00001fff */

      TA[eff_text1] += 1;
```

```
      /* Deal with Equation (2) */

      /* Pl[8,14,25]+Cr[3,8,14,25]+Cl[17]*/
      eff_text2 |=
(((bro[i][0]&0x01000000)>>24)^((bro[i][0]&0x00040000)>>18)
                  ^((bro[i][0]&0x00000080)>>7)
^((bma[i][1]&0x20000000)>>29)

^((bma[i][1]&0x01000000)>>24)^((bma[i][1]&0x00040000)>>18)
                  ^((bma[i][1]&0x00000080)>>7)
^((bma[i][0]&0x00008000)>>15));

      /* Pr[16]-Pr[21] a/h toi F1(Pr,K1)[8,14,25] - S5 */
      eff_text2 |= (bro[i][1]&0x0001f800)>>10;  /* eff_text =
0x0000007f*/

      /* Cr[32], Cr[1]-Cr[5] a/h toi F8(Cr,K8)[17] - S1 */
      eff_text2 |= (bma[i][1]&0xf8000000)>>20;  /* eff_text =
0x00000fff*/
      eff_text2 |= (bma[i][1]&0x00000001)<<12;   /* eff_text =
0x00001fff */
      TB[eff_text2] += 1;
    }
}

/*
   Consider left side of equation (1)
*/
int left_side_1(UINT4 k, UINT4 t)
{
    int tmp = 0;
    UINT4 l=0x00000000L,r;

    UINT4 candidate[2], temp; /* = */

    tmp ^= (t&1); /* Pr[3,8,14,25]+Pl[17]+Cl[8,14,25] */

    /* Chuyen 6 bit khoa a/h toi F1(Pr,K1)[17] - S1 */
    candidate[0]=candidate[1]=0;
    candidate[0] |= (k&0x0000003f)<<24; /* 6 bit dau vao S1 -
0x3f000000 */

    r  = 0;
    r |= (t&0x00000f80)<<20;
    r |= (t&0x00001000)>>12; /* r = 0xf8000001 */

    /* F1(Pr,K1)[17] */
    r      = (r<<1) | (r>>31); /* r = 0xf0000003 */
    Ft(l,r,&candidate[0]);
    l      = (l<<31) | (l>>1);

    tmp ^= ((l>>15)&1);

    r  = 0;
```

```
    r |= (t&0x0000007e)<<10;
    l  = 0x00000000L;

    /* Chuyen 6 bit khoa a/h toi F1(Pr,K1)[17] - S1 */
    candidate[0]=candidate[1]=0;
    candidate[0] |= (k&0x00000fc0)<<2;  /* 6 bit tiep vao S5 -
0x3f003f00*/

    /* F8(Pr,K8)[8,14,25] */
    r      = (r<<1) | (r>>31);
    Ft(l,r,&candidate[0]);

    l      = (l<<31) | (l>>1);

    tmp ^= (((l>>7)^(l>>18)^(l>>24))&1);
    return tmp;
}

/*
   Consider left side of equation (2)
 */
int left_side_2(UINT4 k, UINT4 t)
{
    int tmp = 0;
    UINT4 l=0x00000000L,r;

    UINT4 candidate[2], temp; /* = */

    tmp ^= (t&1); /*  Pl[8,14,25]+Cr[3,8,14,25]+Cl[17] */

    /* Chuyen 6 bit khoa a/h toi F8(Cr,K8)[17] - S1 */
    candidate[0]=candidate[1]=0;
    candidate[0] |= (k&0x0000003f)<<24; /* 6 bit dau vao S1 -
0x3f000000 */

    /* Chuyen 6 bit text a/h toi F8(Cr,K8) - Cr[32], Cr[1]-Pr[5] */
    r  = 0;
    r |= (t&0x00000f80)<<20;
    r |= (t&0x00001000)>>12; /* r = 0xf8000001 */

    r      = (r<<1) | (r>>31); /* r = 0xf0000003 */
    Ft(l,r,&candidate[0]);
    l      = (l<<31) | (l>>1);

    /* F8(Cr,K1)[17] */
    tmp ^= ((l>>15)&1);

    /* Chuyen 6 bit text a/h toi F1(Pr,K1) - Pr[32], Pr[16]-Pr[21]
     *  trong t = 0x0000007e - S5 */
    r  = 0;
    r |= (t&0x0000007e)<<10;
    l  = 0x00000000L;

    /* Chuyen 6 bit khoa a/h toi F1(Pr,K1)[17] - S5 */
```

```
    candidate[0]=candidate[1]=0;
    candidate[0] |= (k&0x00000fc0)<<2;  /* 6 bit tiep vao S5 -
0x00003f00*/

    r      = (r<<1) | (r>>31);
    Ft(l,r,&candidate[0]);

    l      = (l<<31) | (l>>1);


    /* F1(Cr,K1)[8,14,25] */
    tmp ^= (((l>>7)^(l>>18)^(l>>24))&1);

    return tmp;
}

void effective_key()
{
    UINT4 t, k;

    for (k=0; k<4096; k++) {  /* 2^12 effective key bit */
    for (t=0; t<8192; t++)  { /* 2^13 effective text bit */

      /* Kiem tra ve trai cua phuong trinh 1 */
      if (left_side_1(k, t)==0) KA[k] += TA[t];

      /* Kiem tra ve trai cua phuong trinh 2 */
      if (left_side_2(k, t)==0) KB[k] += TB[t];
    }
    }
}

/* Rearrange KA in order of magnitude of |KA - N/2|*/
void rearrangeKA()
{
    long i;
    unsigned long maxKA, minKA;
    unsigned long kmax, kmin, fkey;

    maxKA=minKA=KA[0];
    kmax=kmin=0;
    for (i=1; i<4096; i++)
    {
      if (KA[i]>maxKA) {
          maxKA=KA[i];
          kmax=i;
      }

      if (KA[i]<minKA) {
          minKA=KA[i];
          kmin=i;
      }
    }
```

```
    if (labs(maxKA-(N/2))>labs(minKA-(N/2))){
      /* Chap nhan khoa ung cu vien tuong ung Kmax */
      fkey = kmax; sum1 = 1;fixed_1=kmax;
    }
    else {
      /* Chap nhan khoa ung cu vien tuong ung Kmin */
      fkey = kmin; sum1 = 0;fixed_1 = kmin;     /* [6bit K8] [6 bit K1]
*/
    }

    /*   fkey = [6 bit K8][6bit K1] */

    /* Chuyen khoa tim duoc - fkey vao foundkey */
    foundkey[2] |= (unsigned char)((fkey&1)<<7); /* K1[6] <-> K[17] */
    foundkey[6] |= (unsigned char)((fkey&2)<<6); /* K1[5] <-> K[49] */
    foundkey[7] |= (unsigned char)((fkey&4)<<2); /* K1[4] <-> K[60] */
    foundkey[4] |= (unsigned char)((fkey&8)<<3); /* K1[3] <-> K[34] */
    foundkey[6] |= (unsigned char)((fkey&0x00000010)<<1); /* K1[2] <->
K[51] */
    foundkey[1] |= (unsigned char)((fkey&0x00000020)<<1); /* K1[1] <->
K[10] */

    foundkey[3] |= (unsigned char)((fkey&0x00000040)>>4); /* K8[30] <->
K[30] */
    foundkey[3] |= (unsigned char)((fkey&0x00000080)>>6); /* K8[29] <->
K[31] */
    foundkey[1] |= (unsigned char)((fkey&0x00000100)>>5); /* K8[28] <->
K[13] */
    foundkey[7] |= (unsigned char)((fkey&0x00000200)>>6); /* K8[27] <->
K[61] */
    foundkey[6] |= (unsigned char)((fkey&0x00000400)>>8); /* K8[26] <->
K[54] */
    foundkey[1] |= (unsigned char)((fkey&0x00000800)>>7); /* K8[25] <->
K[12] */
}

/* Rearrange KB in order of magnitude of |KB - N/2|*/
void rearrangeKB()
{
    long i;
    unsigned long maxKB, minKB;
    unsigned long kmax, kmin, fkey;

    maxKB=minKB=KB[0];
    kmax=kmin=0;
    for (i=1; i<4096; i++)
    {
      if (KB[i]>maxKB) {
          maxKB=KB[i];
          kmax=i;
      }

      if (KB[i]<minKB) {
```

```
          minKB=KB[i];
          kmin=i;
      }
    }

    if (labs(maxKB-(N/2))>labs(minKB-(N/2))){
      /* Chap nhan khoa ung cu vien tuong ung Kmax */
      fkey = kmax; sum2=1;fixed_2 = kmax;
    }
    else {
      /* Chap nhan khoa ung cu vien tuong ung Kmin */
      fkey = kmin; sum2=0;fixed_2 = kmin; /* [6bit K1] [6 bit K8] */
    }

    /*   fkey = [6 bit K1][6bit K8] */

    /* Chuyen khoa tim duoc - fkey vao foundkey */
    foundkey[5] |= (unsigned char)((fkey&1)<<5); /* K8[6] <-> K[43] */
                                        /* K8[5] <-> K[10] - dup */
    foundkey[6] |= (unsigned char)((fkey&4)<<4); /* K8[4] <-> K[50] */
                                      /* K8[3] <-> K[60] - dup */
    foundkey[5] |= (unsigned char)((fkey&0x00000010)<<3); /* K8[2] <->
K[41] */
    foundkey[4] |= (unsigned char)((fkey&0x00000020)>>1); /* K8[1] <->
K[36] */

    foundkey[0] |= (unsigned char)((fkey&0x00000040)>>2); /* K1[30] <->
K[4] */
    foundkey[4] |= (unsigned char)((fkey&0x00000080)>>4); /* K1[29] <->
K[37] */
                                         /* K1[28] <-> K[54] - dup */
    foundkey[4] |= (unsigned char)((fkey&0x00000200)>>8); /* K1[27] <->
K[39] */
    foundkey[3] |= (unsigned char)((fkey&0x00000400)>>6); /* K1[26] <->
K[28] */
    foundkey[2] |= (unsigned char)((fkey&0x00000800)>>9); /* K1[25] <->
K[22] */

    /* Tu cac bit tim duoc o phuong trinh (1) + (2) suy them bit so 7
*/
    foundkey[0] |= (unsigned char)

      (((sum2^(foundkey[0]>>4)^(foundkey[6]>>7)^(foundkey[4]>>1))&1)<<1
);
}

/* In ket qua tim duoc ra file hoac man hinh */
void printResults(int a)
{
      long i,j;
      FILE *f;

        f=fopen("Ketqua.txt","a");
```

```
      /* In ket qua tim kiem de so sanh */
      fprintf(f,"\n\n                KET QUA THAM MA DES 8 VONG - Lan
%d", a);
      fprintf(f,"\n\n  Khoa can tham - Khoa tim duoc (DANG NHI
PHAN)\n\n   ");
      for (i=0;i<8;i++)
         for (j=7;j>=0;j--)
            fprintf(f,"%d",(khoatham[i]>>j)&1);
            fprintf(f,"\n   ");
      /* in -> kiem tra lai */
      for (i=0;i<8;i++)
         for (j=7;j>=0;j--)
            fprintf(f,"%d",(foundkey[i]>>j)&1);
      fclose(f);
}


int matsui_linear()
{
    /* Step1: Initial */

    /* Step 2: dem so 13 bit text a/h toi 1 - TA */
    effective_text();

    /* Step 3 + 4: KA - key bit a/h toi 1 */
    effective_key();

    /*  Step 5: Sap xep lai va quyet dinh khoa dung */
    rearrangeKA();
    rearrangeKB();
}


/* DEAL WITH EQUATION 11 */
void effective_text11(void)
{
    unsigned long eff_text, b;
    unsigned long i, sum=0;

    for (i=0; i<N; i++) {
        eff_text  = 0;

      /* Pl[16,17,20] */
      eff_text |=
(((bro[i][0]>>12)^(bro[i][0]>>15)^(bro[i][0]>>16))&1);

      /* Pr[8]...Pr[17] */
      eff_text |= ((bro[i][1]&0x01ff8000)>>14); /* eff_text =
0x000007ff */

      /* Pr[32], Pr[1]-Pr[5] a/h toi F1(Pr,K1)[17] - S1 */
      eff_text |= (bro[i][1]&0xf8000000)>>16;  /* eff_text =
0x0000ffff*/
```

```
      eff_text |= (bro[i][1]&0x00000001)<<16;    /* eff_text =
0x0001ffff */

      b = 0;

      /* Deal with Equation (1) - 9 */

      /* Pr[3,8,14,25]+Pl[17]+Cl[8,14,25] */
      b |= (((bro[i][1]&0x20000000) >>29)^((bro[i][1]&0x01000000)>>24)
           ^((bro[i][1]&0x00040000) >>18)^((bro[i][1]&0x00000080)>>7)
           ^((bro[i][0]&0x00008000) >>15)^((bma[i][0]&0x00000080)>>7)
           ^((bma[i][0]&0x01000000)
>>24)^((bma[i][0]&0x00040000)>>18));

      /* Cr[16-Cr[21] a/h toi F8(Cr,K8)[8,14,25] - S5 */
      b |= (bma[i][1]&0x0001f800)>>10;  /* eff_text = 0x0000007f*/

      /* Pr[32], Pr[1]-Pr[5] a/h toi F1(Pr,K1)[17] - S1 */
      b |= (bro[i][1]&0xf8000000)>>20;  /* eff_text = 0x00000fff*/
      b |= (bro[i][1]&0x00000001)<<12;  /* eff_text = 0x00001fff */

      if (left_side_1(fixed_1, b)==sum1) {
          TC[eff_text] += 1;
      }
    }
}

/* Left Side of : Pl[16,17,20] + F1(Pr,K1)[16,17,20] = K2[25,26,29]
(11)  */
/*
  Description:
  k = 12 bit = [6 bit S3][6 bit S4]
  t = 11 bit = [10 bit S3,S4][1 bit tong]
 */
int leftside_11(unsigned long k, unsigned long t)
{
    int tmp=0;
    UINT4 l=0x00000000L,r=0;

    UINT4 candidate[2], temp; /* = */

    tmp ^= (t&1); /* Pl[16,17,20]  */

    /* Chuyen 6 bit khoa a/h toi F1(Pr,K1)[17] - S1
       tim duoc tu phuong phap cua Matsui*/
    /* fixed_1 = [6bit K8] [6 bit K1] */
    candidate[0]  = candidate[1] = 0;
    candidate[0] |= (fixed_1&0x0000003f)<<24; /* 6 bit dau vao S1 */

    /* Chuyen 6 bit khoa a/h toi F1(Pr,K1)[16] - S3 */
    candidate[0] |= (k&0x00000fc0)<<10; /* 6 bit dau vao S1 */

    /* Chuyen 6 bit khoa a/h toi F1(Pr,K1)[20] - S4 */
    candidate[1] |= (k&0x0000003f)<<16; /* 6 bit dau vao S4 */
```

```
    /* Chuyen 10 bit text a/h toi F1(Pr,K1)[16,20] vao r -
Pr[8]..Pr[17] */
    r |= ((t&0x000007fe)<<14);

    /* 6 bit text a/h toi F1(Pr,K1)[17] */
    r |= ((t&0x0000f800)<<16);
    r |= ((t&0x00010000)>>16);

    r      = (r<<1) | (r>>31);
    Ft(l,r,&candidate[0]);

    l      = (l<<31) | (l>>1);


    /* F1(Pr,K1)[16,17,20] */
    tmp ^= (((l>>12)^(l>>15)^(l>>16))&1);

    return tmp;
}

/* Counting up effective key bits (12 bits) */
void effective_key11(void)
{
    unsigned long k /* 13 key bit */, t, m;
    int right;

    for (k=0; k<4096; k++) { /* 12 key bit */
    for (t=0; t<131072; t++) {
    for (right=0; right<2; right++) {
      m = k;
      m |= ((right&1)<<12);
      if (leftside_11(k, t)==right) {
          U0[m] += TC[t];
      }
      else {
          U0[m] += 4*TC[t];
      }
    }
    }
    }

}


/* DEAL WITH EQUATION 11' (phay) - PHUONG TRINH THU HAI */
void effective_text11p(void)
{
    unsigned long eff_text, b;
    unsigned long i;

    for (i=0; i<N; i++) {
        eff_text  = 0;
      /* NOTE:
```

```
          Pr = bro[i][1]      Pl = bro[i][0]
          Cr = bma[i][1]      Cl = bma[i][0]
         */

        /* Cl[16,17,20] */
      eff_text |=
(((bma[i][0]>>12)^(bma[i][0]>>15)^(bma[i][0]>>16))&1);

      /* Cr[8]...Cr[17] */
      eff_text |= ((bma[i][1]&0x01ff8000)>>14); /* eff_text =
0x000007ff */

      /* Cr[32], Cr[1]-Pr[5] a/h toi F8(Cr,K8)[17] - S1 */
      eff_text |= (bma[i][1]&0xf8000000)>>16;  /* eff_text =
0x0000ffff*/
      eff_text |= (bma[i][1]&0x00000001)<<16;    /* eff_text =
0x0001ffff */

      b = 0;

      /* Deal with Equation (2) */

      /* Pl[8,14,25]+Cr[3,8,14,25]+Cl[17]*/
      b |= (((bro[i][0]&0x01000000)>>24)^((bro[i][0]&0x00040000)>>18)
           ^((bro[i][0]&0x00000080)>>7) ^((bma[i][1]&0x20000000)>>29)
           ^((bma[i][1]&0x01000000)>>24)^((bma[i][1]&0x00040000)>>18)
           ^((bma[i][1]&0x00000080)>>7) ^((bma[i][0]&0x00008000)>>15));

      /* Pr[16]-Pr[21] a/h toi F1(Pr,K1)[8,14,25] - S5 */
      b |= (bro[i][1]&0x0001f800)>>10;  /* eff_text = 0x0000007f*/

      /* Cr[32], Cr[1]-Cr[5] a/h toi F8(Cr,K8)[17] - S1 */
      b |= (bma[i][1]&0xf8000000)>>20;  /* eff_text = 0x00000fff*/
      b |= (bma[i][1]&0x00000001)<<12;     /* eff_text = 0x00001fff */

      if (left_side_2(fixed_2, b)==sum2) {
          TD[eff_text] += 1;
      }
    }
}


/* Left Side of : Cl[16,17,20] + F8(Pr,K8)[16,17,20] = K7[25,26,29]
(11')  */
/*
    Description:
      k = 12 bit = [6 bit S3][6 bit S4]
      t = 11 bit = [10 bit S3,S4][1 bit tong]
 */
int leftside_11p(unsigned long k, unsigned long t)
{
    int tmp=0;
    UINT4 l=0x00000000L,r=0;
```

```
    UINT4 candidate[2], temp; /* = */

    tmp ^= (t&1); /* Cl[16,17,20]  */

    /* Chuyen 6 bit khoa a/h toi F1(Pr,K1)[17] - S1
       tim duoc tu phuong phap cua Matsui*/
    /* fixed_1 = [6bit K8] [6 bit K1] */
    candidate[0]  = candidate[1] = 0;
    candidate[0] |= (fixed_2&0x00000fc0)<<18; /* 6 bit dau vao S1 */

    /* Chuyen 6 bit khoa a/h toi F8(Cr,K8)[16] - S3 */
    candidate[0] |= (k&0x00000fc0)<<10; /* 6 bit dau vao S3 */

    /* Chuyen 6 bit khoa a/h toi F8(Cr,K8)[20] - S4 */
    candidate[1] |= (k&0x0000003f)<<16; /* 6 bit dau vao S4 */

    /* Chuyen 10 bit text a/h toi F8(Cr,K8)[16,20] vao r -
Cr[8]..Cr[17] */
    r |= ((t&0x000007fe)<<14);

    /* 6 bit text a/h toi F8(Pr,K8)[17] */
    r |= ((t&0x0000f800)<<16);
    r |= ((t&0x00010000)>>16);

    r      = (r<<1) | (r>>31);
    Ft(l,r,&candidate[0]);

    l      = (l<<31) | (l>>1);


    /* F8(Cr,K8)[16,17,20] */
    tmp ^= (((l>>12)^(l>>15)^(l>>16))&1);

    return tmp;
}

/* Counting up effective key bits (12 bits) */
void effective_key11p(void)
{
    unsigned long k /* 13 key bit */, t, m;
    int right;

    for (k=0; k<4096; k++) { /* 12 key bit */
    for (t=0; t<131072; t++) {
    for (right=0; right<2; right++) {
      m = k;
      m |= ((right&1)<<12);
      if (leftside_11p(k, t)==right) {
          U1[m] += TD[t];
      }
      else {
          U1[m] += 4*TD[t];
      }
    }
```

```
    }
    }

}

/* Find remaining key bits */
int exhaustive_search(unsigned char fk[8])
{
    unsigned long tmp, i;
    unsigned char testkey[8];
    UINT4 plaintext[2];

    for (tmp = 0; tmp<2097152; tmp++) {
      plaintext[0] = bro[0][0];plaintext[1] = bro[0][1];

        /* Copy found key to testkey */
      for (i=0; i<8; i++) testkey[i] = fk[i];

      /* Move tmp into testkey */
      testkey[0] |= (unsigned char)((tmp&1)<<3); /*K 5 */
      testkey[0] |= (unsigned char)((tmp&2)<<1); /*K 6 */
      testkey[1] |= (unsigned char)((tmp&4)<<5); /*K 9 */
      testkey[1] |= (unsigned char)((tmp&8)<<2); /*K 11 */
      testkey[1] |= (unsigned char)((tmp&0x00000010)>>2); /*K 14 */
      testkey[1] |= (unsigned char)((tmp&0x00000020)>>4); /*K 15 */
      testkey[2] |= (unsigned char)((tmp&0x00000040)>>2); /*K 20 */
      testkey[2] |= (unsigned char)((tmp&0x00000080)>>4); /*K 21 */
      testkey[2] |= (unsigned char)((tmp&0x00000100)>>7); /*K 23 */
      testkey[3] |= (unsigned char)((tmp&0x00000200)>>6); /*K 29 */
      testkey[4] |= (unsigned char)((tmp&0x00000400)>>3); /*K 33 */
      testkey[4] |= (unsigned char)((tmp&0x00000800)>>9); /*K 38 */
      testkey[5] |= (unsigned char)((tmp&0x00001000)>>6); /*K 42 */
      testkey[5] |= (unsigned char)((tmp&0x00002000)>>10) /*K 45 */;
      testkey[5] |= (unsigned char)((tmp&0x00004000)>>12) /*K 46 */;
      testkey[5] |= (unsigned char)((tmp&0x00008000)>>14) /*K 47 */;
      testkey[6] |= (unsigned char)((tmp&0x00010000)>>13) /*K 53 */;
      testkey[6] |= (unsigned char)((tmp&0x00020000)>>16) /*K 55 */;
      testkey[7] |= (unsigned char)((tmp&0x00040000)>>11) /*K 57 */;
      testkey[7] |= (unsigned char)((tmp&0x00080000)>>17) /*K 62 */;
      testkey[7] |= (unsigned char)((tmp&0x00100000)>>19) /*K 63 */;


      /* Call key schedule routine */
      khoades8rt(kst,testkey,1);

      des8r(plaintext,kst);


      /* Check corrective cipher text */
      if ((plaintext[0]==bma[0][0]) & (plaintext[1] ==bma[0][1])) { /*
found */
          /* Save the results to foundkey */
          for (i=0; i<8; i++) foundkey[i] = testkey[i];
          return 1;
```

```
      }
    }

    return 0;
}


int checkkey(unsigned long fk1, unsigned fk2)
{
    unsigned char temp_fk[8];
    int i;

    /* Copy cac bit khoa da tim duoc tu thuat toan cua Matsui */
    for (i=0; i<8; i++) temp_fk[i] = foundkey[i];

    /* Chuyen 12 bit khoa tu fk1 vao khoa da tim duoc foundkey */
    /* fk1 = tong  3  35  26  25  44  58  59  1  36(trung)  27  18
41(trung)*/
                                       /* K1[24] <-> K[41] trung */
    temp_fk[2] |= (unsigned char)((fk1&2)<<5);/* K1[23] <-> K[18] */
    temp_fk[3] |= (unsigned char)((fk1&4)<<3);/* K1[22] <-> K[27] */
                                     /* K1[21] <-> K[36] trung */
    temp_fk[0] |= (unsigned char)((fk1&0x00000010)<<3);/* K1[20] <->
K[1] */
    temp_fk[7] |= (unsigned char)(fk1&0x00000020);/* K1[19] <-> K[59]
*/
    temp_fk[7] |= (unsigned char)(fk1&0x00000040);/* K1[18] <-> K[58]
*/
    temp_fk[5] |= (unsigned char)((fk1&0x00000080)>>3);/* K1[17] <->
K[44] */
    temp_fk[3] |= (unsigned char)((fk1&0x00000100)>>1);/* K1[16] <->
K[25] */
    temp_fk[3] |= (unsigned char)((fk1&0x00000200)>>3);/* K1[15] <->
K[26] */
    temp_fk[4] |= (unsigned char)((fk1&0x00000400)>>5);/* K1[14] <->
K[35] */
    temp_fk[0] |= (unsigned char)((fk1&0x00000800)>>6);/* K1[13] <->
K[3] */


    /* Chuyen 12 bit khoa tu fk2 vao khoa da tim duoc */
    /* fk2 = tong */
    temp_fk[0] |= (unsigned char)((fk2&1)<<6);      /* K8[24] <-> K[2]
*/
                                         /* K8[23] <-> K[44] trung */
                                         /* K8[22] <-> K[17] trung */
                                         /* K8[21] <-> K[26] trung */
                                         /* K8[20] <-> K[27] trung */
                                         /* K8[19] <-> K[49] trung */
    temp_fk[2] |= (unsigned char)((fk2&0x00000040)>>1);/* K8[18] <->
K[19] */
                                         /* K8[17] <-> K[34] trung */
                                         /* K8[16] <-> K[51] trung */
```

```
    temp_fk[6] |= (unsigned char)((fk2&0x00000200)>>5);/* K8[15] <->
K[52] */
                                        /* K8[14] <-> K[25] trung */
                                        /* K8[13] <-> K[58] trung */

    /* Voi moi gia khoa co the, vet can so khoa con lai */
    printf("\n Trying to find correct key ... ");
    if (exhaustive_search(temp_fk)==1) /* tim duoc khoa dung */ {
      return 1;
    }

    return 0; /* correct key */
}


/* Rearrange U - Both UC and UD */
int rearrangeU(int a)
{
    unsigned long i, j, tg; unsigned long fkey1, fkey2;

    for (i=0; i<8192; i++) pos[i] = pos1[i] = i;

    /* Sap xep lai U0 theo do lon cua 5N/4 */
    for (i=0; i<8191; i++) {
      for (j=i+1; j<8192; j++) {
          if
 ((labs(U0[i]-(unsigned long)(5*N/4))) < (labs(U0[j]-(unsigned
long)(5*N/4)))) {
            tg    = U0[i];
            U0[i] = U0[j];
            U0[j] = tg;
            /* position */
            tg          = pos[i];
            pos[i]      = pos[j];
            pos[j]  = tg;
          }
      }
    }

    /* Sap xep lai U1 theo do lon cua 5N/4 */
    for (i=0; i<8191; i++) {
      for (j=i+1; j<8192; j++) {
          if
 ((labs(U1[i]-(unsigned long)(5*N/4))) < (labs(U1[j]-(unsigned
long)(5*N/4)))) {
            tg    = U1[i];
            U1[i] = U1[j];
            U1[j] = tg;
            /* position */
            tg          = pos1[i];
            pos1[i] = pos1[j];
            pos1[j] = tg;
          }
      }
```

```
    }

    /* try to find the correct key */
    for (i=0; i<16; i++)   {
      for (j=0; j<16; j++) {
          fkey1 = pos[i]; fkey2 = pos1[j];
          if (checkkey(fkey1, fkey2)==1) return 1;
      }
    }

    for (i=0; i<16; i++)   {
      for (j=8176; j<8192; j++) {
          fkey1 = pos[i]; fkey2 = pos1[j];
          if (checkkey(fkey1, fkey2)==1) return 1;
      }
    }

    for (i=8176; i<8192; i++)   {
      for (j=0; j<16; j++) {
          fkey1 = pos[i]; fkey2 = pos1[j];
          if (checkkey(fkey1, fkey2)==1) return 1;
      }
    }

    for (i=8176; i<8192; i++)   {
      for (j=8176; j<8192; j++) {
          fkey1 = pos[i]; fkey2 = pos1[j];
          if (checkkey(fkey1, fkey2)==1) return 1;
      }
    }

    /* Could'nt find a correct key */
    return 0;
}

/* Using quadratic relations to attack DES with non-linear and multiple
  linear cryptanalysis methods. Determize 2 following equations:
*/
void multiple_linear(int a)
{
    /* Initial */

    /* Counting up effective text bits (11 bits) */
    effective_text11();
    effective_text11p();

    /* Counting up effective key bits (12 bits) */
    effective_key11();
    effective_key11p();

    /* Guess 12 effective key bits by Multiple Linear Method */
    if (rearrangeU(a)==1) {
      printf("\n FOUND CORRECT KEY !\n");
    }
```

```
    else {
      printf("\n SORRY, I COULDN'T FIND CORRECT KEY !\n");
    }
}

void initial()
{
    long i;

    printf("\n Initialing the arrays ...");
    fixed_1=fixed_2=sum1=sum2=0;
    for (i=0; i<8192; i++) TA[i]=TB[i]=0;
    for (i=0; i<4096; i++) KA[i]=KB[i]=U0[i]=U1[i]=0;

    for (i=0; i<8; i++) foundkey[i] = 0;

    for (i=0; i<131072; i++) {
      TC[i]=TD[i]=0;
    }

}

/*
 *                THE MAIN OF PROGRAM
 */
int main()
{
    int m; /* So lan thu */
    struct timeval t1, t2;
    struct timezone tz;

    for (m=0; m<100; m++) {
    printf("\n LAN THU %d ", m);
    tz.tz_dsttime=0;

    if (gettimeofday(&t1, &tz)!=0) {
          printf("\n Can't get system time in sub function
part1_step2");
    }

    /* Create 8 random bytes key */
    create_randomKey();

    initial();

    /* Create random N Plain/Cipher text pairs */
    create_randomPC();

    matsui_linear();

    /* Using quadratic relations to attack DES with non-linear and
multiple
       linear cryptanalysis methods */
    multiple_linear(m);
```

```
    printResults(m);

    if (gettimeofday(&t2, &tz)!=0) {
          printf("\n Can't get system time in sub function
part1_step2");
    }

    print_time(t2.tv_sec-t1.tv_sec);
    } /* for m */

    return 1;
}
```

PHỤ LỤC B:
LISTING CHƯƠNG TRÌNH THUẬT TOÁN MÃ KHỐI MK_KC-01-01

```
/* Ma khoi cho KC01 - Theo so do MK_KC01 */

/* Phien ban 32 bit (VisualC6 - Linux)  */

/* Dang Van Truong - 18/03/2003     */

/* Modify: Tran Hong Thai - 06/06/2003 */

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>
#include <sys/time.h>

/*

#ifndef WIN32

#error "This module compile on 32 bit processor only"

#endif


*/

#define MK_BLOCK_SIZE         16
#define MK_KC01_KEY_LENGTH    16
#define ENCRYPT         1
#define DECRYPT         0
#define NUMBUF          1600

#define ROUND 16

#define M16       0x0000FFFF



#define M2_00     0x00000003

#define M2_02     0x0000000C

#define M2_04     0x00000030

#define M2_06     0x000000C0

#define M2_08     0x00000300

#define M2_10     0x00000C00
```

```
#define M2_12     0x00003000

#define M2_14     0x0000C000

#define M2_16     0x00030000

#define M2_18     0x000C0000

#define M2_20     0x00300000

#define M2_22     0x00C00000

#define M2_24     0x03000000

#define M2_26     0x0C000000

#define M2_28     0x30000000

#define M2_30     0xC0000000


struct mk_kc01_key_st {
    unsigned long roundkey[4*10 + 1];
};

typedef struct mk_kc01_key_st KC01_KEY;


/* UINT4 always 32 bit */

typedef unsigned long int UINT4;


static const unsigned int

S1[256]={

198, 62,238,222, 79,214,246,163, 12,128,230,212,127,235,213,110,

83, 65,147,190, 95,154,226, 15,181, 43, 69,245, 57, 37, 78,  3,

237,191,201,100,108,252,239, 51, 44,165,167,143,142, 27,140,168,

32,227,196,195, 24,105,160, 89,224, 72,  1, 71,215,228, 77,174,

144,193, 52, 88,216,118, 90,  5, 74,220,107,205,148,199,244, 33,

202,139,  0,183,  4, 63,141,218, 86,211,125,156, 82, 50, 26,243,

11,247, 85,223,194,178,204,161,162,159, 64,254, 10, 60,249, 21,

138,197,  2,241, 73,185, 28,175, 61,109, 91,132,  8,255,207, 75,

179, 48,200, 55,250,233, 34,232, 35,229,126,188, 38,186,152,206,

6,129,242, 59, 68, 84,  9, 17, 98,119, 29, 40,123,122,208,219,
```

```
7, 76, 92, 80,146, 96, 36, 58, 67,203, 53, 70,137,169, 39,158,
231, 19,236,182,177,171,114,149, 54,106, 47, 87,166, 94,117, 16,
93, 30,164,116, 56,101, 45, 99, 23,187, 46,248,210,189,209, 81,
14,124,173,102, 18,192,111,112,134,172,234,157, 97,131,184,121,
135, 31, 25,136,150,155,113, 41,217,120,225,151,115,170, 20,251,
49,133,145,176,253,103, 66, 22,130,153,180,240, 13, 42,221,104},
S2[256]={
218, 85,155, 91,195, 90, 26,206,145,252,154, 94,  3,239, 40,245,
64,196, 14, 59, 67,121,152,227, 72,161,198,104, 37,166,181,224,
233, 77,171,240,241, 31,237, 32,209,200,204,141,251, 97,255,189,
210,238,222,174, 19,133,188, 69,156,179,228,194, 44,158,199,187,
124,170, 80, 51, 93,116, 55,230,183, 95,129,169,126,172, 30,164,
217,143,146, 76,144, 35,137, 89, 52, 46,  7,127, 54, 86, 23,110,
225,108, 70, 45,216, 56,223,202,184, 13,178, 27,151, 81,107,102,
249,168,150,106,197, 75, 17,205, 39,135, 65,254,147,109,173,193,
78, 82,221, 34, 25,235,214,157,160,232,117, 63,212, 57,125,219,
148,138, 24, 33,176, 48,229,100,246,  2,103,211,  1,119, 92, 47,
226,177, 49, 50,120,242,208, 87,192,175, 38,180,139,203,162,123,
236, 96,159, 58, 74,207,118,  8, 84,247,163, 66,186, 53,  6, 18,
71, 21,190,112, 83,134,167,128, 98, 79,213, 29, 88, 73, 42, 68,
149,113,201,244, 22,220,131,114,250,191,153,  9,132,142, 61,  5,
140, 99,101,253,122, 15,  4,165, 43,115,234, 12,  0,185, 16,111,
36,136, 10, 60,105,130,182, 20,248, 11, 62, 28,231,215, 41,243},
LR[10]={5,23,23,5,23,5,5,5,23,5};


static const UINT4 KUV0[4]={0x3210,0x7654,0xBA98,0xFEDC};


#define S_BOX8(x,y) iw=(unsigned char*)(x); ow=(unsigned char*)(y); \
      *ow++ = (unsigned char)S1[*iw++]; *ow++ = (unsigned
char)S2[*iw++]; \
      *ow++ = (unsigned char)S1[*iw++]; *ow++ = (unsigned
char)S2[*iw++]; \
```

```
      *ow++ = (unsigned char)S1[*iw++]; *ow++ = (unsigned
char)S2[*iw++]; \
      *ow++ = (unsigned char)S1[*iw++]; *ow   = (unsigned
char)S2[*iw];


/**
 * Expand the cipher key into the encryption key schedule.
 */
int mk_kc01_set_encrypt_key(const unsigned char *userkey, KC01_KEY
*key)
{
    UINT4 register t0,t1;

      unsigned char *iw,*ow;

    UINT4 r0,r1,i,iround;

    UINT4 roundkey[40],CD[4],KUV[4];


    if (!userkey || !key)
      return -1;

    memcpy(CD,key,16);

    memcpy(KUV,KUV0,16);


    /* Calculate round key */

    for(i=0;i<10;i++) {

      /* Shift Left */

      t0=LR[i];

      t1=32-LR[i];

      r0=CD[0];

      CD[0] = (r0<<t0) | (CD[1]>>t1);

      CD[1] = (CD[1]<<t0) | (CD[2]>>t1);

      CD[2] = (CD[2]<<t0) | (CD[3]>>t1);

      CD[3] = (CD[3]<<t0) | (r0>>t1);


      /* XOR (KVi KUi) */

      CD[0] ^= KUV[0];

      CD[1] ^= KUV[1];

      CD[2] ^= KUV[2];
```

```
      CD[3] ^= KUV[3];

      /* Throught S-box */
      S_BOX8(CD,KUV);
      S_BOX8(CD+2,KUV+2);
      /* Store */
      if (i<6) {
            /* K5 ... K16 == roundkey[4*2] ... roundkey[15*2] */
            roundkey[i*4+ 8]=KUV[0];
            roundkey[i*4+ 9]=KUV[1];
            roundkey[i*4+10]=KUV[2];
            roundkey[i*4+11]=KUV[3];
      } else if (i<8) {
            /* K1 ... K4 == roundkey[0*2] ... roundkey[3*2] */
            roundkey[i*4-24]=KUV[0];
            roundkey[i*4-23]=KUV[1];
            roundkey[i*4-22]=KUV[2];
            roundkey[i*4-21]=KUV[3];
      } else {
            /* First and last key */
            roundkey[i*4  ]=KUV[0];
            roundkey[i*4+1]=KUV[1];
            roundkey[i*4+2]=KUV[2];
            roundkey[i*4+3]=KUV[3];
      }
    }

    return 0;
}

/* Ma hoa/ giai ma 1 block 128 bit, key 128 bit,
  fEnc = 0:dich, #: ma */
```

```
static void MK_KC01(char *data,KC01_KEY *key, int fEnc)
{
    UINT4 register t0,t1,*pl,*pr,*pk;
      unsigned char *iw,*ow;
    UINT4 r0,r1,i,iround;

    pl=(UINT4*)data;

    /* XOR first key */
    if (fEnc) {
            *pl++ ^= key->roundkey[32];
            *pl++ ^= key->roundkey[33];
            *pl++ ^= key->roundkey[34];
            *pl   ^= key->roundkey[35];
            pk=key->roundkey;
    } else {
            *pl++ ^= key->roundkey[36];
            *pl++ ^= key->roundkey[37];
            *pl++ ^= key->roundkey[38];
            *pl   ^= key->roundkey[39];
            pk=key->roundkey+((ROUND-1)*2);
    }

    pl=(UINT4*)data; pr=(UINT4*)(data+8); /* 8 byte = 64 bit = 2 word
32 */

    for (iround=0;iround<ROUND;iround++){
            /* 1 Round */
            r0=pr[0]; r1=pr[1];
            /* XOR key */
            pr[0] ^= pk[0]; pr[1] ^= pk[1];
```

```
            /* set pk point to key of _next_ round */
            if (fEnc)
                  pk += 2;
            else
                  pk += -2;

            /* Hoan vi qua cac hop S1 - S2 */
            S_BOX8(pr,pr);
            /*Thuc hien dich trai 23 bit ca so 64 bit */
            t0=pr[0]; t1=pr[1];
            pr[0]=(t0 << 23 ) | (t1 >> 9);
            pr[1]=(t1 << 23 ) | (t0 >> 9);
            /* Hoan vi qua cac hop S1 - S2 lan 2*/
            S_BOX8(pr,pr);
         /* Thuc hien hoan vi theo bang sau, dung toi t0,t1(UINT4) */
            /* #   0   2   4   6   8  10  12  14  16  18  20  22  24
26  28  30 */
            /* t0  {22, 42, 20, 10,  4,  0,  2, 32, 16,  8, 36, 18,
40, 52, 26, 12,*/
            /* t1 38, 50, 56, 60, 62, 30, 14,  6,     34, 48, 24, 44,
54, 58, 28, 46};*/
            t0=pr[0]; t1=pr[1];
            pr[0] = ((t0 << 22) & M2_22) | ((t0 << 16) & M2_20) |
                  ((t0 <<  4) & M2_10) | ((t0 >>  4) & M2_04) |
                  ((t0 >> 10) & M2_00) | ((t0 >> 10) & M2_02) |
                  ((t0      ) & M2_16) | ((t0 >> 10) & M2_08) |
                  ((t0 >>  4) & M2_18) | ((t0 >>  2) & M2_26) |
                  ((t0 >> 18) & M2_12);
            pr[0] |=((t1 << 20) & M2_30) | ((t1 <<  2) & M2_14) |
                  ((t1 >>  8) & M2_06) | ((t1 <<  4) & M2_24) |
                  ((t1      ) & M2_28);
```

```
            /* #   0   2   4   6   8  10  12  14  16  18  20  22  24
26  28  30 */

            /* t0  { -, 10,  -,  -,  -,  -,  -,  0,  -,  -,  4,  -,
8, 20,  -,  -,*/

            /* t1  6, 18, 24, 28, 30,  -,  -,  -,      2, 16,  -, 12,
22, 26,  -, 14};*/

            pr[1] = ((t0 <<  8) & M2_10) | ((t0 >> 14) & M2_00) |

                  ((t0 >> 16) & M2_04) | ((t0 >> 16) & M2_08) |

                  ((t0 >>  6) & M2_20);

            pr[1] |=((t1 <<  6) & M2_06) | ((t1 << 16) & M2_18) |

                  ((t1 << 20) & M2_24) | ((t1 << 22) & M2_28) |

                  ((t1 << 22) & M2_30) | ((t1 >> 14) & M2_02) |

                  ((t1 >>  2) & M2_16) | ((t1 >> 10) & M2_12) |

                  ((t1 >>  2) & M2_22) | ((t1      ) & M2_26) |

                  ((t1 >> 16) & M2_14);

            /* Last XOR */

            pr[0] ^= pl[0]; pr[1] ^= pl[1];

            pl[0] = r0;     pl[1] = r1;

      //    printf("%5d",iround);

      }

    /* Last revert */

/*    r0=pl[0]; already! */

    pl[0]=pr[0]; pl[1]=pr[1];

    pr[0]=r0;    pr[1]=r1;

    /* XOR last key*/

    if (fEnc) {

            *pl++ ^= key->roundkey[36];

            *pl++ ^= key->roundkey[37];

            *pl++ ^= key->roundkey[38];

            *pl   ^= key->roundkey[39];

    } else {
```

```
            *pl++ ^= key->roundkey[32];

            *pl++ ^= key->roundkey[33];

            *pl++ ^= key->roundkey[34];

            *pl   ^= key->roundkey[35];

    }

}


void mk_kc01_ecb_encrypt(const unsigned char *in, unsigned char *out,
                const unsigned long length, KC01_KEY *key,
                unsigned char *ivec, const int enc) {

      int n;
      unsigned long len = length;
      unsigned char tmp[16];

      assert(in && out && key && ivec);
      assert(length % MK_BLOCK_SIZE == 0);
      assert((ENCRYPT == enc)||(DECRYPT == enc));

      while (len > 0) {
            memcpy(out, in, 16);
            MK_KC01
(out, key, enc);
            len -= 16;
            in += 16;
            out += 16;
      }
}

unsigned char *
padbuf(unsigned char *inbuf, unsigned int inlen, unsigned int
*outlen)
{
    unsigned char *outbuf;
    unsigned int   mk_kc01_len;
    unsigned int   i;
    unsigned char  mk_kc01_pad_len;

    mk_kc01_len = (inlen + MK_BLOCK_SIZE) & ~(MK_BLOCK_SIZE - 1);

    outbuf = (unsigned char*)malloc(mk_kc01_len);
    memcpy(outbuf, inbuf, inlen);

    if (outbuf == NULL) {
      printf("\n padbuf: out of memory!");
      return NULL;
    }

    mk_kc01_pad_len = mk_kc01_len - inlen;

    for (i = inlen; i < mk_kc01_len; i++)
      outbuf[i] = 0x55;
```

```
    *outlen = mk_kc01_len;
    return outbuf;
}



int
main (int argc, char **argv)

{
    FILE *f, *fd;
    unsigned char buf[NUMBUF], out[NUMBUF], userkey[16], ivec[16],
tmp[NUMBUF];
    int n, i, padlen, encrypt;
    struct timeval t1, t2;
    struct timezone tz;
    KC01_KEY *key;

    key = malloc(sizeof(KC01_KEY));
    tz.tz_dsttime=0;

    if (gettimeofday(&t1, &tz)!=0) {
          printf("\n Can't get system time");
    }

    if (argc<5) {
        printf("\n Su dung: mk_kc01-cbc in out keyfile e/d\n");
      printf(" Ngam dinh la encrypt\n");
      exit(1);
    }

    if (strcmp(argv[4],"e")==0) encrypt = 1;
    if (strcmp(argv[4],"d")==0) encrypt = 0;

    if (!(f = fopen(argv[1], "rb"))) {
      fprintf(stderr, "Loi mo file %s\n", argv[1]);
      exit(1);
    }

    if (!(fd = fopen(argv[3], "rb"))) {
      fprintf(stderr, "Loi mo file %s de doc khoa\n", argv[3]);
      exit(1);
    }

    if ((n=fread(userkey, 1, 16, fd))<16) {
      fprintf(stderr, "Trong file %s khong du khoa (512 bits)\n",
argv[3]);
      exit(1);
    }

    fclose(fd);

    if (mk_kc01_set_encrypt_key(userkey, key)>0) {
      fprintf(stderr, "Key equal zero in set_encrypt_key()!\n");
      exit(1);
    }

    for (i=0; i<16; i++) ivec[i]=0;
```

```
    if (!(fd = fopen(argv[2], "w"))) {
      fprintf(stderr, "Loi mo file %s de ghi ma\n", argv[2]);
      exit(1);
    }

    while ((n=fread(buf, 1, NUMBUF, f))>0) {
      if (n % MK_BLOCK_SIZE != 0 ) {
          memcpy(buf,padbuf(buf, n, &padlen), padlen);
          n = padlen;
      }
      mk_kc01_ecb_encrypt(buf, out, n, key, ivec, encrypt);
      fwrite(out, 1, n, fd);
    }

    fclose(fd);
    fclose(f);

    if (gettimeofday(&t2, &tz)!=0) {
          printf("\n Can't get system time");
    }

    printf("\n Tong thoi gian tim khoa la: %d\n\n",t2.tv_sec-
t1.tv_sec);

    return 0;

}
```