HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

ĐỖ XUÂN CHỢ

BÀI GIẢNG
MẬT MÃ HỌC CƠ SỞ

Hà Nội, tháng 12 năm 2021

MỞ ĐẦU

Với xu thế bùng nổ phát triển công nghệ và các ứng dụng của chúng ta trong cuộc sống hiện nay thì vấn đề an toàn và bảo mật hệ thống thông tin đang trở nên cấp thiết hơn bao giờ hết. Có rất nhiều phương pháp, kỹ thuật, cách thức để bảo vệ hệ thống công nghệ thông tin trước các nguy cơ mất an toàn thông tin. Mỗi phương pháp và kỹ thuật có ưu điểm và nhược điểm riêng. Một phương pháp cơ bản để đảm bảo an toàn thông tin hiện nay là ứng dụng các kỹ thuật mật mã. Hiện nay, các ứng dụng của mật mã học được triển khai và áp dụng ở hầu hết các lĩnh vực và công nghệ trong đời sống. Chính vì vậy, việc nghiên cứu và tìm hiểu về mật mã học cũng như các ứng dụng của mật mã trong thực tế là vấn đề rất cần quan tâm hiện nay.

Môn học "Mật mã học cơ sở" là môn cơ sở chuyên ngành thuộc chương trình đào tạo đại học ngành An toàn thông tin của Học Viện Công Nghệ Bưu Chính Viễn Thông. Môn học cung cấp cho sinh viên các kiến thức cơ bản về mật mã học bao gồm: vai trò và tầm quan trọng của mật mã, khái niệm toán học trong mật mã; các thuật toán mã hóa thông dụng; các hàm băm; vấn đề thám mã. Bài giảng "Mật mã học cơ sở" được biên soạn trên cơ sở đề cương chi tiết môn học đã được duyệt và tổng hợp tài liệu từ nhiều nguồn nhằm cung cấp tài liệu phục vụ cho sinh viên nghiên cứu và học tập. Bài giảng được cấu trúc thành các chương như sau:

Chương 1: Tổng quan về mật mã học. Chương này cung cấp các kiến thức cơ bản liên quan đến mật mã học bao gồm: khái niệm, các thuật ngữ, lịch sử phát triển và một số ứng dụng cơ bản. Bên cạnh đó, chương 1 còn trình bày về một số lý thuyết toán học được ứng dụng trong lĩnh vực mật mã như: lý thuyết số, lý thuyết thông tin, độ phức tạp của thuật toán…

Chương 2: Mã hóa khóa đối xứng. Chương 2 cung cấp các kiến thức về hệ mật khóa đối xứng cũng như nguyên tắc mã hóa, giải mã của một số thuật toán mã hóa khóa đối xứng. Ngoài ra, chương 2 trình bày một số ưu và nhược điểm của các kỹ thuật mã hóa khóa đối xứng.

Chương 3: Mã hóa khóa bất đối xứng. Chương 3 trình bày một số kiến thức liên quan đến kỹ thuật mã hóa khóa bất đối xứng bao gồm: khái niệm, nguyên tắc mã hóa, giải mã, ưu điểm và nhược điểm. Bên cạnh đó, chương 3 đề cập một số ứng dụng của mã hóa khóa bất đối xứng trong thực tế.

Chương 4: Các hàm băm. Chương này cung cấp các kiến thức liên quan đến hàm băm bao gồm: định nghĩa, khái niệm, tính chất, vai trò của hàm băm. Ngoài ra, chương 4 còn trình bày về một số hàm băm phổ biến hiện nay cũng như nguyên tắc làm việc của một số hàm băm đó.

Chương 5: Thám mã. Chương này cung cấp các kiến thức cơ bản liên quan đến vấn đề thám mã bao gồm: định nghĩa, khái niệm, phân loại, cách thức tiến hành…. Ngoài ra, chương 5 còn trình bày một số kỹ thuật thám mã đối xứng và bất đối xứng đang được ứng dụng hiện nay như thám mã: vi sai, tuyến tính, nội suy, lựa chọn bản mã thích nghi....

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU VÀ HÌNH VẼ

DANH MỤC CÁC TỪ VIẾT TẮT

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ Tiếng Anh	Thuật ngữ Tiếng Việt
AES	Advanced Encryption Standard	Chuẩn mã hóa nâng cao
CBC	Cipher Block Chaining	Chế độ mã móc xích
CFB	Ciphe Feedback Mode	Chế độ mã phản hồi
CRHF	Collision resistant hash functions	Hàm băm chống đụng độ
DES	Data Encryption Standard	Chuẩn mã hóa dữ liệu
ECB	Electronic code book	Chế độ bảng tra mã điện tử
H	Hash function	Hàm băm
KDC	Key distribution center	Trung tâm phân phối khóa
KTC	Key translation center	Trung tâm dịch khóa
MAC	Message authentication codes	Mã xác thực thông điệp
MD	Message Digest	Tóm tắt thông điệp (có thể xem như một loại hàm băm)
MDC	Modification detection codes	Mã phát hiện sửa đổi
OWHF	One-way hash functions	Hàm băm một chiều
PKC	Public-key certificate	Chứng chỉ khóa công khai
SHA	Secure Hash Algorithm	Thuật toán băm an toàn
SPN	Substitution-Permutation Network	Mạng thay thế hoán vị

CHƯƠNG 1: TỔNG QUAN VỀ MẬT MÃ HỌC

Chương này cung cấp các kiến thức cơ bản liên quan đến mật mã học bao gồm: khái niệm, các thuật ngữ, lịch sử phát triển và một số ứng dụng cơ bản. Bên cạnh đó, chương 1 còn trình bày về một số lý thuyết toán học được ứng dụng trong lĩnh vực mật mã như: lý thuyết số, lý thuyết thông tin, độ phức tạp của thuật toán…

1.1. CÁC THUẬT NGỮ VÀ CÁC KHÁI NIỆM CƠ BẢN

Có nhiều định nghĩa khác nhau về mật mã học như: *Mật mã học là kỹ thuật, nghệ thuật viết các ký tự bí mật* (theo Webster's Revised Unabridged Dictionary). Hoặc *mật mã học là việc mã hóa dữ liệu mà nó chỉ có thể được giải mã bởi một số người chỉ định* (Free Online Dictionary of Computing)… Tuy nhiên, theo quan điểm của tác giả thì có thể hiểu một cách đơn giản về mật mã học như sau: *Mật mã học là một ngành khoa học chuyên nghiên cứu, áp dụng các phương pháp, kỹ thuật, thuật toán nhằm che giấu, xử lý hoặc biến đổi thông tin thông thường thành dạng không đọc trực tiếp được là văn bản mã hóa.* Để thực hiện được quy trình biến đổi từ thông tin thông thường thành văn bản mã hóa bao gồm nhiều công đoạn với nhiều thuật ngữ và các khái niệm liên quan. Dưới đây sẽ trình bày một số thuật ngữ và khái niệm cơ bản được sử dụng phổ biến nhất hiện nay.

- ***Người gửi, người nhận***

Người gửi (sender): là người chịu trách nhiệm gửi thông điệp được mã hóa cho các bên liên quan.

Người nhận (Receiver): là người có quyền hợp pháp nhận được thông điệp từ người gửi.

- ***Thông điệp và Mã hóa***

Thông điệp (plaintext): hay còn gọi là bản rõ là một dạng văn bản có nội dung mà Người gửi muốn gửi tới Người nhận. Thông điệp thường được ký hiệu là *M* (*message*) hoặc *P* (*plaintext*). *P hoặc M* có thể là chuỗi bit, một file văn bản, tập tin ảnh bitmap, một dòng âm thanh hay ảnh số…

Mã hóa (encryption): quá trình cải trang một thông điệp, che giấu nội dung thật sự của thông điệp để nội dung của thông điệp không bị lộ.

Bản mã (*ciphertext*): là kết quả của một thông điệp được mã hóa. Bản mã được ký hiệu là *C (ciphertext).*

Giải mã (decryption): là quá trình chuyển đổi từ bản mã trở lại thông điệp ban đầu.

Giả sử *E* là hàm mã hóa *(encrypt),* áp dụng trên *P* để tạo ra *C,* có biểu diễn toán học dạng:

$$E(P) = C \qquad (1.1)$$

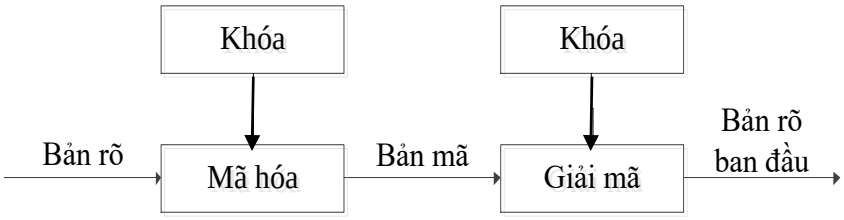
Trong quy trình ngược lại, *D* là hàm giải mã *(decrypt)* áp dụng trên *C* để có *P*:

$$D(C) = P \qquad (1.2)$$

Để đảm bảo quá trình từ mã hóa đến giải mã là đúng, cần đảm bảo:

$$D(E(P)) = P \qquad (1.3)$$

- ***Thuật toán và khóa***



Hình 1. 1. Mô hình mã hóa và giải mã đơn giản

Hình 1.1. thể hiện một mô hình mã hóa và giải mã đơn giản. Từ mô hình này có thể thấy một số khái niệm cụ thể như sau :

Thuật toán mã hóa (hay còn gọi là *giải thuật mã hóa*)*:* là các phương pháp, mô hình toán học được ứng dụng trong quá trình mã hóa thông tin.

Thuật toán giải mã (hay còn gọi là *giải thuật giải mã*)*:* là kỹ thuật, thuật toán, mô hình toán học được ứng dụng trong quá trình giải mã thông tin.

Khóa (key) *:* bản chất của Key là chìa. Tuy nhiên, thuật ngữ chìa không được sử dụng phổ biến mà thường sử dụng thuật ngữ Key là khóa. Chính vì vậy, trong bài giảng sẽ thống nhất sử dụng Key – khóa. Khóa là công cụ được ứng dụng trong quá trình mã hóa và giải mã. Khóa có thế là một phần tử trong một tập giá trị nào đó, phạm vi các giá trị của khóa được gọi là không gian khóa. Khóa dùng để mã hóa và giải mã có thể giống hoặc khác

nhau, tùy theo từng phương pháp, kỹ thuật mã hóa sử dụng. Các chương sau sẽ phân tích chi tiết hơn về vấn đề này.

- *Thám mã*: là ngành khoa học khôi phục lại thông điệp mà không cần biết khóa. Thám mã thành công có thể thu được thông điệp hoặc khóa hoặc cả hai. Nó cũng có thể tìm thấy điểm yếu của hệ mật mà đã để lộ ra các kết quả trước đó. Thám mã có thể chia thành 4 loại chính:

- Tấn công khi chỉ biết bản mã (ciphertext-only attack);
- Tấn công đã biết thông điệp (known-plaintext attack);
- Tấn công chọn thông điệp rõ (chosen plaintext attack);
- Tấn công chọn thông điệp thích hợp (adaptive-chosen-ciphertext attack).

- **Độ an toàn thuật toán:** Trong các nghiên cứu của Lars Knudsen đã trình bày chi tiết và cụ thể về độ an toàn của thuật toán mã hóa cũng như các tiêu chí đánh giá mức độ an toàn của một thuật toán. Dưới đây là một số tiêu chí cơ bản để đánh giá độ an toàn của một thuật toán mã hóa:

- Chi phí yêu cầu cho việc phá mã lớn hơn so với việc mã hóa thì thuật toán đó tương đối an toàn;
- Thời gian yêu cầu để phá mã dài hơn so với thời gian mà dữ liệu mã hóa cần giữ bí mật thì thuật toán đó là an toàn;
- Tổng lượng dữ liệu mã hóa bằng 1 khóa ít hơn lượng dữ liệu đưa vào để phá mã thì nó được coi là an toàn.

1.2. LỊCH SỬ PHÁT TRIỂN

Mật mã học là một ngành khoa học có lịch sử từ hàng nghìn năm nay. Trong phần lớn thời gian phát triển của mình, lịch sử mật mã học chính là lịch sử của những phương pháp mật mã học cổ điển. Vào đầu thế kỷ 20, sự xuất hiện của các cơ cấu cơ khí và điện cơ, chẳng hạn như máy Enigma, đã cung cấp những cơ chế phức tạp và hiệu quả hơn cho việc mật mã hóa. Bài giảng này sẽ trình bày một số cột mốc quan trọng trong lịch sử phát triển của mật mã học.

Mật mã học cổ điển: cách đây khoảng 4500 năm người Hy Lạp cổ đại đã sử dụng các kỹ thuật mật mã ví dụ như gậy mật mã. Cũng có những bằng chứng cho thấy người La Mã nắm được các kỹ thuật mật mã (mật mã *Caesar* và các biến thể).

Thời trung cổ: Nguyên do xuất phát có thể là từ việc phân tích bản kinh Koran, do nhu cầu tôn giáo mà kỹ thuật phân tích tần suất đã được phát minh để phá vỡ các hệ thống mật mã đơn ký tự vào khoảng năm 1000. Đây chính là kỹ thuật phá mã cơ bản nhất được sử dụng, mãi cho tới tận thời điểm thế chiến thứ II. Về nguyên tắc, mọi kỹ thuật mật mã đều không chống lại được kỹ thuật mã này cho tới khi kỹ thuật mật mã dùng nhiều bảng chữ cái được Alberti sáng tạo (năm 1465).

Mật mã học từ 1800 tới Thế chiến II: Tới thế kỷ 19, mật mã học mới được phát triển một cách có hệ thống. Trong thập niên 1840, Edgar Allan Poe đã xây dựng một số phương pháp có hệ thống để giải mật mã. Sau này ông có viết một đề tài về các phương pháp mã hóa và chúng trở thành những công cụ rất có lợi, được áp dụng vào việc giải mã của Đức trong Thế chiến II.

Mật mã học trong Thế chiến II: Trong thế chiến II, các hệ thống mật mã cơ khí và cơ điện tử được sử dụng rộng rãi. Người Đức đã sử dụng rộng rãi một hệ thống máy rôto cơ điện tử, dưới nhiều hình thức khác nhau, có tên gọi là máy Enigma. Quân đội Đức cũng cho triển khai một số thử nghiệm cơ học sử dụng thuật toán mật mã dùng một lần (one-time pad). Bộ ngoại giao của Nhật cũng cục bộ xây dựng một hệ thống dựa trên nguyên lý của "bộ điện cơ chuyển mạch dịch bước" (được Mỹ gọi là Purple), và đồng thời cũng sử dụng một số máy tương tự để trang bị cho một số tòa đại sứ Nhật Bản. Các máy mật mã mà phe Đồng Minh sử dụng trong thế chiến II, bao gồm cả máy TypeX của Anh và máy SIGABA của Mỹ, đều là những thiết kế cơ điện dùng rôto trên tinh thần tương tự như máy Enigma, song có nhiều nâng cấp lớn.

Mật mã học hiện đại: Năm 1949 Claude Shannon công bố bài: Lý thuyết về truyền thông trong các hệ thống bảo mật trên tập san Bell System Technical Journal - Tập san kỹ thuật của hệ thống Bell và Lý thuyết toán học trong truyền thông cùng với tác giả Warren Weaver. Ngày 17 tháng 3 năm 1975, IBM (International Business Machines) đề xuất Tiêu chuẩn mật mã hóa dữ liệu DES (Data Encryption Standard). Vào năm 1976, Whitfield

Diffie và Martin Hellman giới thiệu một phương pháp hoàn toàn mới về cách thức phân phối các khóa mật mã. Năm 2001, DES đã chính thức được thay thế bởi AES (Advanced Encryption Standard - Tiêu chuẩn mã hóa tiên tiến) khi NIST công bố phiên bản FIPS 197.

1.3. PHÂN LOẠI

Có nhiều phương pháp và tiêu chí để phân loại các phương pháp mã hóa. Dưới đây sẽ trình bày một số phương pháp phổ biến dùng để phân loại mã hóa.

Cách 1: Phân loại mã hóa theo đặc trưng của khoá:

Mã hóa khóa đối xứng (Hay còn gọi là mã hóa khóa bí mật): là phương pháp mã hóa mà quá trình mã hóa và giải mã dùng chung một khóa. Khóa được dùng trong mã hóa đối xứng gọi là khóa bí mật. Các thuật toán mã hóa đối xứng thường gặp: DES, AES…

Mã hóa khóa bất đối xứng (Hay còn gọi là mã hóa khóa công khai): là phương pháp mã hóa mà khóa sử dụng để mã hóa sẽ khác với khóa dùng để giải mã. Khóa dùng để mã hóa gọi là khóa công khai và khóa dùng để giải mã gọi là khóa bí mật. Tất cả mọi người đều có thể biết được khóa công khai (kể cả hacker), và có thể dùng khóa công khai để mã hóa thông tin. Nhưng chỉ có người nhận mới nắm giữ khóa bí mật, nên chỉ có người nhận mới có thể giải mã được thông tin.

Cách 2: Phân loại mã hóa theo đặc trưng xử lý thông điệp

Mã hóa dòng: là kiểu mã hóa mà từng bit (hoặc từng ký tự) của thông điệp được kết hợp với từng bit (hoặc từng ký tự) tương ứng của khóa để tạo thành bản mã. Một số thuật toán mã hóa dòng phổ biến hiện nay: A5/1; RC4. Trong chương 3 của bài giảng sẽ trình bày chi tiết hơn về các thuật toán này.

Mã hóa khối: là kiểu mã hóa mà dữ liệu được chia ra thành từng khối có kích thước cố định để mã hóa.

1.4. VAI TRÒ VÀ ỨNG DỤNG CỦA MẬT MÃ

1.4.1. Vai trò

Mật mã học có vai trò rất lớn trong việc đảm bảo an toàn cho hệ thống thông tin cũng như các ứng dụng hiện nay. Việc ứng dụng mật mã học nhằm đảm bảo các tính chất:

- Tính bí mật (confidentiality): thông tin chỉ được phép truy cập bởi những đối tượng hợp lệ, những đối tượng được cấp phép.
- Tính toàn vẹn thông tin (integrity): đảm bảo thông tin không bị thay đổi trong quá trình truyền tin; hoặc nếu có thay đổi thì sẽ bị phát hiện.
- Tính sẵn sàng (availability): đảm bảo thông tin phải luôn luôn sẵn sàng khi cần thiết. Điều đó có nghĩa rằng mật mã học cung cấp một cơ chế bảo mật tốt, có thể tránh được những rủi ro cả về phần cứng, phần mềm và tránh được các cuộc tấn công của tin tặc.
- Tính xác thực (authentication): đảm bảo các bên liên quan nhận biết và tin tưởng nhau trong một hệ truyền tin, đồng thời đảm bảo thông tin trao đổi là thông tin thật.
- Tính chống chối bỏ (non-repudiation): đảm bảo rằng các bên liên quan không thể chối bỏ các hành động đã thực hiện trước đó.

1.4.2. Một số ứng dụng của mật mã

Các giải thuật mã hóa hiện nay được ứng dụng ở hầu hết các lĩnh vực khác nhau trong đời sống hàng ngày. Một số ứng dụng phổ biến của mã hóa như:

- Các dịch vụ xác thực: chữ ký số; điều khiển truy nhập, phân phối khóa.
- Các giao thức an toàn mạng như: SSL/TLS, SSH, SET, IPSec; VPN; các chuẩn đảm bảo an toàn cho truyền thông không dây.
- Các nền tảng bảo mật dịch vụ: tường lửa, mã hóa email, file, mã hóa tin nhắn, thẻ thông minh....

1.5. MỘT SỐ KHÁI NIỆM TOÁN HỌC TRONG MẬT MÃ

Có thể thấy rằng mật mã học là một lĩnh vực khoa học rộng lớn có liên quan rất nhiều ngành toán học như: Đại số tuyến tính, Lý thuyết thông tin, Lý thuyết độ phức tạp tính toán.... Trong bài giảng này, tác giả sẽ mô tả ngắn gọn một số khái niệm toán học chủ yếu và thường được áp dụng để xây dựng các phương pháp mã hóa.

1.5.1. Lý thuyết thông tin

1.5.1.1. Độ an toàn của hệ mật

Có 2 quan điểm cơ bản về độ an toàn của một hệ mật.

1. **Độ an toàn tính toán:** Độ đo này liên quan đến những nỗ lực tính toán cần thiết để phá một hệ mật. Một hệ mật là an toàn về mặt tính toán nếu một thuật toán tốt nhất để

phá nó cần ít nhất *N* phép toán, *N* là số rất lớn nào đó. Vấn đề ở chỗ, không có một hệ mật thực tế đã biết nào có thể được chứng tỏ là an toàn theo định nghĩa này. Trên thực tế, gọi một hệ mật là "an toàn về mặt tính toán" nếu có một phương pháp tốt nhất phá hệ này nhưng yêu cầu thời gian lớn đến mức không chấp nhận được.

Một quan điểm chứng minh về độ an toàn tính toán là quy độ an toàn của một hệ mật về một bài toán đã được nghiên cứu và bài toán này được coi là khó. Ví dụ, có thể chứng minh một khẳng định: "Một hệ mật đã cho là an toàn nếu không thể phân tích ra thừa số một số nguyên *n* cho trước". Các hệ mật loại này đôi khi gọi là "An toàn chứng minh được". Tuy nhiên cần phải hiểu rằng, quan điểm này chỉ cung cấp một chứng minh về độ an toàn có liên quan để một bài toán khác chứ không phải là một chứng minh hoàn chỉnh về độ an toàn.

2. Độ an toàn không điều kiện: Độ đo này liên quan đến độ an toàn của các hệ mật khi không có một hạn chế nào được đặt ra về khối lượng tính toán mà kẻ tấn công được phép thực hiện. Một hệ mật được gọi là an toàn không điều kiện nếu nó không thể bị phá trong trường hợp lý tưởng (với khả năng tính toán không hạn chế).

Mặt khác độ an toàn không điều kiện của một hệ mật không thể được đánh giá theo quan điểm độ phức tạp tính toán vì thời gian tính toán cho phép không hạn chế. Ở đây lý thuyết xác suất là nền tảng thích hợp để nghiên cứu về độ an toàn không điều kiện.

Định nghĩa 1.1:

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

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

$$p(x,y)= p(x|y)\ p(y) \qquad (1.8)$$

Đổi chỗ *x* và *y* có:

$$p(x,y)= p(y|x)\ p(x) \qquad (1.9)$$

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

Định lý 1.1: (Định lý Bayes)

Nếu $p(y) > 0$ thì:

$$\rho\left(x/y\right) = \frac{p(x)\,p(y/x)}{p(y)} \qquad (1.10)$$

Hệ quả 1.1. *x và y là các biến độc lập khi và chỉ khi:* $p(x|y)= p(x)\ ,\forall\ x,y.$

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

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

$$C(K) =\{e_K(x): x\in P\} \qquad (1.11)$$

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

$$p_C(y)= \sum_{\{K:y\in C(K)\}} p_K(K)\ p_P(d_K(y)) \qquad (1.12)$$

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

$$p_C(y \mid x) = \sum_{\{K:x=d_K(y)\}} p_K(K) \qquad (1.13)$$

Từ đó có thể tính được xác suất có điều kiện $p_P(x|y)$ (tức xác suất để x là thông điệp với điều kiện y là bản mã) bằng cách dùng định lý Bayes. Công thức thu được như sau:

$$p_P(y \mid x) = \frac{p_P(x) = \sum_{\{K:x=d_K(y)\}} p_K(K)}{\sum_{\{K:y\in C(K)\}} p_K(K)\, p_P(d_K(y))} \qquad (1.14)$$

Các phép tính này có thể thực hiện được nếu biết được các phân bố xác suất. Sau đây sẽ trình bày một ví dụ đơn giản để minh hoạ việc tính toán các phân bố xác suất này.

Ví dụ:

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

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

Giả sử $C=\{1,2,3,4\}$ và các hàm mã được xác định là:

$e_{K_1}(a)=1, e_{K_1}(b)=2, e_{K_2}(a)=2, e_{K_2}(b)=3, e_{K_3}(a)=3, e_{K_3}(b)=4$.

Hệ mật này được biểu thị bằng ma trận mã hóa sau:

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

Tính phân bố xác suất p_C có:

$p_C(1)=1/8$

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

$p_C(3)=3/16+1/16=1/4$

$p_C(4)=3/16$

Bây giờ đã có thể các phân bố xác suất có điều kiện trên thông điệp với điều kiện đã biết bản mã. Từ đó có:

$p_P(a|1)=1$ $p_P(b|1)=0$ $p_P(a|2)=1/7$ $p_P(b|2)=6/7$

$p_P(a|3)=1/4$ $p_P(b|3)=3/4$ $p_P(a|4)=0$ $p_P(b|1)=1$

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

Định nghĩa 1.2.

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

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

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

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

Định lý 1.2.

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

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

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

Tuy nhiên, giả sử rằng $|C|= |K|$, bởi vậy phải có:

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

Tức là ở đây không tồn tại hai khoá *K1* và *K2* khác nhau để $e_{K1}(x)=e_{K2}(x)=y$. Như vậy đã chứng minh được rằng, với bất kỳ $x \in P$ và $y \in C$ có đúng một khoá K $e_K(x)=y$.

Ký hiệu $n=|K|$. Giả sử $P=\{x_i: 1 \le i \le n\}$ và cố định một giá trị $y \in C$. Có thể ký hiệu các khoá *K1, K2,...Kn* sao cho $e_{Ki}(x_i)=y_i$, $1 \le i \le n$. Sử dụng định lý Bayes có:

$$p_P(x_i \mid y) = \frac{p_C(y \mid x_i)p_P(x_i)}{p_C(y)} = \frac{p_K(K_i).(p_P(x_i))}{p_C(y)}$$

(1.15)

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

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

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

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

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

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

1.5.1.2. Entropy

Phần trước đã trình bày về khái niệm độ mật hoàn thiện và thể hiện mối quan tâm vào một trường hợp đặc biệt, khi một khoá chỉ được dùng cho một lần mã. Trong phần này, bài giảng sẽ nghiên cứu khi có nhiều thông điệp được mã hóa bằng cùng một khoá và bằng

cách nào mà thám mã có thể thực hiện có kết quả phép tấn công chỉ với bản mã trong thời gian đủ lớn. Công cụ cơ bản trong nghiên cứu bài toán này là khái niệm Entropy. Đây là khái niệm trong lý thuyết thông tin do Shannon đưa ra vào năm 1948. Có thể coi Entropy là đại lượng đo thông tin hay còn gọi là độ bất định. Nó được tính như một hàm của phân bố xác suất.

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

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

Định nghĩa 1.3

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

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 P_i \tag{1.16}$$

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

$$H(X) = -\sum_{i=1}^{n} p(X = x_i) \log_2 P(X = x_i) \tag{1.17}$$

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

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

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

Các tính chất của Entropy: Trong phần này sẽ chứng minh một số kết quả quan trọng liên quan đến Entropy. Trước tiên sẽ nghiên cứu về bất đẳng thức Jensen. Bất đẳng thức Jensen có liên quan đến hàm lồi có định nghĩa như sau:

Định nghĩa 1.4

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

$$f\left(\frac{x+y}{2}\right)\ge\frac{f(x)+f(y)}{2} \qquad (1.18)$$

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

$$f\left(\frac{x+y}{2}\right)>\frac{f(x)+f(y)}{2} \qquad (1.19)$$

với mọi $x,y\in I$, $x\ne y$.

Định lý 1.3*: (Bất đẳng thức Jensen không chứng minh)*

Giả sử h là một hàm lồi thực sự và liên tục trên khoảng l, $\sum_{i=1}^{n} a_i = 1$ *và* $a_i>0$; $1\le I\le n$.

Khi đó:

$$\sum_{i=1}^{n} a_i f(x_i)\le f\left(\sum_{i=1}^{n} a_i x_i\right) \qquad (1.20)$$

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

Tiếp theo sẽ đưa ra một số kết quả về Entropy. Trong định lý 1.4 sử dụng khẳng định: hàm $\log_2 x$ là một hàm lồi thực sự trong khoảng $(0,\infty)$..

Định lý 1.4

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

Chứng minh:

Áp dụng bất đẳng thức Jensen có:

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

$$\le \log_2 \sum_{i=1}^{n} (p_i \times 1/p_i) = \log_2 n$$

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

Định lý 1.5

$$H(X,Y) \le H(X) + H(Y) \qquad (1.21)$$

Đẳng thức xảy ra khi và chỉ khi X và Y là các biến cố độc lập

Chứng minh:

Giả sử X nhận các giá trị $x_i, 1 \le i \le m$; Y nhận các giá trị $y_j, 1 \le j \le n$.

Ký hiệu:

$$p_i = p(X = x_i), 1 \le i \le m$$

$$q_i = p(Y = y_j), 1 \le j \le n$$

.

$$r_{ij} = p(X = x_i, Y = y_j), 1 \le i \le, 1 \le j \le n$$ (Đây là phân bố xác suất).

Nhận thấy rằng:

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

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

Có:

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

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

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

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

Kết hợp lại được kết quả sau:

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

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

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

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

Điều này dẫn đến c=1. Bởi vậy đẳng thức xảy ra khi và chỉ khi $r_{ij}=p_i q_j$, nghĩa là:

$$P(X=x_i,Y=y_j)=p(X=x_i)p(Y=y_j) \text{ với } 1\le i\le m,\ 1\le j\le n.$$

Điều này có nghĩa là X và Y độc lập.

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

Định nghĩa 1.5

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

$$H(X \mid y) = -\sum_{x} p(x \mid y) \log_2 p(x \mid y) \qquad (1.22)$$

Định nghĩa 1.6

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

$$H(X \mid Y) = -\sum_{y}\sum_{x} p(y)\,p(x \mid y)\log_2 p(x \mid y) \qquad (1.23)$$

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

Sau đây là 2 kết quả trực tiếp.

Định lý 1.6

$$H(X,Y)=H(Y)+H(X|Y) \qquad (1.24)$$

Hệ quả 1.2

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

Dấu "=" xảy ra khi và chỉ khi X và Y độc lập.

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

Định lý 1.7

Giả sử (P, C, K, E, D) là một hệ mật. Khi đó:

$$H(K|C) = H(K) + H(P) - H(C) \qquad (1.25)$$

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

Ví dụ, giả sử Oscar thu được một xâu bản mã **WNAJW** mã bằng phương pháp mã dịch vòng (**MDV**). Dễ dàng thấy rằng, chỉ có hai xâu thông điệp có ý nghĩa là *river* và *arena* tương ứng với các khoá *F(= 5)* và *W(= 22).* Trong hai khoá này chỉ có một khoá đúng, khoá còn lại là khoá giả. Trên thực tế, việc tìm một bản mã của **MDV** có độ dài 5 và

2 bản giải mã có nghĩa không phải quá khó khăn, bạn đọc có thể tìm ra nhiều ví dụ khác. Mục đích của là phải tìm ra giới hạn cho số trung bình các khoá giả. Trước tiên, phải xác định giá trị này theo Entropy (cho một ký tự) của một ngôn ngữ tự nhiên **L** (ký hiệu là $\mathbf{H_L}$). $\mathbf{H_L}$ là lượng thông tin trung bình trên một ký tự trong một xâu có nghĩa của thông điệp. Chú ý rằng, một xâu ngẫu nhiên các ký tự của bảng chữ cái sẽ có Entropy trên một ký tự bằng $\log_2 26 \approx 4{,}76$. Có thể lấy $H(P)$ là xấp xỉ bậc nhất cho H_L. Trong trường hợp L là Anh ngữ, tính được $H(P) \approx 4{,}19$. Dĩ nhiên các ký tự liên tiếp trong một ngôn ngữ không độc lập với nhau và sự tương quan giữa các ký tự liên tiếp sẽ làm giảm Entropy. Ví dụ, trong Anh ngữ, chữ Q luôn kéo theo sau là chữ U. Để làm xấp xỉ bậc hai, tính Entropy của phân bố xác suất của tất cả các bộ đôi rồi chia cho 2. Một cách tổng quát, định nghĩa P^n là biến ngẫu nhiên có phân bố xác suất của tất cả các bộ n của thông điệp. Sẽ sử dụng tất cả các định nghĩa sau:

Định nghĩa 1.7

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

$$H_L = \lim_{n\to\infty} \frac{H(P^n)}{n} \qquad (1.26)$$

Độ dư của L là:

$$R_L = 1 - (H_L / \log_2 |P|) \qquad (1.27)$$

Nhận xét: H_L đo Entropy trên mỗi ký tự của ngôn ngữ L. Một ngôn ngữ tự nhiên sẽ có Entropy là $\log_2|P|$. Bởi vậy đại lượng R_L đo phần "ký tự vượt trội" là phần dư. Trong trường hợp Anh ngữ, dựa trên bảng chứa một số lớn các bộ đôi và các tần số, có thể tính được $H(P^2)$. Ước lượng theo cách này, tính được $H(P^2) \approx 3{,}90$. Cứ tiếp tục như vậy bằng cách lập bảng các bộ ba v.v... thu được ước lượng cho H_L. Trên thực tế, bằng nhiều thực nghiệm khác nhau, có thể đi tới kết quả sau $1{,}0 \le H_L \le 1{,}5$. Tức là lượng thông tin trung bình trong tiếng Anh vào khoảng 1 bit tới 1,5 bit trên mỗi ký tự.

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

tiếng Anh xuống còn 1/4 độ dài của bản gốc). Với các phân bố xác suất C^n đã cho trên K và P^n. Có thể xác định phân bố xác suất trên là tập các bộ n của bản mã. Đã xác định P^n là biến ngẫu nhiên biểu diễn bộ n của thông điệp. Tương tự C^n là biến ngẫu nhiên biểu thị bộ n của bản mã.

Định lý 1.8

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

$$\bar{s} \geq \{|K|/(|P|nR_L)\} - 1$$

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

Định nghĩa 1.8

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

1.5.2. Lý thuyết số

1.5.2.1. Số nguyên

Tập các số nguyên:*{..., -3, -2, -1, 0, 1, 2, 3,...}=**Z***

Định nghĩa 1.9

Cho $a,b \in$ **Z**, *a* là ước nguyên của *b*, nếu $\exists c \in Z : b = a.c$. *Ký hiệu* $a|b$

Các tính chất chia hết*:* $\forall a,b,c \in Z$ *có:*

- ✓ $a|a$
- ✓ *Nếu* $a|b$ *và* $b|c$ *thì* $a|c$
- ✓ *Nếu* $a|b$ *và* $a|c$ *thì* $a|(bx+cy)$ $\forall x, y \in Z$

- ✓ *Nếu a|b và b|a thì a= ± b*

Định nghĩa 1.10: Thuật toán chia đối với các số nguyên:

Nếu a và b là các số nguyên với b ≥1 thì a = qb +r , 0 ≤r <b q và r là duy nhất.

Phần dư của phép chia a và b được ký hiệu a mod b = r.

Thương của phép chia a và b được ký hiệu a div b = q.

Có a div b $= \left[\frac{a}{b}\right]$, *a mob b= a-* $b\left[\frac{a}{b}\right]$

Ví dụ: *a=73, b=17* $\Rightarrow$ *73 div 17=4, 73 mod 17=5*

Định nghĩa 1.11: Ước chung c là ước chung của a và b nếu c|a và c|b

Định nghĩa 1.12: Ước chung lớn nhất(ƯCLN)

Số nguyên dương d là ƯCLN của các số nguyên a và b, ký hiệu d=(a,b) nếu:

- ✓ *d là ước chung của a và b.*
- ✓ *Nếu có c|a và c|b thì c|d*

Như vậy *(a,b)* là số nguyên dương lơn nhất là ước của cả *a* và *b* không kể *(0,0)=0.*

Ví dụ: Các ước chung của 12 và 18 là $\{\pm1,\pm2,\pm3,\pm6\}$ → *(12,18)=6*

Định nghĩa 1.13: Bội chung nhỏ nhất (BCNN)

Số nguyên dương d là BCNN của các số nguyên a và b, ký hiệu d= BCNN(a,b) nếu:

- ✓ *a|d, b|d.*
- ✓ Nếu có *a|c, b|c thì d|c*

Như vậy *d* là số nguyên dương nhỏ nhất là bội của cả *a* và *b.*

Tính chất:

$$BCNN(a,b)=\frac{a.b}{(a,b)}$$

Ví dụ:(*12,18) =6* $\Rightarrow$ *BCNN(12,18)=* $\frac{12.18}{6}$ *=36*

Định nghĩa 1.14: Hai số nguyên dương a và b được gọi là nguyên tố cùng nhau nếu (a,b)=1.

Định nghĩa 1.15: Số nguyên p≥2 được gọi là số nguyên tố nếu các ước dương của nó chỉ là 1 và *p*. Ngược lại *p* được gọi là hợp số.

Định lý 1.9: (Định lý cơ bản của số học) Với mỗi số nguyên n≥2 luôn phân tích được dưới dạng tích của lũy thừa các số nguyên tố.

$$n = p_1^{e_1} p_2^{e_2} ... p_k^{e_k} \qquad (1.28)$$

Trong đó p_i là các số nguyên tố khác nhau và e_i là các số nguyên dương. Hơn nữa phân tích trên là duy nhất.

Định nghĩa 1.16: Với n≥2, hàm $\varphi(n)$ được xác định là các số nguyên trong khoảng [1.n] nguyên tố cùng nhau với *n*.

Các tính chất của hàm $\varphi(n)$

- Nếu *p* là số nguyên tố thì $\varphi(p) = p - 1$
- Nếu *(m,n)=1* thì $\varphi(m.n) = \varphi(m).\varphi(n)$
- Nếu $n = p_1^{e_1} p_2^{e_2} ... p_k^{e_k}$ là phân tích ra thừa số nguyên tố của *n* thì:

$$\varphi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2})...(1 - \frac{1}{p_k}) \qquad (1.29)$$

1.5.2.2. Khái niệm đồng dư modulo

Giả sử *a* và *b* là các số nguyên và *m* là một số nguyên dương. Khi đó viết *a≡b(mod m)* nếu *a* và *b* khi chia cho *m* có cùng số dư, tức là *(a-b)* chia hết cho *m*. Mệnh đề *a≡b(mod m)* được gọi là "*a* đồng dư với *b* theo mođun *m*".

Chứng minh: Giả sử chia *a* và *b* cho *m* và thu được thương nguyên và phần dư, các phần dư nằm giữa 0 và *m-1*. Nghĩa là:

$$a = q_1 m + r1 \text{ và } b = q_2 m + r2$$

Trong đó : ***0***≤*r1*≤*m-1 và 0*≤*r2*≤*m-1*. Khi đó có thể dễ dàng thấy rằng: *a ≡ b(mod m) khi và chỉ khi r1 = r2*

Tiếp theo sẽ dùng ký hiệu *a mod m* để xác định phần dư khi *a* được chia cho *m* (chính là giá trị *r1* ở trên). Như vậy: *a≡b(mod m)* khi và chỉ khi: *(a mod m) = (b mod m)*

Nếu thay giá trị của *a* bằng giá trị *(a mod m)* thì nói *a được rút gọn theo modulo m*

1.5.2.3. Số học modulo

Định nghĩa số học mođun m:

Z_m được coi là tập hợp *{0,1,…,m-1}* có trang bị hai phép toán cộng và nhân. Việc cộng và nhân trong Z_m được thực hiện giống như cộng và nhân các số thực ngoại trừ một điểm là các kết quả được rút gọn theo mođun *m*.

Ví dụ: Z_{26} được coi là tập hợp *{0,1, 2…,25}* có trang bị hai phép toán cộng và nhân.

25+7 = 32 mod 26 = 6 (ví dụ K=7, 'Z' → 'H')

*5*9 = 45 mod 26 = 19*

Định lý về đồng dư thức

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

Chứng minh:

- Giả sử rằng, *UCLN(a,m) = d >1*.
- Với *b = 0* thì đồng dư thức $ax \equiv 0$ (*mod m*) sẽ có ít nhất hai nghiệm phân biệt trong Z_m là *x = 0 và x = m/d*.

Khái niệm phần tử nghịch đảo:

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

$$aa^{-1} \equiv a^{-1}a \equiv 1 \ (mod\ m)$$

Tính chất

- *a* có nghịch đảo theo mođun *m* khi và chỉ khi *UCLN(a,m) = 1,*
- Nếu nghịch đảo tồn tại thì phải là duy nhất.
- Nếu $b = a^{-1}$ thì $a = b^{-1}$.
- Nếu *m* là số nguyên tố thì mọi phần tử khác không của Z_m đều có nghịch đảo.

1.5.3. Độ phức tạp thuật toán

1.5.3.1. Khái niệm về thuật toán

Có thể định nghĩa thuật toán theo nhiều cách khác nhau. Ở đây sẽ hiểu khái niệm thuật toán theo một cách thông thường nhất.

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

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

Cho *n* số *X[1], X[2],..., X[n],* cần tìm *m* và *j* sao cho: $m=X[j] = \max_{1\le k\le n} X[k]$ và *j* là lớn nhất có thể. Điều đó có nghĩa là cần tìm cực đại của các số đã cho và chỉ số lớn nhất trong các số cực đại. Với mục tiêu tìm số cực đại với chỉ số lớn nhất, xuất phát từ giá trị *X [n]*. Bước thứ nhất, vì mới chỉ có một số có thể tạm thời xem *m = X [n]* và *j = n*. Tiếp theo so sánh *X [n*] với *X[n-1]*. Nếu *X[n]* không nhỏ hơn *X [n-1]* thì giữ nguyên, trong trường hợp ngược lại, *X[n-1]* chính là số cực đại trong hai số đã xét và phải thay đổi *m* và *j*. Đặt *m = X [n-1], j=1,...,n-1*. Với cách làm như trên, ở mỗi bước luôn nhận được số cực đại trong số những số đã xét. Bước tiếp theo là so sánh nó với những số đứng trước hoặc kết thúc thuật toán trong trường hợp không còn số nào đứng trước nó.

1.5.3.2. Độ phức tạp của thuật toán

Thời gian làm việc của máy tính khi chạy một thuật toán nào đó không chỉ phụ thuộc vào thuật toán mà còn phụ thuộc vào máy tính được sử dụng. Vì thế, để có một tiêu chuẩn chung, sẽ đo độ phức tạp của một thuật toán bằng số các phép tính phải làm khi thực hiện thuật toán. Khi tiến hành cùng một thuật toán, số các phép tính phải thực hiện còn phụ thuộc vào cỡ của bài toán, tức là độ lớn của đầu vào. Vì thế độ phức tạp của thuật toán sẽ là một hàm số của độ lớn đầu vào. Trong những ứng dụng thực tiễn, không cần biết chính xác hàm này mà chỉ cần biết "cỡ" của chúng, tức là cần có một ước lượng đủ tốt của chúng. Trong khi làm việc, máy tính thường ghi các chữ số bằng bóng đèn "sáng, tắt", bóng đèn sáng chỉ số 1, bóng đèn tắt chỉ số 0. Vì thế để thuận tiện nhất là dùng hệ đếm cơ số 2, trong đó để biểu diễn một số, chỉ cần dùng hai ký hiệu 0 và 1. Một ký hiệu 0 hoặc 1 được gọi là 1bit

"viết tắt của binary digit". Một số nguyên n biểu diễn bởi *k* chữ số 1 và 0 được gọi là một số *k* bit .

Độ phức tạp của một thuật toán được đo bằng số các phép tính bit. Phép tính bit là một phép tính logic hay số học thực hiện trên các số một bit 0 và 1. Để ước lượng độ phức tạp của thuật toán dùng khái niệm bậc *O* lớn.

Định nghĩa: Giả sử *f[n]* và *g[n]* là hai hàm xác định trên tập hợp các số nguyên dương. Nói *f[n]* có bậc *O* lớn của *g[n]* và viết *f[n] =O(g[n])*, nếu tồn tại 1 số *C>0* sao cho với *n* đủ lớn, các hàm *f[n]* và *g[n]* đều dương thì *f[n] < C g[n]*.

Ví dụ:

- Giả sử f[n] là đa thức có công thức: $f[n] = a_d n^d + a_{d-1} n^{d-1} + ... + a_1 n + a_0$ trong đó $a_d > 0$. Dễ dàng chứng minh được $f[n] = O(n^d)$.
- *Nếu* $f_1[n] = O(g[n])$, $f_2[n] = O(g[n])$ *thì* $f_1 + f_2 = O(g)$.
- *Nếu* $f_1 = O(g_1)$, $f_2 = O(g_2)$ *thì* $f_1 f_2 = O(g_1 g_2)$.
- *Nếu tồn tại giới hạn hữu hạn*:

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

  thì f=O(g)

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

Định nghĩa:

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

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

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

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

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

$$\exp = (\sqrt{\log n \log \log n})$$

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

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

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

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

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

Bảng 1. 1. Độ phức tạp để phân tích số nguyên ra thừa số nguyên tố

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

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

1.6. CÂU HỎI VÀ BÀI TẬP

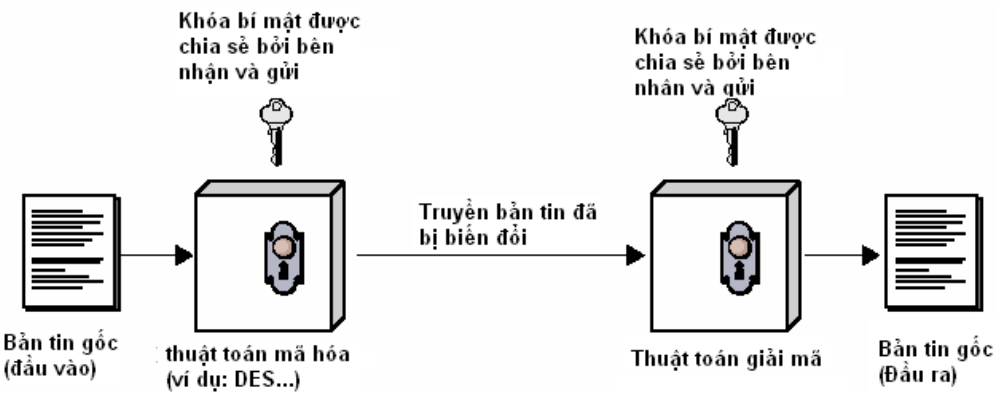
1) Độ an toàn thuật toán là gì? Hãy trình bày một số tiêu chí cơ bản để đánh giá độ an toàn của một thuật toán?
2) Hãy trình bày về các phương pháp phân loại mã hóa?
3) Hãy trình bày sự khác nhau cơ bản giữa mã hóa khóa bí mật và mã hóa khóa công khai?
4) Hãy phân tích vai trò của mật mã học?
5) Hãy định nghĩa về độ an toàn tính toán của hệ mật trong lý thuyết xác suất?
6) Hãy nêu định nghĩa về độ phức tạp tính toán?
7) Hãy liệt kê một số ứng dụng của mã hóa trong thực tế?

CHƯƠNG 2: MÃ HÓA KHÓA ĐỐI XỨNG

Chương 2 cung cấp các kiến thức về hệ mật khóa đối xứng cũng như nguyên tắc mã hóa, giải mã của một số thuật toán mã hóa khóa đối xứng. Ngoài ra, chương 2 trình bày một số ưu và nhược điểm của các kỹ thuật mã hóa khóa đối xứng.

2.1. GIỚI THIỆU VỀ MÃ HÓA KHÓA ĐỐI XỨNG

Hình 2.1 mô tả quy trình mã hóa và giải mã của mã hóa khóa bất đối xứng.



Hình 2. 1. Mô hình mã hóa đối xứng

Từ mô hình mã hóa khóa đối xứng được trình bày trên hình 2.1 có thể thấy rằng trong mô hình mã hóa khóa đối xứng cũng bao gồm 5 thành phần chính là *P*, *C*, *K*, *E*, *D*. Trong đó:

- *P* là một tập hữu hạn các bản rõ có thể.
- *C* là một tập hữu hạn các bản mã có thể.
- *K* là một tập hữu hạn các khoá có thể (không gian khoá). Trong mô hình mã hóa khóa đối xứng thì chỉ sử dụng một khóa duy nhất cho cả quá trình mã hóa và giải mã.
- *E:* thuật toán, kỹ thuật mã hóa.
- *D:* thuật toán, kỹ thuật giải mã

Trong lịch sử phát triển mã hóa thì các kỹ thuật mã hóa đối xứng được ra đời và phát triển từ rất lâu đời. Trong suốt thời kỳ phát triển của kỹ thuật mã hóa này thì đã có rất nhiều thuật toán và phương pháp mã hóa khác nhau được ra đời. Trong bài giảng này, tác giả sẽ đi vào phân tích và làm rõ một số thuật toán mã hóa đối xứng theo lịch sử phát triển của chúng. Cụ thể, 2 nhóm thuật toán, kỹ thuật mã hóa được phân loại trong bài giảng này bao

gồm: Kỹ thuật mã hóa đối xứng cổ điển và kỹ thuật đối xứng mã hóa hiện đại. Trong đó, đối với phương pháp mã hóa đối xứng cổ điển tác giả tập trung vào trình bày và phân tích một số thuật toán như: kỹ thuật thay thế, kỹ thuật hoán vị... Đối với mã hóa đối xứng hiện đại, bài giảng sẽ chia thành 2 nhóm chính dựa trên kỹ thuật xử lý dữ liệu là kỹ thuật mã hóa khối và kỹ thuật mã hóa dòng. Trong các nội dung tiếp theo, bài giảng sẽ trình bày và làm rõ các phương pháp mã hóa đối xứng này.

2.2. CÁC KỸ THUẬT MÃ HÓA ĐỐI XỨNG CỔ ĐIỂN

2.2.1. Phương pháp thay thế

Mã hóa thay thế là phương pháp thay thế mỗi ký tự thông điệp thành ký tự bản mã. Từ đó, người nhận có thể thay thế ngược lại những ký tự của bản mã để được thông điệp.

Trong mật mã học cổ điển, có một số phương pháp mã hóa thay thế:

- Mã hóa thay thế đơn bảng.
- Mã hóa thay thế đa ký tự.
- Mã hóa thay thế đa bảng.

2.2.1.1. Phương pháp mã hóa thay thế đơn bảng

Trước khi tìm hiểu phương pháp thay thế đơn bảng, hãy đi tìm hiểu mã hóa *Caesar* (*Caesar Cipher*).

a. Phương pháp Caesar

Thế kỷ thứ 3 trước công nguyên, nhà quân sự người La Mã *Julius* Caesar đã nghĩ ra phương pháp mã hóa một thông điệp như sau: thay thế mỗi chữ trong thông điệp bằng chữ đứng sau nó *k* vị trí trong bảng chữ cái. Giả sử chọn *k = 3*, có bảng chuyển đổi như sau:

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

Chú ý: sau Z sẽ vòng lại là A, do đó x → A, y → B và z → C

Giả sử có *thông điệp* gốc: meet me after the toga party

Như vậy *bản mã* sẽ là: PHHW PH DIWHU WKH WRJD SDUWB

Thay vì gửi trực tiếp thông điệp cho các cấp dưới, Ceasar gửi bản mã. Khi cấp dưới nhận được bản mã, tiến hành giải mã theo quy trình ngược lại để có được thông điệp. Như vậy nếu đối thủ của *Ceasar* có lấy được bản mã, thì cũng không hiểu được ý nghĩa của bản mã.

Gán cho mỗi chữ cái theo thứ tự từ A đến Z một con số nguyên theo thứ tự từ 0 đến 25. Phương pháp *Ceasar* được biểu diễn như sau: với mỗi chữ cái *p* thay bằng chữ mã hóa *C*, trong đó:

$C = (p + k) \bmod 26$ (trong đó *mod* là phép chia lấy số dư)

Và quá trình giải mã đơn giản là: $p = (C - k) \bmod 26$.

Trong đó *k* được gọi là khóa. Dĩ nhiên là *Ceasar* và cấp dưới phải cùng dùng chung một giá trị khóa *k*, nếu không thông điệp giải mã sẽ không giống thông điệp ban đầu.

Ngày nay phương pháp mã hóa của Ceasar không được xem là an toàn. Giả sử đối thủ của **Ceasar** có được bản mã **PHHW PH DIWHU WKH WRJD SDUWB** và biết được phương pháp mã hóa và giải mã là phép cộng trừ *modulo* 26. "*Kẻ tấn công*" có thể thử tất cả 25 trường hợp của *k* như bảng 2.1:

Bảng 2. 1. Bảng thống kê kết quả tấn công vét cạn

KEY	PHHW	PH	DIWHU	WKH	WRJD	SDUWB
1	oggv	og	chvgt	vjg	vqic	rctva
2	nffu	nf	bgufs	uif	uphb	qbsuz
3	**meet**	**me**	**after**	**the**	**toga**	**party**
4	ldds	ld	zesdq	sgd	snfz	ozqsx
5	kccr	kc	ydrcp	rfc	rmey	nyprw
6	jbbq	jb	xcqbo	qeb	qldx	mxoqv
7	iaap	ia	wbpan	pda	pkcw	lwnpu
8	hzzo	hz	vaozm	ocz	ojbv	kvmot
9	gyyn	gy	uznyl	nby	niau	julns

10	fxxm	fx	tymxk	max	mhzt	itkmr
11	ewwl	ew	sxlwj	lzw	lgys	hsjlq
12	dvvk	dv	rwkvi	kyv	kfxr	grikp
13	cuuj	cu	qvjuh	jxu	jewq	fqhjo
14	btti	bt	puitg	iwt	idvp	epgin
15	assh	as	othsf	hvs	hcuo	dofhm
16	zrrg	zr	nsgre	gur	gbtn	cnegl
17	yqqf	yq	mrfqd	ftq	fasm	bmdfk
18	xppe	xp	lqepc	esp	ezrl	alcej
19	wood	wo	kpdob	dro	dyqk	zkbdi
20	vnnc	vn	jocna	cqn	cxpj	yjach
21	ummb	um	inbmz	bpm	bwoi	xizbg
22	tlla	tl	hmaly	aol	avnh	whyaf
23	skkz	sk	glzkx	znk	zumg	vgxze
24	rjjy	rj	fkyjw	ymj	ytlf	ufwyd
25	qiix	qi	ejxiv	xli	xske	tevxc

Trong 25 trường hợp trên, chỉ có trường hợp $k = 3$ thì bản giải mã tương ứng là có ý nghĩa. Do đó "*Kẻ tấn công*" có thể chắc chắn rằng "***meet me after the toga party***" là thông điệp ban đầu.

b. Phương pháp thay thế đơn bảng (Monoalphabetic Substitution Cipher)

Xét lại phương pháp *Caersar* với $k = 3$:

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

Phương pháp đơn bảng tổng quát hóa phương pháp Ceasar bằng cách dòng mã hóa không phải là một dịch chuyển k vị trí của các chữ cái A, B, C, …nữa mà là một hoán vị của 26 chữ cái này. Lúc này mỗi hoán vị được xem như là một khóa. Giả sử có hoán vị sau:

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

Như vậy thông điệp **meet me after the toga party**

được mã hóa thành: **NJJU NJ ZRUJM UKJ UVSZ DZMUE**

Quá trình giải mã được tiến hành ngược lại để cho ra thông điệp ban đầu.

Việc mã hóa được tiến hành bằng cách thay thế một chữ cái trong thông điệp thành một chữ cái trong bản mã, nên phương pháp này được gọi là phương pháp thay thế. Số lượng hoán vị của *26* chữ cái là *26!*, đây cũng chính là số lượng khóa của phương pháp này. Vì 26! là một con số khá lớn nên việc tấn công phá mã vét cạn khóa là bất khả thi (*6400* thiên niên kỷ với tốc độ thử khóa là *109 khóa/giây*). Vì vậy mã hóa đơn bảng đã được xem là một phương pháp mã hóa an toàn trong suốt *1000* năm sau công nguyên.

2.2.1.2. Phương pháp thay thế đa ký tự

a. Mã hóa Playfair

Bảng 2. 2: Ma trận mã hóa Playfair

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Mã hóa *Playfair* sử dụng đơn vị mã hóa là *2* ký tự đứng sát nhau. Như vậy hai ký tự này được thay thế cùng một lúc bởi hai ký tự khác. Mã hóa Playfair dùng một ma trận *5x5* (*Matrix State*).

Trong bảng trên, khóa *MONARCHY* được điền vào dòng đầu tiên và 3 ký tự dòng thứ 2 của bảng. Các ô còn lại được điền các chữ cái theo thứ tự của bảng chữ cái Tiếng Anh. Riêng 2 chữ *I* và *J* được điền vào chung 1 ô vì trong Tiếng Anh hai chữ cái này ít bị nhầm lẫn. Ví dụ: khi gặp chữ *SM_LE* thì sẽ đoán được chữ đó là *SMILE* chứ không thể nào là *SMJLE* được.

Trước khi tiến hành mã hóa, thông điệp được tách ra thành các cặp ký tự. Nếu 2 ký tự trong một cặp giống nhau thì sẽ được thay chữ cái đứng sau trong cặp đó bằng chữ *X* (vì trong Tiếng Anh thì thường 2 chữ *X* rất hiếm khi đứng cạnh nhau). Ví dụ: từ *football* sẽ được tách thành 4 cặp *fo ot ba lx*.

Quy tắc mã hóa như sau :

- Nếu 2 ký tự được mã hóa thuộc cùng 1 hàng, thì được thay thế bằng 2 ký tự tiếp theo trong hàng. Nếu đến cuối hàng thì quay về đầu hàng. Ví dụ: cặp ***LP*** thay bằng ***PQ***, cặp ***ST*** thay bằng ***TU***.
- Nếu 2 ký tự trong cặp thuộc cùng 1 cột, thì được thay bằng 2 ký tự tiếp theo trong cột. Nếu đến cuối cột thì quay về đầu cột. Ví dụ cặp *OV* được mã hóa thành *HO*.
- Trong các trường hợp còn lại, 2 ký tự được mã hóa sẽ tạo thành đường chéo của 1 hình chữ nhật và được thay bằng 2 ký tự trên đường chéo kia. Ví dụ: *HS* trở thành *BP* (*B* cùng dòng với *H* và *P* cùng dòng với *S*); *EA* trở thành *IM* (hoặc *JM*).

Như vậy nếu chỉ xét trên 26 chữ cái thì mã hóa *Playfair* có 26x26=676 cặp chữ cái, do đó các cặp chữ cái này ít bị chênh lệch về tần suất hơn so với sự chênh lệnh tần suất của từng chữ cái. Ngoài ra số lượng các cặp chữ cái nhiều hơn cũng làm cho việc phá mã tần suất khó khăn hơn. Đây chính là lý do mà người ta tin rằng mã hóa *Playfair* không thể bị phá và được quân đội Anh sử dụng trong chiến tranh thế giới lần thứ nhất.

b. Mã hóa Hill

Trong mã hóa Hill, mỗi chữ cái (A – Z) được gán tương ứng với 1 số nguyên từ 0 đến 25. Đơn vị mã hóa của mã hóa *Hill* là 1 khối (*block*) *m* ký tự thông điệp (ký hiệu $p_1p_2...p_m$) thành 1 khối (*block*) *m* ký tự bản mã (ký hiệu $c_1c_2...c_m$). Việc thay thế này được

thực hiện bằng *m* phương trình tuyến tính. Giả sử *m = 3*, minh họa *m* phương trình đó như sau:

Dạng phương trình :

$$c_1 = k_{11}p_1 + k_{12}p_2 + k_{13}p_3 \ mod\ 26$$

$$c_2 = k_{21}p_1 + k_{22}p_2 + k_{23}p_3 \ mod\ 26$$

$$c_3 = k_{31}p_1 + k_{32}p_2 + k_{33}p_3 \ mod\ 26$$

Dạng ma trận :

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \bmod 26$$

Hay: *C = KP mod* 26 với *P và C* là vector đại diện cho thông điệp và bản mã, còn *K* là ma trận dùng làm khóa.

Xét ví dụ thông điệp là ***paymoremoney*** cùng với khóa K là :

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

Ba chữ cái đầu tiên của thông điệp tương ứng với vector (15, 0, 24). Vậy

$$\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 5 \\ 0 \\ 24 \end{bmatrix} \bmod 26 = \begin{bmatrix} 11 \\ 23 \\ 18 \end{bmatrix} = LNS$$

Thực hiện tương tự có bản mã đầy đủ là *LNSHDLEWMTRW*.

Để giải mã cần sử dụng một ma trận nghịch đảo của *K* là K^{-1}, tức là $K^{-1}K \ mod$ 26 = 1 là ma trận đơn vị (không phải mọi ma trận *K* đều tồn tại ma trận nghịch đảo, tuy nhiên nếu tốn tại thì có thể tìm được ma trận đơn vị bằng cách tính hạng *det* của ma trận).

Ví dụ ma trận nghịch đảo của ma trận trên là :

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

Vì:

$$\begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} = \begin{bmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{bmatrix} \bmod 26 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Khi đó bảng giải mã là $K^{-1}C \bmod 26 = K^{-1}KP \bmod 26 = P$.

Có thể thấy mã hóa *Hill* ẩn giấu các thông tin về tần suất nhiều hơn mã hóa Playfair do có thể mã hóa 3 hoặc nhiều hơn nữa các ký tự cùng lúc.

2.2.1.3. Phương pháp mã hóa thay thế đa bảng

Với sự phát hiện ra quy luật phân bố tần suất, các nhà phá mã đang tạm thời chiếm ưu thế trong cuộc chiến mã hóa-thám mã. Cho đến thế kỷ thứ 15, một nhà ngoại giao người Pháp tên là Vigenere đã tìm ra phương án mã hóa thay thế đa bảng.

Trong Vigenere Cipher, người ta dùng tất cả 26 bảng thế thu được từ bảng gốc chữ cái tiếng Anh mà dịch đi từ 0-25 vị trí. Sự hoà trộn này có quy luật hoàn toàn xác định bởi khoá. Mỗi chữ của khoá sẽ xác định mỗi bảng thế được dùng. Bảng 2.3 (bảng thế A – dịch đi 0 vị trí từ bảng gốc chữ cái tiếng Anh) là một trong 26 bảng được sử dụng trong phương pháp Vigenere.

Bảng 2. 3. Bảng thế A

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

h	H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
i	I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
j	J K L M N O P Q R S T U V W X Y Z A B C D E F G H I
k	K L M N O P Q R S T U V W X Y Z A B C D E F G H I J
l	L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
m	M N O P Q R S T U V W X Y Z A B C D E F G H I J K L
n	N O P Q R S T U V W X Y Z A B C D E F G H I J K L M
o	O P Q R S T U V W X Y Z A B C D E F G H I J K L M N
p	P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
q	Q R S T U V W X Y Z A B C D E F G H I J K L M N O P
r	R S T U V W X Y Z A B C D E F G H I J K L M N O P Q
s	S T U V W X Y Z A B C D E F G H I J K L M N O P Q R
t	T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
u	U V W X Y Z A B C D E F G H I J K L M N O P Q R S T
v	V W X Y Z A B C D E F G H I J K L M N O P Q R S T U
w	W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
x	X Y Z A B C D E F G H I J K L M N O P Q R S T U V W
y	Y Z A B C D E F G H I J K L M N O P Q R S T U V W X
z	Z A B C D E F G H I J K L M N O P Q R S T U V W X Y

Dòng thứ *t* chính là một mã hóa *Ceasar t-1* vị trí. Ví dụ: Ở dòng thứ 9 ứng với từ khóa *I* là một mã hóa *Ceasar 8* vị trí. Nếu tổng quát hóa, mỗi dòng của mã hóa đa bảng chính là một mã hóa đơn bảng.

Ví dụ:

Khóa	r	a	d	i	o	r	a	d	i	o	r	a
Bản rõ	c	o	d	e	b	r	e	a	k	i	n	g
Bản mã	t	o	g	m	p	i	e	d	s	w	e	g

Theo ví dụ trên, khóa có độ dài là 5 (radio), nên các ký tự trong bản rõ sẽ được mã hóa theo modulo của 5. Cụ thể, các ký tự có vị trí mod 5 bằng 0 (c, r, n) sẽ mã hóa bởi bảng thế R (A thành R), bằng 1 (o, e, g) sẽ mã hóa bởi bảng thế A, bằng 2 (d, a) sẽ mã hóa bởi

bảng thế D, bằng 3 (e, k) sẽ mã hóa bởi bảng thế I và bằng 4 (b, i) sẽ mã hóa bởi bảng thế O.

2.2.1.4. One Time Pad

Để khắc phục điểm yếu của việc lặp lại khóa trong mã hóa thay thế đa bảng, bỏ sự liên quan giữa thông điệp và bản mã, *Joseph Mauborgne*, giám đốc viện nghiên cứu mật mã của quân đội Mỹ, vào cuối cuộc chiến tranh thế giới lần thứ nhất đã đề xuất phương án dùng khóa ngẫu nhiên. Khóa ngẫu nhiên có chiều dài bằng chiều dài của thông điệp, mỗi khóa chỉ sử dụng một lần.

Ví dụ mã hóa:

Thông điệp *P*: **wearediscoveredsaveyourself**

Khóa *K1*: **FHWYKLVMKVKXCVKDJSFSAPXZCVP**

Bản mã *C*: **BLWPOODEMJFBTZNVJNJQOJORGGU**

Nếu dùng khóa K1 để giải mã thì sẽ được lại thông điệp:

“wearediscoveredsaveyourself”

Tuy nhiên xét hai trường hợp giải mã bản mã trên với 2 khóa khác như sau:

Trường hợp 1: Bản mã C: **BLWPOODEMJFBTZNVJNJQOJORGGU**

Khóa K2: **IESRLKBWJFCIFZUCJLZXAXAAPSY**

Bản giải mã: **theydecidedtoattacktomorrow**

(they decided to attack tomorrow)

Trường hợp 2: Bản mã C: **BLWPOODEMJFBTZNVJNJQOJORGGU**

Khóa K3: **FHAHDDRAIQFIASJGJWQSVVBJAZB**

Bản giải mã: **wewillmeetatthepartytonight**

(we will meet at the party tonight)

Trong cả hai trường hợp trên thì bản giải mã đều có ý nghĩa. Điều này có nghĩa là nếu người phá mã thực hiện phá mã vét cạn thì sẽ tìm được nhiều khóa ứng với nhiều bản 24 tin có ý nghĩa, do đó sẽ không biết được thông điệp nào là thông điệp chính. Điều này chứng minh phương pháp OTP là phương pháp mã hóa an toàn tuyệt đối, và được xem là

"chén thánh" của khóa mật mã cổ điển. Một điều cần chú ý là để phương pháp OTP là an toàn tuyệt đối thì mỗi khóa chỉ được sử dụng một lần. Nếu một khóa được sử dụng nhiều lần thì cũng không khác gì việc lặp lại một từ trong khóa (ví dụ khóa có từ *DECEPTIVE* được lặp lại). Ngoài ra các khóa phải thật sự ngẫu nhiên với nhau. Nếu các điều này bị vi phạm thì sẽ có một mối liên hệ giữa thông điệp và bản mã, và người phá mã sẽ tận dụng mối quan hệ này.

Tuy nhiên, phương pháp *OTP* không có ý nghĩa sử dụng thực tế. Vì chiều dài khóa bằng chiều dài thông điệp, mỗi khóa chỉ sử dụng một lần, nên thay vì truyền khóa trên kênh an toàn thì có thể truyền trực tiếp thông điệp mà không cần quan tâm đến vấn đề mã hóa. Vì vậy sau chiến tranh thế giới thứ nhất, người ta vẫn chưa thể tìm ra loại mật mã nào khác mà không bị phá mã. Mọi cố gắng vẫn là tìm cách thực hiện một mã thay thế đa bảng dùng một khóa dài, ít lặp lại, để hạn chế phá mã. Máy *ENIGMA* được quân đội Đức sử dụng trong chiến tranh thế giới lần thứ 2 là một máy như vậy. Sử dụng máy *ENIGMA*, Đức đã chiếm ưu thế trong giai đoạn đầu của cuộc chiến. Tuy nhiên trong giai đoạn sau, các nhà phá mã người Ba Lan và Anh (trong đó có *Alan Turing*, người phá minh ra máy tính có thể lập trình được) đã tìm ra cách phá mã máy *ENIGMA*. Việc phá mã thực hiện được dựa vào một số điểm yếu trong khâu phân phối khóa của quân Đức. Điều này đóng vai trò quan trọng vào chiến thắng của quân đồng minh trong cuộc chiến.

2.2.2. Phương pháp mã hóa hoán vị

Trong mã hóa hoán vị (đổi chỗ), các ký tự trong thông điệp được sắp xếp, đổi chỗ lại để tạo thành bản mã. Do thứ tự các ký tự đã bị thay đổi nên bản mã sẽ trở thành thông điệp vô nghĩa, không thể đọc được thông tin chứa trong nó.

Một ví dụ đơn giản, dịch vị trí các ký tự thông điệp sang phải 2 đơn vị. Khi đó thông điệp "hello world" sẽ thành "*ldhello wor*" hoàn toàn vô nghĩa.

Có thể áp dụng cho từng khối *n* ký tự và thực hiện hoán vị theo một khóa K định sẵn.

Ví dụ thực hiện đổi chỗ khối 8 ký tự :

Cho khóa *K* như sau : 1→4, 2→8, 3→1, 4→5, 5→7, 6→2, 7→6, 8→3

Thông điệp(*Plaintext*): SACKGAUL | SPARENOO | NE | |

Bản mã(*Ciphertext*) : UKAGLSCA | ORPEOSAN | E N | |

Số thứ tự : 8 6 7 5 4 3 2 1|8 6 7 5 4 3 2 1|8 6 7 5 4 3 2 1|8 6 7 5 4 3 2 1|

2.2.3. Phương pháp mã hóa XOR

Phương pháp mã hóa *XOR* sử dụng phép toán logic *XOR* để tạo ra bản mã. Từng bit của thông điệp được *XOR* tương ứng với từng bit của khóa để cho ra bản mã.

Bảng 2. 4. Bảng chân trị phép toán XOR

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

Ví dụ: Mã hóa từ LOVE với khóa *K* là từ HATE sử dụng phương pháp *XOR*.

Chuyển các từ và khóa về dạng nhị phân như sau:

Plainttext L O V E

Plaintext_bin 0100 1100 | 0100 1111 | 0101 0110 | 0100 0101

Key H A T E

Key_bin 0100 1000 | 0100 0001 | 0101 0100 | 0100 0101

Ciphertext_bin 0000 0100 | 0000 | 1110 | 0000 0010 0000 0000

2.2.4. Phương pháp sách hoặc khóa chạy

Trong các tiểu thuyết trinh thám, phim trinh thám thường bắt gặp phương pháp này trong việc mã hóa và giải mã sử dụng khóa chứa trong các trang sách hoặc chứa trong các dòng của một thông điệp .v.v.v..

Ví dụ: với bản mã là 259, 19, 8; 22, 3, 8; 375, 7, 4; 394, 17, 2 và cuốn sách được dùng là **"A Fire Up on the Deep":**

- Trang 259, dòng 19, từ thứ 8 → *sack*
- Trang 22, dòng 3, từ thứ 8 → *island*
- Trang 375, dòng 7, từ thứ 4 → *sharp*
- Trang 394, dòng 17, từ thứ 2 → *path*

- Thông điệp tương ứng của bản mã "259,19,8;22,3,8;375,7,4;394,17,2 " là "sack island sharp path".

2.3. CÁC KỸ THUẬT MÃ HÓA KHỐI ĐỐI XỨNG

2.3.1. Giới thiệu về mã hóa khối

Như đã trình bày ở trên, các thuật toán mã hóa khối là các thuật toán sử dụng các kỹ thuật và phương pháp nhằm xử lý các dữ liệu dưới dạng khối. Có 3 vấn đề cần quan tâm và xem xét trong các kỹ thuật mã hóa khối như sau:

Vấn đề 1: Các yêu cầu an toàn đối với mã khối: Mỗi kỹ thuật và phương pháp có những tiêu chí riêng để xây dựng và xử lý các khối. Tuy nhiên, để xây dựng thuật toán mã hóa khối an toàn các thuật toán mã hóa khối thường sử dụng kết hợp các thao tác mã hóa tạo ra tính hỗn loạn và tính khuếch tán thông tin. Trong đó:

- Tính hỗn loạn (*confusion*) giúp phá vỡ mối quan hệ giữa thông điệp và bản mã, tạo ra mối quan hệ phức tạp và chặt chẽ giữa khóa và bản mã.
- Sự khuếch tán (*diffusion*) giúp phá vỡ và phân tán các phần tử trong các mã xuất hiện trong thông điệp để không thể phát hiện ra các mẫu này trong bản mã.

Vấn đề 2: Các chế độ hoạt động của mã hóa khối: Trong kỹ thuật mã hóa khối, các thuật toán mã hóa thường lựa chọn các chế độ hoạt động của các khối khác nhau nhằm tính toán và xử lý dữ liệu của từng khối. Trong mã hóa khối có 4 chế độ hoạt động cơ bản như sau:

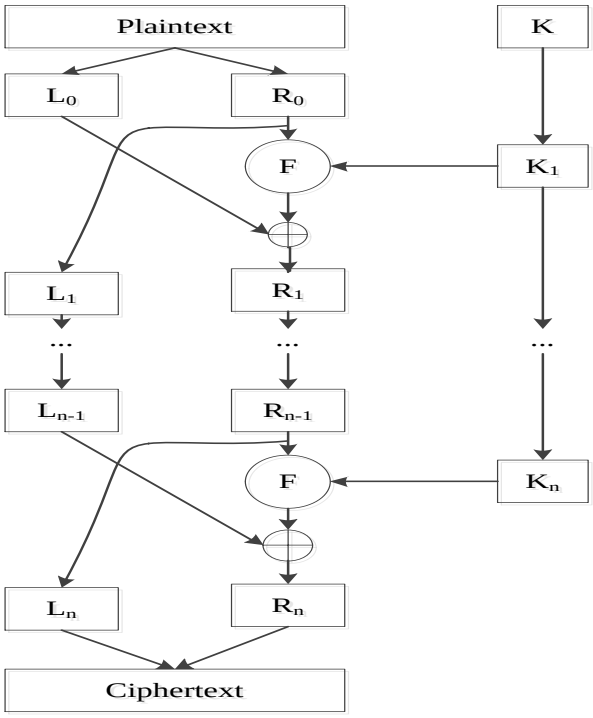
- Chế độ ECB (Electronic Codebook): cùng khối thông điệp đầu vào, khối bản mã giống nhau. Các khối bản mã hoàn toàn độc lập nhau.
- Chế độ CBC (Cipher-Block Chaining): cùng khối thông điệp đầu vào, khối bản mã giống nhau với cùng khóa và phần nối đuôi. Khối mã c_j phụ thuộc vào khối rõ x_j và các khối rõ trước đó (x_1-x_{j-1}).
- Chế độ CFB (Cipher Feedback): cùng khối thông điệp đầu vào, khối bản mã khác nhau. Khối mã c_j phụ thuộc vào khối rõ x_j và các khối rõ trước đó (x_1-x_{j-1}).
- Chế độ OFB (Output Feedback): cùng khối thông điệp đầu vào, khối bản mã khác nhau. Luồng khóa độc lập với thông điệp.

Vấn đề 3: Các mô hình mã hóa khối. Về mô hình mã khối, các nhà mật mã chia làm 2 mô hình chính: mạng thay thế hoán vị (*Substitution-Permutation Network - SPN*) và Mô hình Feistel.

2.3.2. Thuật toán mã hóa DES/3DES

2.3.2.1. Giới thiệu về mô hình Feistel

Hình 2.2 mô tả cấu trúc đề xuất bởi *Feistel*. Đầu vào của thuật toán mã hóa là khối văn bản gốc có chiều dài *2w* bit và một khóa *K*. Khối thông điệp gốc được chia thành hai nửa, L_0 và R_0. Hai nửa này đi qua *n* vòng xử lý rồi tổ hợp để tạo ra khối bản mã hóa. Mỗi vòng *i* có đầu vào L_{i-1} và R_{i-1}, lấy từ vòng trước đó, giống như 1 khoá con K_i, lấy từ tổng thể *K*. Nói chung, những khóa con K_i khác với khóa *K* và các khóa khác.



Hình 2. 2. Mô hình Feistel

Tất cả các vòng có cấu trúc tương tự nhau. Một thay thế được thực hiện trên một nửa trái của dữ liệu. Điều này được thực hiện bằng cách áp dụng một hàm vòng *F* cho nửa bên phải của dữ liệu và sau đó lấy *exclusive-OR* của các đầu ra của hàm đó và một nửa còn lại của dữ liệu. Tất cả các vòng có cùng một cấu trúc chung cho mỗi vòng, nhưng là tham số của mỗi vòng là khóa con K_i. Sau sự thay thế này, một hoán vị được thực hiện mà bao

gồm việc trao đổi hai nửa của dữ liệu. Việc thực hiện chính xác của một mạng *Feistel* phụ thuộc vào sự lựa chọn của các tham số và đặc điểm thiết kế sau:

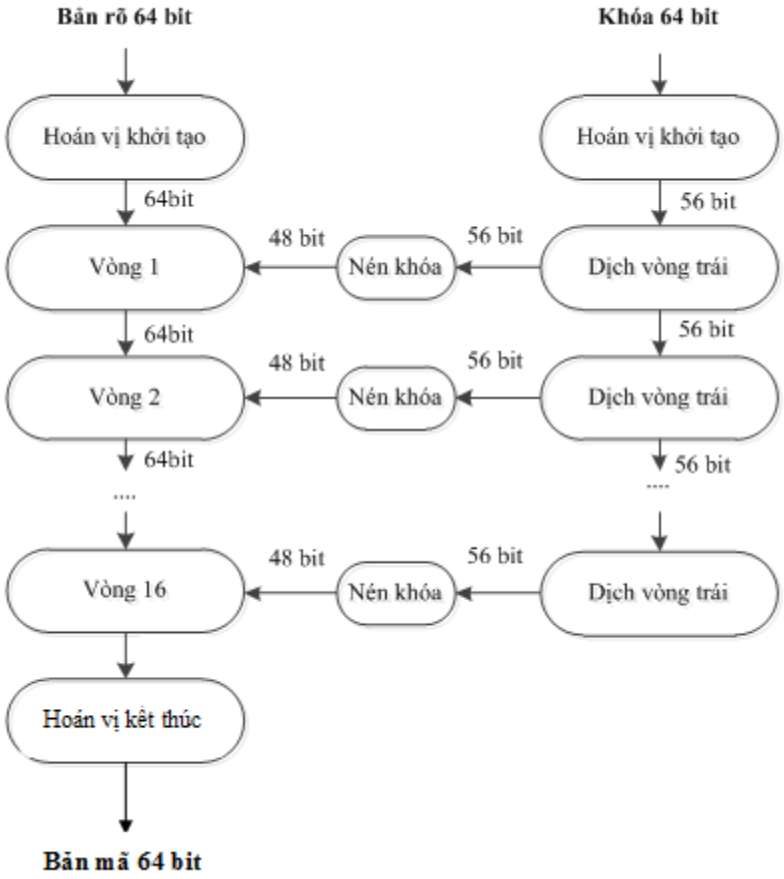
- Kích cỡ khối**:** khối kích thước lớn hơn có nghĩa là bảo mật cao hơn (tất cả những thứ khác bằng nhau), nhưng giảm đi tốc độ mã hóa/giải mã của một thuật toán cho trước. Theo truyền thống, một kích thước khối 64 bit đã được coi là một sự cân bằng hợp lý và đã được phổ cập trong thiết kế gần mã hóa khối.
- Kích thước khóa: kích thước lớn hơn có nghĩa là an toàn hơn, nhưng có thể làm giảm tốc độ mã hóa/giải mã. Việc bảo mật cao hơn đạt được bằng cách chống tấn công *brute-force*(tấn công vét cạn) tốt hơn. Kích thước khóa của 64 bit hoặc ít hơn bây giờ nhiều người xem là không đủ an toàn, và 128 bit đã trở thành một kích thước khóa thông thường.
- Số vòng: Bản chất của mã hóa *Feistel* là một vòng duy nhất cung cấp bảo mật không đầy đủ nhưng nhiều vòng cung cấp sẽ tăng cường bảo mật hơn. Một kích thước điển hình là 16 vòng.
- Thuật toán sinh khóa phụ: phức tạp lớn hơn trong thuật toán này nên dẫn đến khó khăn lớn hơn của phân tích mật mã.
- Hàm vòng: Một lần nữa, phức tạp hơn thường có nghĩa chống phân tích mật mã tốt hơn. Có hai quan tâm khác trong thiết kế của một mã hóa *Feistel*:
- Tốc độ phần mềm mã hóa/giải mã: Trong nhiều trường hợp, mã hóa được nhúng trong các ứng dụng hoặc các chức năng tiện ích. Vì vậy, tốc độ thực hiện của thuật toán sẽ trở thành một mối quan tâm.
- Dễ dàng trong việc phân tích**:** Mặc dù muốn làm cho thuật toán khó khăn nhất có thể để chống lại việc phân tích mã. Vì vậy, nếu thuật toán có thể được giải thích ngắn gọn và rõ ràng, nó được dễ dàng hơn trong việc phân tích thuật toán để giảm rủi ro và do đó chống lại các phân tích mã phát triển ở một mức độ cao hơn.

2.3.2.2. Thuật toán mã hóa DES

DES được phát triển tại *IBM* vào đầu những năm 1970 và được thừa nhận là chuẩn mã hóa tại Mỹ (*NSA*) vào năm 1976. Sau đó *DES* được sử dụng rộng rãi trong những năm 70 và 80. Mã hóa *DES* có các tính chất sau:

- Là mã thuộc hệ mã *Feistel* gồm 16 vòng, ngoài ra *DES* có thêm một hoán vị khởi tạo trước khi vào vòng 1 và một hoán vị kết thúc sau vòng 16.
- Kích thước của khối là 64 bit: ví dụ thông điệp"*meetmeafterthetogaparty*" biểu diễn theo mã *ASCII* thì mã *DES* sẽ mã hóa làm 3 lần, mỗi lần 8 chữ cái (64 bit): *meetmeaf - tertheto - gaparty*.
- Kích thước khóa là 64 bit (thực chất chỉ có 56 bit được sử dụng để mã hóa);
- Mỗi vòng của *DES* dùng khóa con có kích thước 48 bit được trích ra từ khóa chính.

Hình 2.3 trình bày tổng quát về quy trình mã hóa của thuật toán mã hóa DES.



Hình 2. 3. Sơ đồ tổng quát quá trình mã hóa DES

Sơ đồ mã hóa *DES* gồm 3 phần, phần thứ nhất là các hoán vị khởi tạo và hoán vị kết thúc. Phần thứ hai là các vòng Feistel, phần thứ ba là thuật toán sinh khóa con. Tiếp theo bài giảng sẽ đi vào phân tích từng thành phần này.

a) Hoán vị khởi tạo và hoán vị kết thúc

Giá trị đầu vào là khối M có kích thước 64 bit $m_1,m_2,m_3,\ldots,m_{64}$. Hoán vị khởi tạo sẽ thực hiện biến đổi giá trị các bit theo nguyên tắc sau: $(m_1,m_2,m_3,\ldots,m_{64} \rightarrow m_{58},m_{50},m_{42},\ldots,m_7)$. Bảng 2.5 thể hiện bảng hoán bị khởi tạo của DES

Bảng 2. 5. Bảng hoán vị khởi tạo *DES*

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

Hoán vị kết thúc hoán đổi các bit theo quy tắc sau: $(c_1,c_2,c_3,\ldots,c_{64} \rightarrow c_{40},c_8,c_{48},\ldots,c_{25})$

Bảng 2. 6. Bảng hoán vị kết thúc *DES*

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

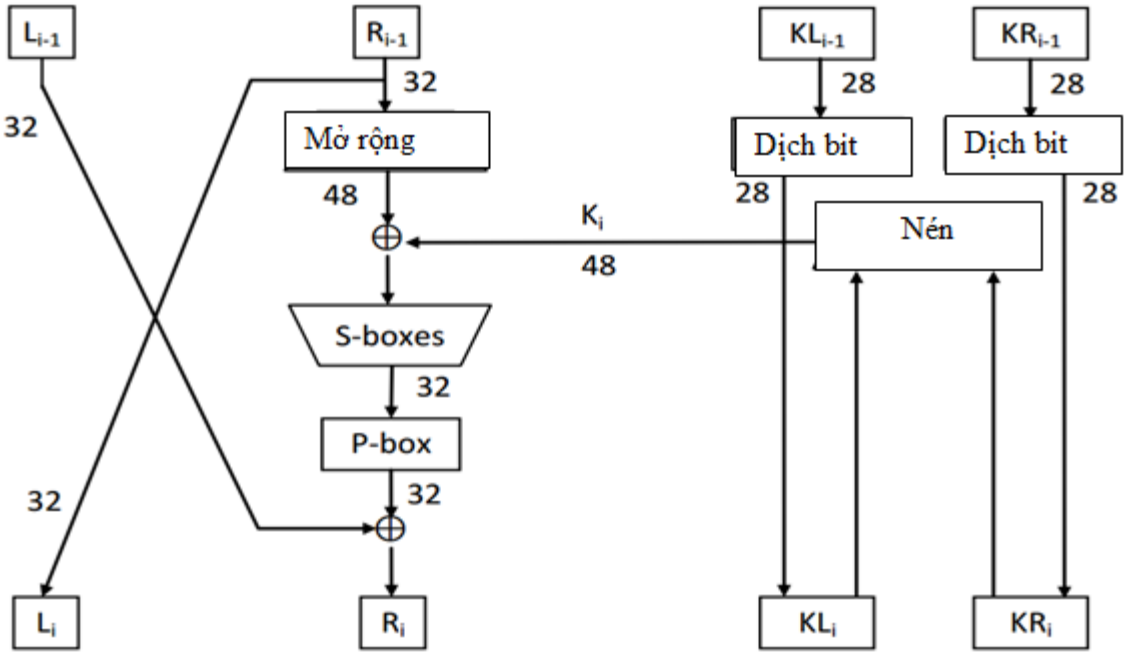
Ví dụ: Cho giá trị đầu vào thông điệp dưới dạng thập lục phân (*hexa*) như sau: $0x0002\ 0000\ 0000\ 0001_{16}$.

→ Phân tích thấy chỉ có bit 15 và bit 64 mang giá trị sử dụng bảng *IP* thấy giá trị tại bit 15 chuyển thành bit 63, bit 64 trở thành bit 25 trong output.

→ Sau hoán vị khởi tạo, giá trị sẽ là 0x0000 0080 0000 0002.

Hoán vị kết thúc chính là hoán vị nghịch đảo của hoán vị khởi tạo. Đối với *knownplaintext* hay *chosen-plaintext attack*, hoán vị khởi tạo và hoán vị kết thúc không có ý nghĩa bảo mật, sự tồn tại của hai hoán vị trên được nhận định là do yếu tố lịch sử.

b) Các vòng của *DES*



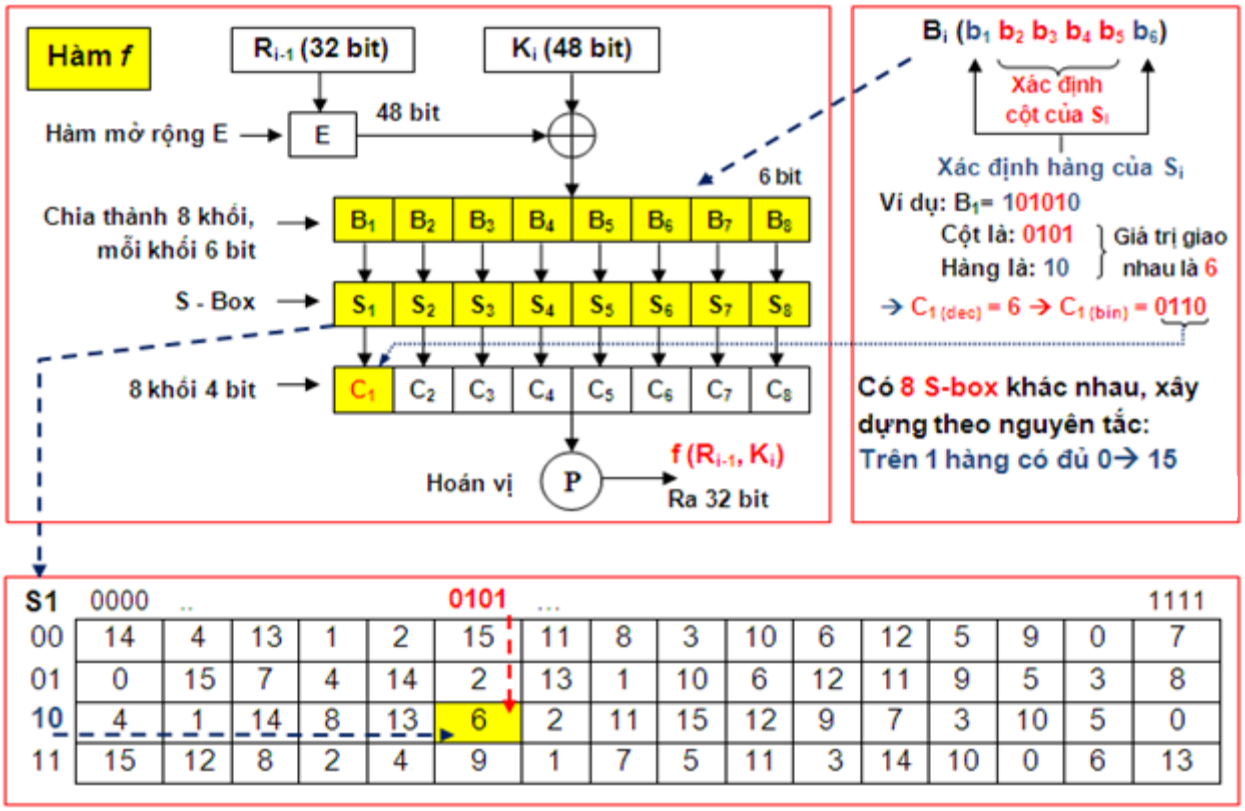
Hình 2. 4. Mô hình vòng lặp *DES*

Trong *DES* biểu thức hàm *F* là :

$$F(R_{i-1}, K_i) = P-box(s-boxes(Expand(R_{i-1}) \oplus K_i))$$

Trong đó hàm *Expand (Mở rộng)* vừa mở rộng vừa mở rộng R_{i-1} từ 32 bit lên 48 bit. Hàm *S-boxes* lại nén 48bit xuống còn 32 bit. Hàm *P-box* là một hoán vị 32bit.

Hình 2.5 mô tả 1 hàm *f* trong 1 vòng lặp của thuật toán DES.



Hình 2. 5. Mô tả hàm *F* trong *DES*

Mô tả của các hàm trên như sau:

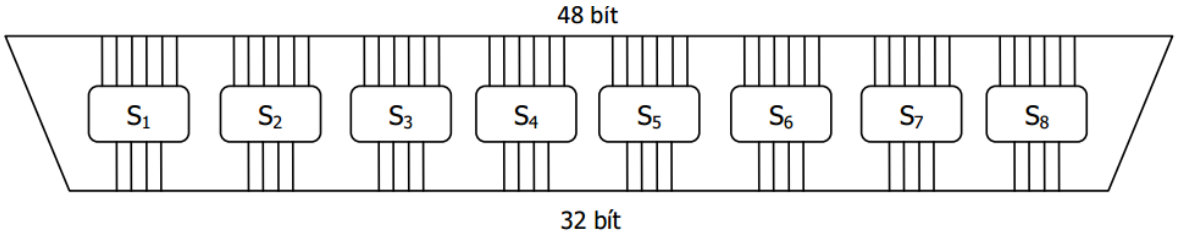
- Hàm *Expand*: hàm thực hiện hoán vị và mở rộng từ 32 bit lên 48 bit theo nguyên tắc sau: $\underbrace{r_1, r_2, r_3, ..., r_{32}}_{32bit} \rightarrow \underbrace{r_{32}, r_1, r_2, ..., r_1}_{48bit}$

Bảng 2. 7. Bảng *Expand DES*

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

48 bit

- Hàm *S-boxes*: Hàm *S-boxes* của *DES* biến đổi một số 48 bit thành một số 32 bit. Hàm *S-boxes* được chia thành 8 hàm *S-box* con, mỗi hàm biến đổi số 6 bit thành số 4 bit.



Hàm *S-box* đầu tiên, S_1 có quy tắc như bảng sau:

Bảng 2. 8. Bảng *S-box1*

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
b_0b_5	00	1110 0100 1101 0001 0010 1111 1011 1000 0011 1010 0110 1100 0101 1001 0000 0111
	01	0000 1111 0111 0100 1110 0010 1101 0001 1010 0110 1100 1011 1001 0101 0011 1000
	10	0100 0001 1110 1000 1101 0110 0010 1011 1111 1100 1001 0111 0011 1010 0101 0000
	11	1111 1100 1000 0010 0100 1001 0001 0111 0101 1011 0011 1110 1010 0000 0110 1101

Ví dụ : $\underbrace{000000}_{b_0b_1b_2b_3b_4b_5} \rightarrow \underset{row[1]col[1]}{1110}$, $\underbrace{100001}_{b_0b_1b_2b_3b_4b_5} \rightarrow \underset{row[4]col[1]}{1111}$

Tương tự các bảng $S_2 \rightarrow S_8$ sẽ biến đổi theo nguyên tắc của các bảng sau :

- Bảng S_2

Bảng 2. 9. Bảng *S-box2*

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
b_0b_5	00	1111 0001 1000 1110 0110 1011 0011 0100 1001 0111 0010 1101 1100 0000 0101 1010
	01	0011 1101 0100 0111 1111 0010 1000 1110 1100 0000 0001 1010 0110 1001 1011 0101
	10	0000 1110 0111 1011 1010 0100 1101 0001 0101 1000 1100 0110 1001 0011 0010 1111
	11	1101 0 1010 0001 0011 1111 0100 0010 1011 0110 0111 1100 0000 0101 1110 1001

- Bảng S_3

Bảng 2. 10. Bảng *S-box3*

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
b_0b_5	00	1010 0000 1001 1110 0110 0011 1111 0101 0001 1101 1100 0111 1011 0100 0010 1000
	01	1101 0111 0000 1001 0011 0100 0110 1010 0010 1000 0101 1110 1100 1011 1111 0001
	10	1101 0110 0100 1001 1000 1111 0011 0000 1011 0001 0010 1100 0101 1010 1110 0111
	11	0001 1010 1101 0000 0110 1001 1000 0111 0100 1111 1110 0011 1011 0101 0010 1100

- Bảng S_4

Bảng 2. 11. Bảng *S-box4*

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
b_0b_5	00	0111 1101 1110 0011 0000 0110 1001 1010 0001 0010 1000 0101 1011 1100 0100 1111
	01	1101 1000 1011 0101 0110 1111 0000 0011 0100 0111 0010 1100 0001 1010 1110 1001
	10	1010 0110 1001 0000 1100 1011 0111 1101 1111 0001 0011 1110 0101 0010 1000 0100
	11	0011 1111 0000 0110 1010 0001 1101 1000 1001 0100 0101 1011 1100 0111 0010 1110

- Bảng S_5

Bảng 2. 12. Bảng *S-box5*

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
b_0b_5	00	0010 1100 0100 0001 0111 1010 1011 0110 1000 0101 0011 1111 1101 0000 1110 1001
	01	1110 1011 0010 1100 0100 0111 1101 0001 0101 0000 1111 1010 0011 1001 1000 0110
	10	0100 0010 0001 1011 1010 1101 0111 1000 1111 1001 1100 0101 0110 0011 0000 1110
	11	1011 1000 1100 0111 0001 1110 0010 1101 0110 1111 0000 1001 1010 0100 0101 0011

- Bảng S_6

Bảng 2. 13. Bảng *S-box6*

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
b_0b_5	00	1100 0001 1010 1111 1001 0010 0110 1000 0000 1101 0011 0100 1110 0111 0101 1011
	01	1010 1111 0100 0010 0111 1100 1001 0101 0110 0001 1101 1110 0000 1011 0011 1000
	10	1001 1110 1111 0101 0010 1000 1100 0011 0111 0000 0100 1010 0001 1101 1011 0110
	11	0100 0011 0010 1100 1001 0101 1111 1010 1011 1110 0001 0111 0110 0000 1000 1101

- Bảng S_7

Bảng 2. 14. Bảng S-box7

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
b_0b_5	00	0100 1011 0010 1110 1111 0000 1000 1101 0011 1100 1001 0111 0101 1010 0110 0001
	01	1101 0000 1011 0111 0100 1001 0001 1010 1110 0011 0101 1100 0010 1111 1000 0110
	10	0001 0100 1011 1101 1100 0011 0111 1110 1010 1111 0110 1000 0000 0101 1001 0010
	11	0110 1011 1101 1000 0001 0100 1010 0111 1001 0101 0000 1111 1110 0010 0011 1100

- Bảng S8

Bảng 2. 15. Bảng S-box8

		$b_1b_2b_3b_4$
		0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
b_0b_5	00	1101 0010 1000 0100 0110 1111 1011 0001 1010 1001 0011 1110 0101 0000 1100 0111
	01	0001 1111 1101 1000 1010 0011 0111 0100 1100 0101 0110 1011 0000 1110 1001 0010
	10	0111 1011 0100 0001 1001 1100 1110 0010 0000 0110 1010 1101 1111 0011 0101 1000
	11	0010 0001 1110 0111 0100 1010 1000 1101 1111 1100 1001 0000 0011 0101 0110 1011

Ví dụ: Input *S-box1* là $\underline{100011}_2$. Bit số $b_0=1$ và bit $b_5=1 \rightarrow b_0b_5=11_2$ tương ứng với 3_{10}. Tương tự giá trị $b_1b_2b_3b4=0001_2=1_{10}$. Đối chiếu với bảng *S-box1* ở hàng 4, cột 2 nhận được giá trị *S-box1(100011)= 1100*$_2$.

Có thể thấy, mỗi hàm *S-box* con là một phép thay thế *Substitution*. Các hàm *S-box* con không khả nghịch, do đó hàm *S-boxes* cũng không khả nghịch. Sự phức tạp này của *S-boxes* là yếu tố chính làm cho *DES* có độ an toàn cao.

- *P-box*: hàm *P-box* cũng thực hiện hoán vị 32 bit đầu vào theo quy tắc:

Bảng 2. 16. Bảng *P-box*

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

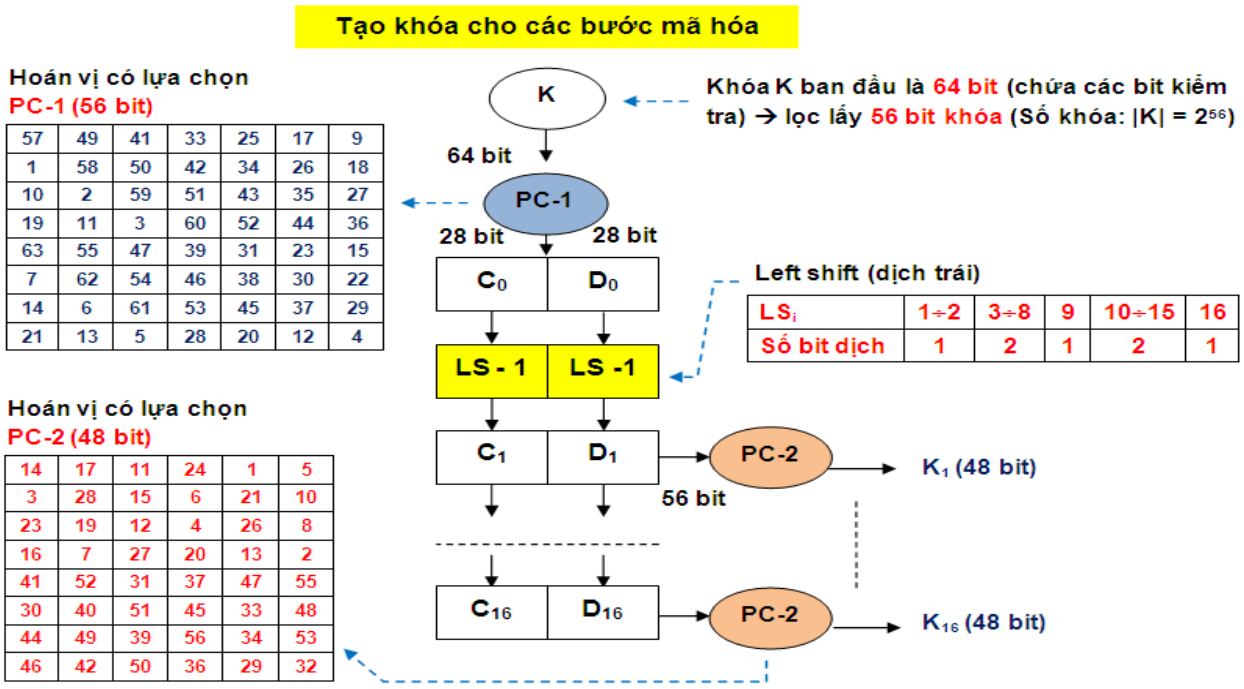
$$\underbrace{p_1, p_2, p_3 \ldots, p_{32}}_{32bit} \rightarrow \underbrace{p_{16}, p_7, p_{20} \ldots, p_{25}}_{32bit}$$

c) Thuật toán sinh khóa con

Thuật toán sinh khóa con cho từng vòng mã hóa của DES được thể hiện trong hình 2.6. Khóa *K* 64 bit ban đầu được rút trích và hoán vị thành một khóa 56 bit (tức chỉ sử dụng 56 bit) theo quy tắc nén khóa (xem bảng 2.17). Khóa *56* bit này được chia thành 2 nửa trái phải KL_0 và KR_0 , mỗi nửa có kích thước 28 bit. Tại vòng thứ i (i = 1, 2, 3,…,16), KL_{i-1} và KR_{i-1} được dịch vòng trái r_i bit để có được KL_i và KR_i, với r_i được định nghĩa như sau:

$$r_i = \begin{cases} 1 & if \quad i \in (1,2,9,16) \\ 2 & if \quad i \notin (1,2,9,16) \end{cases} \qquad (2.1)$$

Cuối cùng khóa K_i của mỗi vòng được tạo ra bằng cách hoán vị và nén 56 bit của KL_i và KR_i thành 48 bit (xem bảng 2.18)



Hình 2. 6. Mô hình các bước tạo khóa của *DES*

Bảng 2. 17. Bảng nén khóa 64bit xuống 56bit *DES*

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

Nửa trên

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Nửa dưới

Bảng 2. 18. Bảng nén khóa 56 bit xuống 48 bit *DES*

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

Ví dụ mã hóa DES:

Cho Thông điệp P : $123456ABCD132536_{16}$

Khóa K : $AABB09182736CCDD_{16}$

Bảng 2.19 trình bày kết quả qua các vòng lặp của *DES*

Bảng 2. 19. Ví dụ mã hóa DES

Thông điệp : *123456ABCD132536*			
Sau hoán vị khởi tạo :*14A7D6818CA18AD*			
Chia hai nửa trái phải L_0=*14A7D681*, R_0= *18CA18AD*			
Vòng lặp	Nửa trái	Nửa phải	Khóa sinh ra
Vòng lặp	*Nửa trái*	*Nửa phải*	*Khóa sinh ra*
Round1	*18CA18AD*	*5A78E394*	*194CD072DE8C*
Round2	*5A78E394*	*4A1210F6*	*4568581ABCCE*
Round3	*4A1210F6*	*B8089591*	*06EDA4ACF5B5*
Round4	*B8089591*	*236889C2*	*DA2D032B6EE3*
Round5	*236779C2*	*2E8F9C65*	*69A629FEC913*
Round6	*A15A4B87*	*2E8F9C65*	*C1948E87485E*
Round7	*2E8F9C65*	*A15A4B87*	*708AD2DDB3C0*
Round8	*A9FC20A3*	*308BEE97*	*34F822F0C66D*
Round9	*308BEE97*	*10AF9D37*	*84BB4473DCCC*
Round10	*10AF9D37*	*6CA6CB20*	*02765708B5BF*
Round11	*6CA6CB20*	*FF3C485F*	*6D5560AF7CA5*
Round12	*FF3C485F*	*22A5963B*	*C2C1E96A4BF3*
Round13	*22A5963B*	*387CCDAA*	*99C31397C91F*
Round14	*387CCDAA*	*BD2DD2AB*	*251B8BC717D0*
Round15	*BD2DD2AB*	*CF26B472*	*3330C5D9A36D*

Round16	*CF26B472*	*19BA9212*	*181C5D75C66D*
Giá trị thu được sau khi lặp 16 vòng : *CF26B47219BA9212*			
Bản mã (Sau khi thực hiện hoán vị kết thúc): *C0B7A8D05F3A829C*			

2.3.1.2. Triple DES (3DES)

Một trong những cách để khắc phục yếu điểm về kích thước khóa ngắn của mã hóa *DES* là sử dụng mã hóa *DES* nhiều lần với các khóa khác nhau cho cùng một thông điệp. Đơn giản nhất là dùng *DES* hai lần với hai khóa khác nhau, cách thức này được gọi là Double DES. Công thức 2.2 mô tả về quy trình mã hóa của thuật toán Double DES.

$$C = E(E(P,K_1),K_2) \qquad (2.2)$$

Theo công thức 2.2 có thể hiểu Double *DES* dùng một khóa có kích thước là 112 bit, chỉ có một hạn chế là tốc độ chậm hơn *DES* vì phải dùng *DES* hai lần. Tuy nhiên người ta đã tìm được một phương pháp tấn công *Double DES* có tên gọi là gặp-nhau-ở-giữa (*meet-in-the-middle*). Đây là một phương pháp tấn công *chosen-plaintext* (trình bày ở chương 1). Vì vậy người chọn dùng *DES* ba lần với ba khóa khác nhau, cách thức này được gọi là *Triple DES (*xem hình 2.7*)*. Công thức 2.3 mô tả quá trình mã hóa của thuật toán *Triple DES*.

$$C = E(\ D(\ E(\ P,\ K_1)\ ,K_2),\ K_3) \qquad (2.3)$$

Từ công thức 2.3 có thể thấy chiều dài khóa là 168 bit sẽ gây phức tạp hơn nhiều cho việc phá mã bằng phương pháp tấn công gặp-nhau-ở-giữa. Về thuật toán *Triple DES* có nhiều phương pháp khác nhau để lựa chon các cặp khóa xử dụng mà vẫn đảm bảo an toàn cần thiết. Ví dụ công thức 2.4 mô tả thuật toán *Triple DES* chỉ dùng hai khóa K_1, K_2.

$$C = E(D(E(P,K_1),K_2),K_1) \qquad (2.4)$$

Nguyên nhân của việc dùng *EDE* thay cho *EEE* là nếu với *K1 = K2 = K* thì:

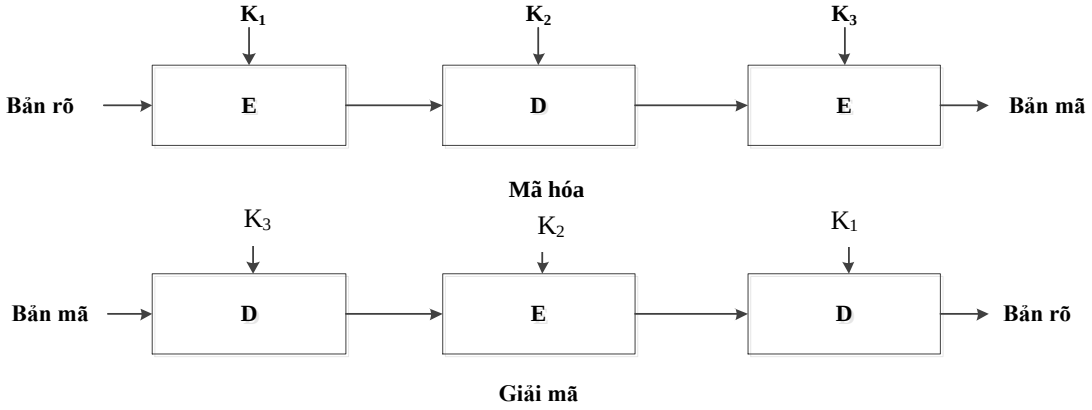
$$C = E(E(E(P,K),K),K) = E(P,K) \qquad (2.5)$$

Nghĩa là *Triple DES* suy giảm thành *DES*.

- Các lựa chọn khóa :

- Lựa chọn 1: cả 3 khóa độc lập (168 bit)
- Lựa chọn 2: ví dụ K_1 và K_2 độc lập, $K_3 = K_1$ (112 bit)
- Lựa chọn 3: 3 khóa giống nhau, $K_1 = K_2 = K_3$ (56 bit)

- Sơ đồ mã hóa và giải mã *Triple DES*



Hình 2. 7. Sơ đồ mã hóa và giải mã *Triple DES*

2.3.3. Thuật toán mã hóa AES

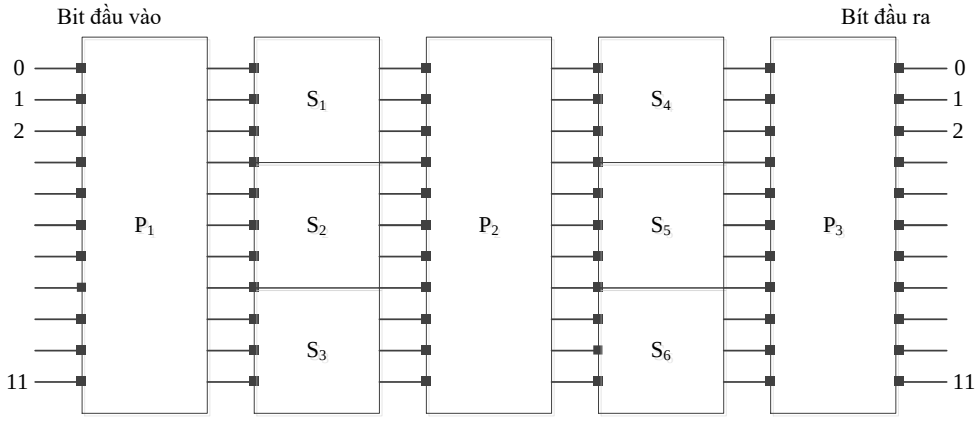
2.3.3.1. Nguồn gốc ra đời của thuật toán AES

Năm 1997 nhận thấy nguy cơ mã hóa *DES* là kích thước khóa ngắn có thể bị phá vỡ, do đó Viện chuẩn quốc gia Hoa Kỳ *US NIST* đã kêu gọi xây dựng một phương pháp mã hóa mới nhằm thay thế thuật toán này. Đến tháng 10/2000 thuật toán có tên là Rijndael được chọn làm mật mã nâng cao và xuất bản là chuẩn *FIPS PUB* 197. Vào 11/2001 và được đổi tên thành *Andvanced Encryption Standard* (*AES*). Cơ chế mã hóa của AES sử dụng mạng thay thế hoàn vị (*SPN - Substitution-Permutation Network*). Đầu vào và đầu ra là khối dữ liệu 128 bit. Khóa dùng để mã hóa và giải mã của giải thuật AES là 128 ,192 hoặc 256 bit. Kích thước khóa quyết định số vòng lặp trong mã hóa và giải mã.

2.3.3.2. Mạng thay thế hoán vị

Trong thực tế, cần tìm phương pháp, cách thức sao cho chỉ cần dùng một khóa có kích thước ngắn để giả lập một bảng tra cứu có độ an toàn xấp xỉ độ an toàn của mã khối lý tưởng. Cách thực hiện là kết hợp hai hay nhiều mã hóa đơn giản lại với nhau để tạo thành một mã hóa tổng (*product cipher*), trong đó mã hóa tổng an toàn hơn rất nhiều so với các

mã hóa thành phần. Các mã hóa đơn giản thường là phép thay thế (*substitution, S-box*) và hoán vị (*Permutation, P-box*). Do đó người ta hay gọi mã hóa tổng là *Substitution-Permutation Network* (mạng *SPN*). Hình dưới minh họa một mạng *SPN*.



Hình 2. 8. Sơ đồ SPN

Việc kết hợp các *S-box* và *P-box* tạo ra hai tính chất quan trọng của mã hóa là tính khuếch tán (Tính chất này có được dựa vào sử dụng *P-box* kết hợp *S-box*) và tính gây lẫn (Tính chất này có được dựa vào sử dụng *S-box*). Hai tính chất này do *Claude Shannon* giới thiệu vào năm 1946, và là cơ sở của tất cả các mã khối hiện nay. Chương 1 bài giảng đã trình bày 2 tính chất này.

2.3.3.3. Biểu diễn dữ liệu trong AES

Biểu diễn ma trận byte

AES là một mã khối, nhưng khác với các mã khối khác được biết đến trước đây (*DES, IDEA*,…), dữ liệu trong *AES* không được biểu diễn dưới dạng một mảng các byte hay các bit mà được biểu diễn dưới dạng một ma trận *4xNb* và được gọi là mảng trạng thái (*state*). Trong đó, đối với *AES, Nb* luôn có giá trị bằng 4. Trong khi thuật toán *Rijndael* hỗ trợ ba giá trị của *Nb* là 4, 6, 8 tương ứng với kích thước khối 128, 192 và 256 bit. Biểu diễn ma trận trạng thái của *AES* được thể hiện trong biểu thức 2.6.

Dữ liệu đầu vào được đọc vào ma trận *state* theo từng cột, theo thứ tự từ trên xuống dưới, từ trái qua phải. Dữ liệu đầu ra được đọc từ ma trận cũng theo quy tắc trên.

$$s = \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix} \quad (2.6)$$

Khóa vòng trong *AES* cũng được biểu diễn hoàn toàn tương tự như cách biểu diễn dữ liệu (xem công thức 2.8). Tuy nhiên, tùy vào kích thước khóa mà số cột của ma trận khóa vòng *Nk* sẽ khác nhau. Cụ thể, *Nk* nhận các giá trị 4, 6, 8 tương ứng với các kích thước khóa là 128, 192 và 256 bit. Đây chính là điểm mạnh của thuật toán AES trong vấn đề mở rộng khóa.

$$k = \begin{pmatrix} k_{00} & k_{01} & \dots & k_{0Nk} \\ k_{10} & k_{11} & \dots & k_{1Nk} \\ k_{20} & k_{21} & \dots & k_{2Nk} \\ k_{30} & k_{31} & \dots & k_{3Nk} \end{pmatrix} \quad (2.8)$$

Số vòng lặp *Nr*

Số vòng lặp ký hiệu là N_r, phụ thuộc vào hai đại lượng N_b và N_k. Vì N_b trong *AES* có giá trị cố định nên N_r chỉ phụ thuộc vào N_k. Giá trị của N_r tương ứng với ba giá trị của N_k là N_r = 10, 12, 14. Cụ thể, giá trị N_r được xác định bởi bảng bên dưới:

N_k	4	6	8
N_r	10	12	14

Khái niệm từ (*Word*) trong *AES*

Bốn byte trên mỗi cột trong mảng trạng thái *state* tạo thành 1 từ 32 bit, trong đó số thứ tự của hàng r ($0 \le r < 4$) cho biết chỉ số của bốn byte trong mỗi từ. Từ định nghĩa *state* ở trên có thể coi *state* là mảng một chiều chứa các từ 32 bit $s_0 \dots s_3$ bởi công thức 2.9.

$$\begin{aligned} s_0 &= s_{00} s_{10} s_{20} s_{30} \\ s_1 &= s_{01} s_{11} s_{21} s_{31} \\ s_2 &= s_{02} s_{12} s_{22} s_{32} \\ s_3 &= s_{03} s_{13} s_{23} s_{33} \end{aligned} \quad (2.9)$$

Tương tự như đối với mảng khóa cũng có thể biểu diễn thành mảng một chiều chứa các từ 32 bit như công thức dưới đây với số lượng từ khóa phụ thuộc vào N_k (N_k = 4, 6, 8).

$$k_0, k_1 ... k_{Nk}$$

Như vậy mảng khóa vòng sẽ được biểu diễn bởi: $k_0, k_1 ... k_{(Nr+1)*4}$

2.3.3.4. Cơ sở toán học của AES

Phép cộng trên trường GF(2^8): Chính là phép *XOR* hay phép cộng theo modulo 2 giữa các bit tương ứng giữa 2 byte. Nghĩa là:

Giả sử: $A=a_7a_6a_5a_4a_3a_2a_1a_0$

$B=b_7b_6b_5b_4b_3b_2b_1b_0$

$C=A+B=c_7c_6c_5c_4c_3c_2c_1c_0$

trong đó: $c_i=(a_i+b_i) mod\ 2\ (0<=i<=7)$

<u>Ví dụ:</u> tổng của $A=73_H$ và $4E_H$ là:

- Dạng nhị phân 01110011 + 01001110 = 00111101
- Dạng đa thức $(x^6+x^5+x^4+x+1)$ + $(x^6+x^3+x^2+x)$ = $(x^5+x^4+x^3+x^2+1)$
- Dạng thập lục phân 73_H + $4E_H$ = $3D_H$

Phép nhân trên trường *GF (2^8)*: Phép nhân được thực hiện trên trường *GF (2^8)* bằng cách nhân hai đa thức rút gọn theo modulo của một đa thức bất khả quy *m(x).* Đa thức bất khả quy là đa thức chỉ có ước là 1 và chính nó. Trong *AES* đa thức bất khả quy này là:

$$m(x)=x^8+x^4+x^3+x+1$$

Giả sử: $a(x)=a_7x^7+a_6x^6+a_5x^5+a_4x^4+a_3x^3+a_2x^2+a_1x^1+1$

$b(x)=b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+1$

=> $c(x)=a(x).b(x)\ mod\ m(x)$

Ví dụ: tích của $A=73_H$ *và* $B=4E_H$ *là* $C=AB \bmod m(x)$

- Dạng nhị phân: 01110011 • 01001110 = 01011011
- Dạng đa thức: $(x^6+x^5+x^4+x+1)$ • $(x^6+x^3+x^2+x)$ = $(x^6+x^4+x^3+x+1)$
- Dạng thập lục phân: 73_H • $4E_H$ = $5B_H$

2.3.3.5. *Thuật toán AES*

Thuật toán *AES* khá phức tạp, được mô tả khái quát gồm 3 bước như sau:

- Vòng khởi tạo chỉ gồm phép *AddRoundKey*
- Vòng lặp gồm 4 phép biến đổi lần lượt: *SubBytes, ShiftRows, MixColumns, AddRoundKey*.
- Vòng cuối gồm các phép biến đổi giống vòng lặp và không có phép *MixColumns*.

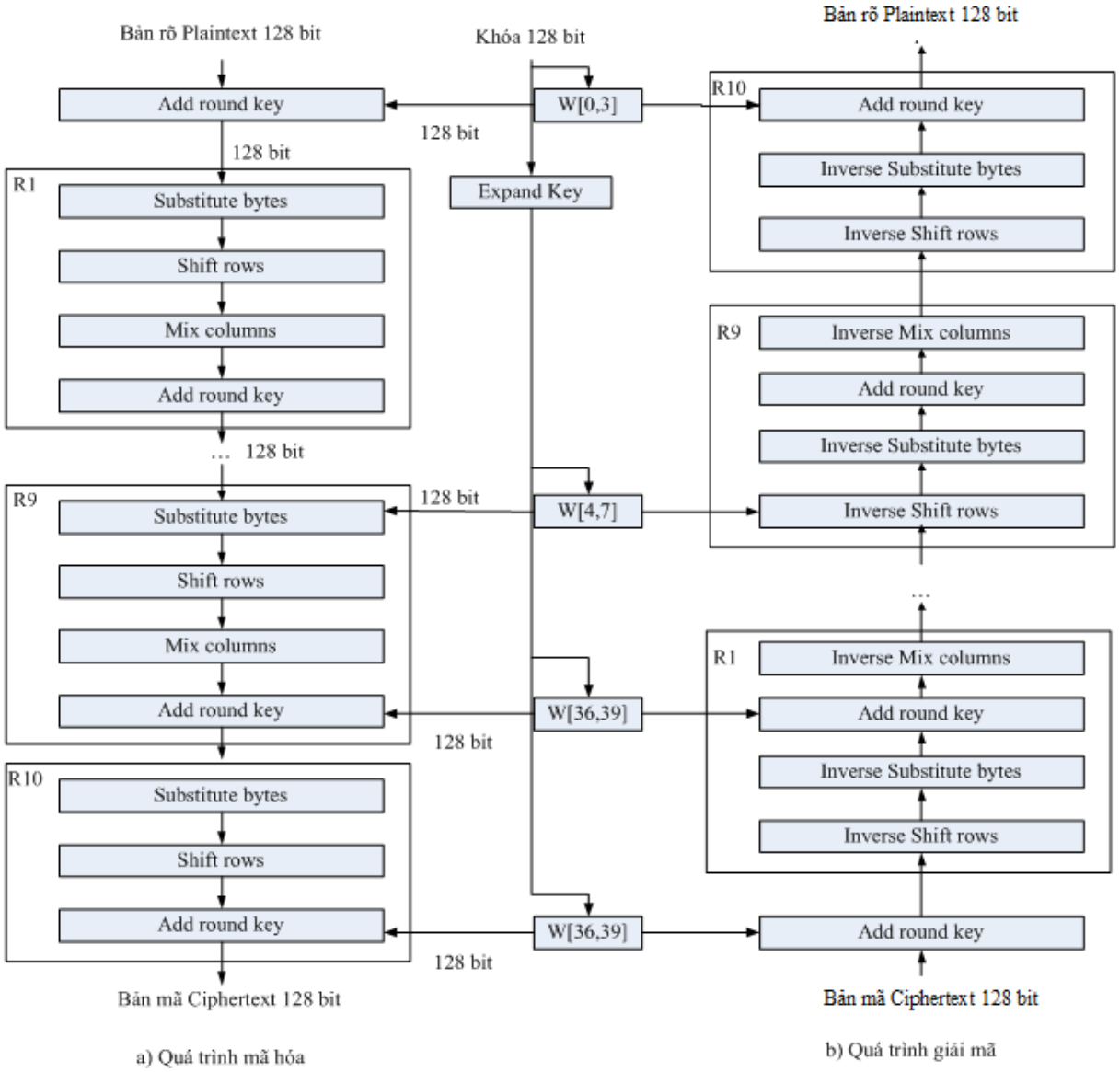
Đặc tả chi tiết thuật toán mã hóa *AES*

Cấu trúc sơ đồ mã hóa được thể hiện trên Hình 2.9 gồm: vòng khởi tạo, *Nr-1* vòng lặp và vòng kết thúc. Trong đó vòng khởi tạo chỉ có phép biến đổi *AddRounKey*, vòng lặp gồm 4 phép biến đổi chính: *SubBytes, ShiftRows, MixColumns, AddRounKey*; vòng kết thúc khác với vòng lặp chính ở chỗ không có phép *MixColumns.*

Riêng đối với cấu trúc giải mã trong *AES* gồm 2 chế độ giải mã:

- Ở cấu trúc giải mã ngược, gồm vòng khởi tạo, *Nr-1* vòng lặp và vòng kết thúc. Trong đó vòng khởi tạo chỉ có phép biến đổi *AddRounKey*, vòng lặp gồm lần lượt 4 phép biến đổi chính: *InvShiftRows, InvSubBytes, AddRounKey, InvMixColumns*; vòng kết thúc khác với vòng lặp chính ở chỗ không có phép *InvMixColumns.*
- Ngược lại với cấu trúc giải mã ngược là cấu trúc giải mã xuôi, việc ngược lại thể hiện ở điểm: trong cấu trúc giải mã xuôi việc sắp xếp các phép biến đổi ngược giống hệt với cấu trúc mã hóa, cụ thể bao gồm: vòng khởi tạo, Nr-1 vòng lặp và vòng kết thúc. Trong đó vòng khởi là phép *AddRounKey*; ở vòng lặp thứ tự các phép biến đổi ngược lần lượt là: *InvSubBytes, InvShiftRows, InvMixColumns, AddRounKey*; vòng kết thúc giống vòng lặp nhưng được lược bỏ phép *InvMixColumns*.

Một điểm khác biệt nữa trong hai cấu trúc giải mã ngược và giải mã xuôi đó là: trong giải mã ngược khóa vòng giải mã chính là khóa vòng mã hóa với thứ tự đảo ngược; riêng trong giải mã xuôi thì khóa giải mã ngoài việc đảo ngược thứ tự khóa vòng mã hóa còn phải thực hiện phép *InvMixColumns* đối với các khóa vòng của vòng lặp giải mã.



Hình 2. 9. Sơ đồ mã hóa giải mã *AES*

Từ mô hình mã hóa của thuật toán AES trong hình 2.9, mô tả giải thuật AES gồm các bước chính trong vòng lặp như sau:

SubBytes và InvSubBytes

- *Phép biến đổi SubBytes:* Là phép thay thế byte phi tuyến tính, ở phép thay thế này nó tác động độc lập đến từng byte trong trạng thái hiện hành. Phép biến đổi *SubBytes* được thực hiện bằng cách tra cứu bảng thay thế (S-box) với tham số đầu vào là các byte trong bảng trạng thái. S-box được xây dựng như sau:

Bước 1: Điền các con số từ 0 đến 255 vào bảng theo từng hàng. Vậy hàng 0 gồm các con số {00}, {01}, …{0F} (thập lục phân). Hàng 1 gồm các con số {10}, {11},…, {1F}. Điều này có nghĩa là tại hàng x cột y có giá trị {xy}.

Bước 2: Thay thế mỗi byte trong bảng bằng giá trị nghịch đảo trong trường $GF(2^8)$. Quy ước nghịch đảo của {00} cũng là {00}.

Bước 3: Mỗi byte trong ma trận *state* được thay thế bởi 1 byte trong *Rijndael* S-box, hay $b_{ij} = S(a_{ij})$.

$$b_i' = b_i \oplus b_{(i+4) mod\ 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \qquad (2.10)$$

trong đó, $0 \le i \le 8$ là bit thứ i của byte b tương ứng và c_i là bit thứ thứ i của byte c với giá trị {63} hay {01100011}.

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad (2.11)$$

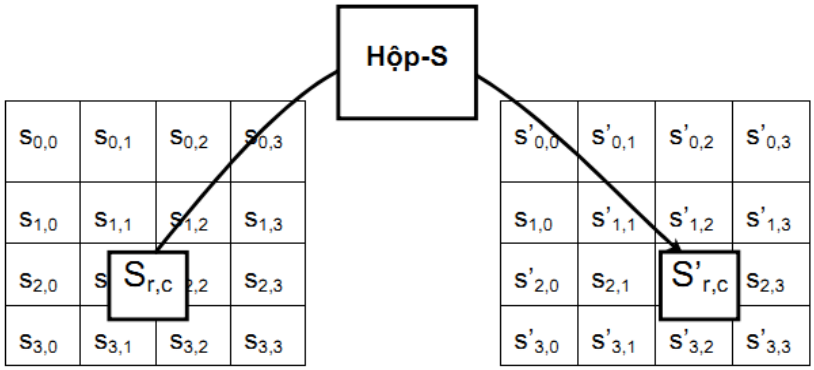
Trong đó phép cộng thực hiện như phép XOR. Bảng 2.20 trình bày nội dung bảng S-box sau khi tính toán.

Bảng 2. 20. Bảng S-box AES

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7E	F2	6b	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15

3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Hình 2.10 minh họa tác động của phép SubByte () lên mảng trạng thái *State*:



Hình 2. 10. Phép biến đổi SubByte

Như vậy nếu giá trị byte cần thay thế là $S_{r,c}=\{xy\}$ thì sẽ dóng vào *S-box* hàng *x* cột *y* thì thu được byte thay thế $S'_{r,c}$.

Ví dụ: phép *SubByte*

ea	04	65	85
80	50	5d	96
5e	33	98	b0
f0	ab	ad	c5

SubByte →

87	f2	4d	97
ec	6e	4c	90
4a	c3	46	e7
8c	d8	95	a6

- Phép biến đổi ngược *InvSubBytes*: là phép thay thế biến đổi ngược với *SubBytes*. Là một phép thay thế byte, các byte thay thế được thực hiện bằng cách tra bảng thay thế ngược *IS*. Bảng thay thế ngược *IS* này được xây dựng như sau:

Trước tiên, cũng phải xây dựng một bảng *Inverse SubBytes (IS- box)*. Nghĩa là nếu với đầu vào *{95}, S-box* cho ra kết quả *{2A}*, thì với đầu vào là *{2A}, IS* sẽ cho ra lại kết quả *{95}*. Việc xây dựng hộp *IS* cũng giống như xây dựng *S-box* tại bước 1 và bước 2. Tại bước 3, *IS* thực hiện phép thay thế sau:

$$b_i = b'_i \oplus b'_{(i+2) mod\ 8} \oplus b'_{(i+5) \bmod 8} \oplus b'_{(i+7) \bmod 8} \oplus d_i \qquad (2.12)$$

Với d_i là bit thứ i của số *{05}* tức $d_7\ d_6\ d_0 = 00000101$. Và phép nhân ma trận tương đương $B = YB' \oplus D$:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Bảng 2.21 trình bày nội dung hộp thay thế ngược *IS*

Bảng 2. 21. Hộp thay thế ngược *IS*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB

2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

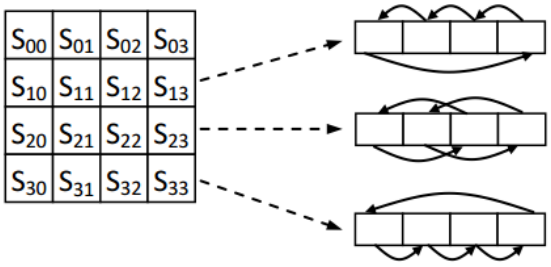
Như vậy: phép biến đổi *InvSubBytes* thực hiện như sau: Mỗi byte trong ma trận state *S*, dưới dạng thập lục phân là *{xy}*, được thay thế bằng giá trị trong bảng *IS* tại dòng *x* cột *y*.

- *Mục đích của phép biến đổi SubBytes: S-box* dùng để chống lại hình thức tấn công thám mã vi sai và thám mã tuyến tính. Giữa input và output của phép *Substitute* bytes không thể mô tả bằng một công thức toán đơn giản.

ShiftRows* và *InvShiftRows

- *Phép biến đổi ShiftRows:* Thao tác *ShiftRows* (xem hình 2.11) thực hiện hoán vị các byte trong ma trận *state* theo cách thức sau:

  - ✓ Dòng thứ nhất giữ nguyên
  - ✓ Dòng thứ 2 dịch vòng trái 1 byte
  - ✓ Dòng thứ 3 dịch vòng trái 2 byte
  - ✓ Dòng thứ 4 dịch vòng trái 3 byte



Hình 2. 11. Phép biến đổi ShiftRows

- *Phép biến đổi InvShiftRows:* Phép biến đổi *InvShiftRows* thực hiện ngược lại với phép *ShiftRows*, nghĩa là:

  ✓ Dòng thứ nhất giữ nguyên

  ✓ Dòng thứ 2 dịch vòng phải 1 byte

  ✓ Dòng thứ 3 dịch vòng phải 2 byte

  ✓ Dòng thứ 4 dịch vòng phải 3 byte

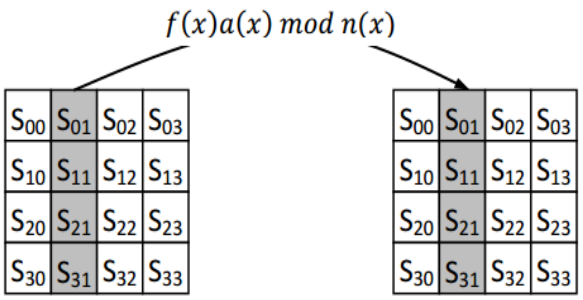
- *Mục đích của ShiftRows:* Xáo trộn các byte để tạo các cột khác nhau trước khi sử dụng cột cho thao tác *MixColumns.*

MixColumns* và *InvMixColumns

- *Phép biến đổi MixColumns:* Phép biến đổi *MixColumns* thực hiện biến đổi độc lập từng cột trong ma trận *state* bằng một phép nhân đa thức. Mỗi cột của state đươc coi là biểu diễn của một đa thức *f(x)* trong *GF* (2^8) như vậy phép biến đổi *MixColumns* chính là phép nhân theo *modulo* với x^4+1 với một đa thức cố định định nghĩa như sau:

$$a(x)=\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \qquad (2.13)$$

Hình 2.12 mô tả thao tác trên các cột của phép biến đổi *MixColumn*:



Hình 2. 12. Phép biến đổi MixColumn

Phép nhân đa thức trên có thể biểu diễn dưới dạng phép nhân ma trận như sau:

$$\begin{bmatrix} s'_{01} \\ s'_{11} \\ s'_{21} \\ s'_{31} \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} s_{01} \\ s_{11} \\ s_{21} \\ s_{31} \end{bmatrix} \quad (2.14)$$

Hình 2.13 trình bày một ví dụ về phép *MixColumns:*

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	13	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	B

Hình 2. 13. Ví dụ về phép MixColumn

- *Phép biến đổi ngược InvMixColumns:* Là phép biến đổi ngược với phép biến đổi *MixColumns. InvMixColumns* cũng thực hiện thao tác theo từng cột của trạng thái, xem mỗi cột như một đa thức bậc 3 gồm 4 hạng tử trên trường *GF* (2^8). Các cột của phép *InvMixColumns* được nhân theo modulo (x^4+1) với đa thức nghịch đảo *a(x)* chính là đa thức $a^{-1}(x)$ được định nghĩa:

$$a^{-1}(x)=\{0B\}.x^3+\{0D\}.x^2+\{09\}.x^1+\{0E\} \quad (2.14)$$

Như vậy phép *InvMixColumns* cũng được biểu diễn tương đương với phép nhân ma trận sau:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 0E\ 0B\ 0D\ 09 \\ 09\ 0E\ 0B\ 0D \\ 0D\ 09\ 0E\ 0B \\ 0B\ 0D\ 09\ 0E \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}$$

trong đó c là cột thứ c $(0 \le c \le 3)$ của ma trận trạng thái *state*

- *Mục đích của MixColumns:*

Việc coi mỗi cột là một đa thức bậc 3, rồi nhân mỗi cột với đa thức *a(x)* sau đó *modulo* $(x^4 + 1)$ đã làm cho mỗi byte trong cột kết quả đều phụ thuộc vào bốn byte trong cột ban đầu. Thao tác *MixColumns* kết hợp với *ShiftRows* đảm bảo rằng sau một vài vòng biến đổi, 128 bit trong kết quả đều phụ thuộc vào tất cả 128 bit ban đầu. Điều này tạo ra tính khuếch tán (*diffusion*) cần thiết cho mã hóa.

AddRoundKey

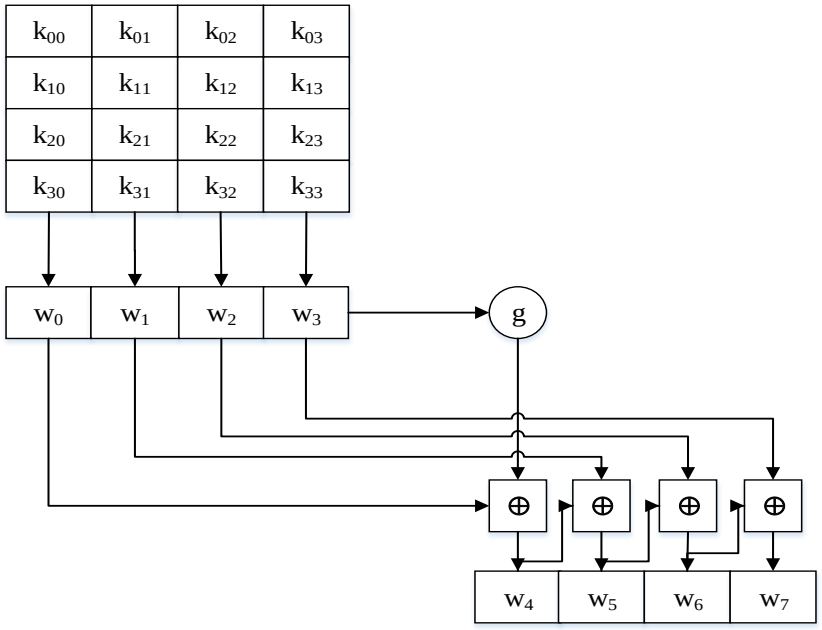
Trong thao tác *AddRoundKey*, 128 bit của ma trận *state* sẽ được *XOR* với 128 bit của khóa con của từng vòng. Vì sử dụng phép *XOR* nên phép biến đổi ngược của *AddRoundKey* trong cấu trúc giải mã cũng chính là *AddRoundKey*.

Việc kết hợp với khóa bí mật tạo ra tính làm rối (*confusion*) của mã hóa. Sự phức tạp của thao tác mở rộng khóa (*KeySchedule*) giúp gia tăng tính làm rối này.

Mở rộng khóa (*ExpandKey*)

ExpandKey là thao tác tạo lược đồ khóa hay mở rộng khóa, tạo ra *Nr+1* khóa vòng từ khóa chính *K*, mỗi khóa vòng gồm N_b từ 32 bit, trong đó đối với *AES* thì N*b* = 4, còn N*r* được xác định theo. Các phép biến đổi để tạo khóa vòng trong *ExpandKey* là khác nhau đối với các giá trị khác nhau của kích thước khóa *K*.

Sau đây là việc mở rộng khóa đối với khóa mã 128 bit



Hình 2. 14. Biểu diễn sinh N_k khóa đầu tiên đối với khóa mã 128 bit

Trong thao tác mở rộng khóa với khóa mã 128 bit có đầu vào là 16 byte (4 *word*) của khóa mã, và sinh ra một mảng khóa vòng (*Nr+1*)x4=44 từ (*word)* hay 176 byte. 44 word này được sử dụng cho 11 vòng mã hóa của *AES*, mỗi vòng dùng 4 *word*.

Từ bốn word đầu vào $w_0w_1w_2w_3$, trong lần lặp đầu tiên thao tác *ExpandKey* sinh ra bốn word $w_4w_5w_6w_7$, lần lặp thứ 2 từ $w_4w_5w_6w_7$ sinh ra $w_8w_9w_{10}w_{11}$, cứ như thế cho đến lần lặp thứ 10 (tùy thuộc chiều dài khóa) sinh ra bốn word cuối cùng $w_{40}w_{41}w_{42}w_{43}$ như hình vẽ bên dưới.

Trong mỗi lần lặp để sinh ra 4 word, word đầu tiên sinh ra theo quy tắc $w_i = w_{i-4} \oplus g$ với g = $SubWord(RotWord(w_{i-1})) \oplus Rcon[_{i/4}]$. Ba word tiếp theo sinh ra theo quy tắc $wi = w_{i-4} \oplus w_{i-1}$. Sau đây, sẽ tìm hiểu các hàm SubWord, RotWord và mảng Rcon.

- ✓ *RotWord*: dịch vòng trái một byte. Giả sử word đầu vào có 4 byte là [b_0, b_1, b_2, b_3] thì kết quả của *RotWord* là [b_1, b_2, b_3, b_4].
- ✓ *SubWord*: thay thế mỗi byte trong word đầu vào bằng cách tra cứu bảng S-box trong thao tác Substitute Bytes.

- ✓ *Rcon*: là một mảng hằng số. Mảng này gồm 10 word ứng với 10 vòng AES. Bốn byte của một phần tử *Rcon* [j] là (*RC*[j], 0, 0, 0) với *RC*[j] là mảng 10 byte như sau (với *RC[j] = RC[j-1]*2* với phép nhân thực hiện trong $GF(2^8)$):

j	1	2	3	4	5	6	7	8	9	10
RC[j]	1	2	4	8	1	20	4	80	1B	36

- *Ví du ExpandKey:*

Cho key ban đầu : $(24\ 75\ A2\ B3\ 34\ 75\ 56\ 88\ 31\ E2\ 12\ 00\ 13\ AA\ 54\ 87)_{16}$. Bảng 2.22 chính là quá trình sinh khóa.

- *Mục đích của ExpandKey:* dùng để chống lại *known-plaintext attack*
  - ✓ Biết một số bit của khóa hay khóa con cũng không thể tính các bit còn lại.
  - ✓ Không thể tính ngược: biết một khóa con cũng không thể tính lại các khóa con trước đó.
  - ✓ Tính khuếch tán: một bit của khóa chính tác động lên tất cả các bit của các khóa con.

Bảng 2. 22. Bảng kết quả của quá trình sinh khóa

Round	T giá trị trung gian lưu *w*	First word in the round	Seconds word in the round	Third word in the round	Fourth word in the round
		w00=2475A2B3	w01=34755688	w02=31E21200	w03=13AA5487
1	AD20177D	w04=8955B5CE	w05=BD20E346	w06=8CC2F146	w07=9F68A5C1
2	470678DB	w08=CE53CD15	w09=73732E53	w10=FFB1DF15	w11=60D97AD4
3	31DA48DO	w12=FF8985C5	w13=8CFAAB96	w14=734B7483	w15=13920E57
4	47AB5B7D	w16=B822DEB8	w17=34D8752E	W18=479301AD	w19=54010FFA
5	6C762D20	w20=D454F398	w21=E08C86B6	w22=A71F871B	w23=F31E88E1
6	52C4F80D	w24=86900B95	w25=661C8D23	w26=C1030A38	w27=321D82D9
7	E4133523	w28=62833EB6	w29=049FB395	w30=C59CB9AD	w31=F7813B74

8	8CE29268	w32=EE61ACDE	w33=EAFE1F4B	w34=2F62A6E6	w35=D8E39D92
9	OA5E4F61	w36=E43FE3BF	w37=OEC1FCF4	w38=21A35A12	w39=F940C780
10	3PC6CD99	w40=DBF92E26	w41=D538D2D2	w42=F49B88CO	w43=ODDB4F40

Kết luận: Phương pháp mã hóa *AES* đơn giản, có thể thực hiện hiệu quả trên các vi xử lý 8 bit (dùng trong *smartcard*) cũng như trên các vi xử lý 32 bit, chỉ dùng phép *XOR* và phép *Shift* bit. Đây chính là yếu tố cơ bản để phương pháp này được chọn làm chuẩn mã hóa của *Hoa Kỳ*.

2.4. CÁC THUẬT TOÁN MÃ HÓA DÒNG ĐỐI XỨNG

2.4.1. Tổng quan về mã dòng

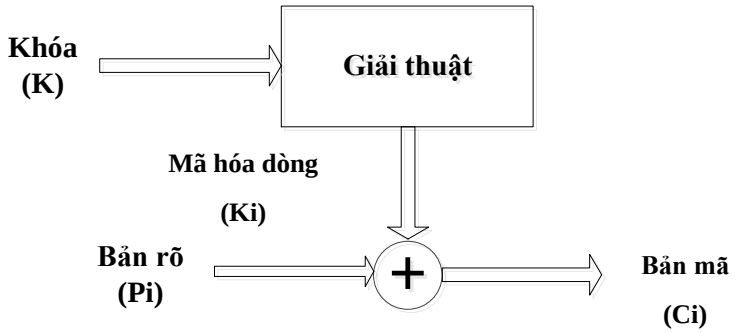
Các thuật toán mã hóa dòng là một trong những phương pháp mã hóa hiện đại đang được ứng dụng nhiều hiện nay. Mã dòng có một số đặc trưng cơ bản như sau:

- Kích thước một đơn vị mã hóa là : *k* bit.
- Thông điệp được chia thành các đơn vị mã hóa $P \rightarrow P_0P_1P_2...P_{n-1}$ <***length***$(P_i) = k$ *bit*>
- Một bộ sinh số ngẫu nhiên: Dùng khóa *K* ban đầu để sinh ra các số ngẫu nhiên có kích thước bằng kích thước của đơn vị mã hóa :

$$StreamCipher(K) \rightarrow S = S_0S_1S_2...S_{n-1} <\textbf{length}\ (S_i) = k\ bit>$$

- Mỗi đơn vị mã hóa được XOR tương ứng với số ngẫu nhiên để cho ra bản mã.

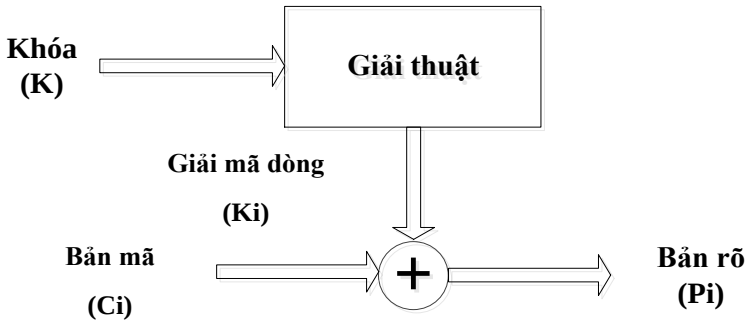
$$C_0=P_0 \oplus S_0\ ,\ C_1=P_1 \oplus S_1\ .....,\ C_{n-1}=P_{n-1} \oplus S_{n-1} \rightarrow C=C_0C_1C_{2...}C_{n-1;}$$



Hình 2. 15. Sơ đồ mã hóa của mã dòng

- Quá trình giải mã là một quá trình ngược lại, bản mã được XOR với bộ sinh số ngẫu nhiên cho ra thông điệp.

$$P_0=C_0 \oplus S_0\ ,\ P_1=C_1 \oplus S_1\ \ldots\ldots,\ P_{n-1}=C_{n-1} \oplus S_{n-1} \rightarrow P=P_0P_1P_{2\ldots}P_{n-1;}$$



Hình 2. 16. Sơ đồ giải mã của mã dòng

Các thuật toán mã hóa dòng hiện nay ngoài việc mang trong mình các đặc trưng cơ bản như trên thì còn phải đảm bảo đủ 3 tính chất là:

- Chiều dài khóa bằng chiều dài bản rõ;
- Khóa sinh ngẫu nhiên;
- Khóa chỉ được sử dụng một lần.

Để hiểu rõ hơn về đặc trưng và tính chất của các thuật toán mã hóa dòng, trong các nội dung tiếp theo bài giảng sẽ đi vào trình bày và phân tích một số thuật toán mã hóa dòng đang được áp dụng phổ biến hiện nay.

2.4.2. Thuật toán mã hóa A5/1

Trong mạng điện thoại GSM, để bảo mật dữ liệu trong quá trình truyền tin giữa trạm thu phát sóng vô tuyến và máy điện thoại người sử dụng A5/1 để mã hóa dữ liệu. Đơn vị mã hóa của A5/1 là một bit. Bộ sinh số ngẫu nhiên mỗi lần sẽ sinh ra bit 1 hoặc bit 0 để XOR với thông điệp để cho ra bản mã.

Như đã trình bày trong phần phân loại mật mã trong bày giảng, thuật toán mã hóa A5/1 là thuật toán mã dòng và mang đầy đủ các tính chất, đặc trưng cơ bản của mã dòng. Để hiểu rõ hơn về cơ chế mã hóa và giải mã của thuật toán mã hóa A5/1 trước tiên, bài giảng sẽ trình bày về mô hình thu nhỏ của A5/1 là Tiny A5/1.

a. Tiny A5 / 1

Cơ chế hoạt động của bộ sinh số trong *A5/1*:

Bộ sinh số gồm 3 thanh ghi *X, Y, Z;* thanh ghi *X* gồm 6 bit ký hiệu *(x_0 x_1 x_2… x_5)*; thanh ghi *Y* gồm 8 bit *(y_0 y_1 y_2… y_7);* thanh ghi *Z* gồm 9 bit *(z_0 z_1 z_2… z_8).* Khóa K ban đầu có chiều dài 23 bit và lần lượt phân bố vào các thanh ghi *X, Y, Z.* Các thanh ghi *X, Y, Z* biến đổi theo 3 nguyên tắc sau đây:

- Quay *X* gồm các thao tác
  - $t=x_2 \oplus x_4 \oplus x_5$
  - $t_j=x_{j-1}$ *với* $j= 5, 4, 3, 2, 1$
  - $x_0=t$

Ví dụ: giả sử *X* là 100101, dẫn đến $t = 0 \oplus 0 \oplus 1 = 1$, vậy sau khi quay giá trị của *X* là 110010.

- Quay *Y* gồm các thao tác
  - $t=y_6 \oplus y_7$
  - $y_j=y_{j-1}$ *với* $j= 7, 6, 5, 4, 3, 2, 1$
  - $y_0=t$

Ví dụ: giả sử *Y* là 10010111, dẫn đến $t = 0 \oplus 1 \oplus 1 = 0$, vậy sau khi quay giá trị của *Y* là 01001011.

- Quay *Z* gồm các thao tác
  - $t=z_2 \oplus z_7 \oplus z_8$
  - $z_j=z_{j-1}$ *với* $j= 8, 7, 6, 5, 4, 3, 2, 1$
  - $z_0=t$

Ví dụ: giả sử *Z* là 10010111, dẫn đến $t = 0 \oplus 1 \oplus 1 = 0$, vậy sau khi quay giá trị của *Z* là 010010111.

Cho ba bit *x, y, z*, định nghĩa một hàm *maj(x, y, z)* là hàm "*chiếm đa số*", nghĩa là nếu trong 3 bit *x, y, z* có từ hai bit 0 trở lên thì hàm trả về giá trị 0, nếu không hàm trả về giá trị 1.

Tại bước sinh số thứ *i*, các phép tính sau được thực hiện:

$$m = maj(x_1, y_2, z_3)$$

- Nếu $x_1 = m$ thì *thực hiện* quay *X*
- Nếu $y_2 = m$ thì *thực hiện* quay *Y*
- Nếu $z_3 = m$ thì *thực hiện* quay *Z*

- Bit sinh ra là : $s_i = x_5 \oplus y_7 \oplus z_8$

Bit s_i được *XOR* với bit thứ i trong thông điệp để có được bit thứ *i* trong bản mã theo quy tắc của mã dòng.

Ví dụ: mã hóa thông điệp *P=111* (chữ h) với khóa *K* là :

100101. 01001110.100110000

Ban đầu giá trị của các thanh ghi *X, Y, Z* là:

X = 10010

Y = 01001110

Z = 100110000

- Bước 0: $x_1 = 0, y_3 = 0, z_3 = 1 \rightarrow m = 0 \rightarrow$ *quay X, quay Y*

  X = 110010

  $Y = 10100111 \rightarrow s_0 = 0 \oplus 1 \oplus 0 = 1$

  Z = 100110000

- Bước 1: $x_1 = 1, y_3 = 0, z_3 = 1 \rightarrow m = 1 \rightarrow$ quay *X*, quay *Z*

  X = 111001

  $Y = 10100111 \rightarrow s_1 = 1 \oplus 1 \oplus 0 = 1$

  Z = 010011000

- Bước 2: $x_1 = 1, y_3 = 0, z_3 = 0 \rightarrow m = 0 \rightarrow$ quay *Y*, quay *Z*

  X = 111001

  $Y = 01010011 \rightarrow s_2 = 1 \oplus 1 \oplus 0 = 0$

  Z = 001001100

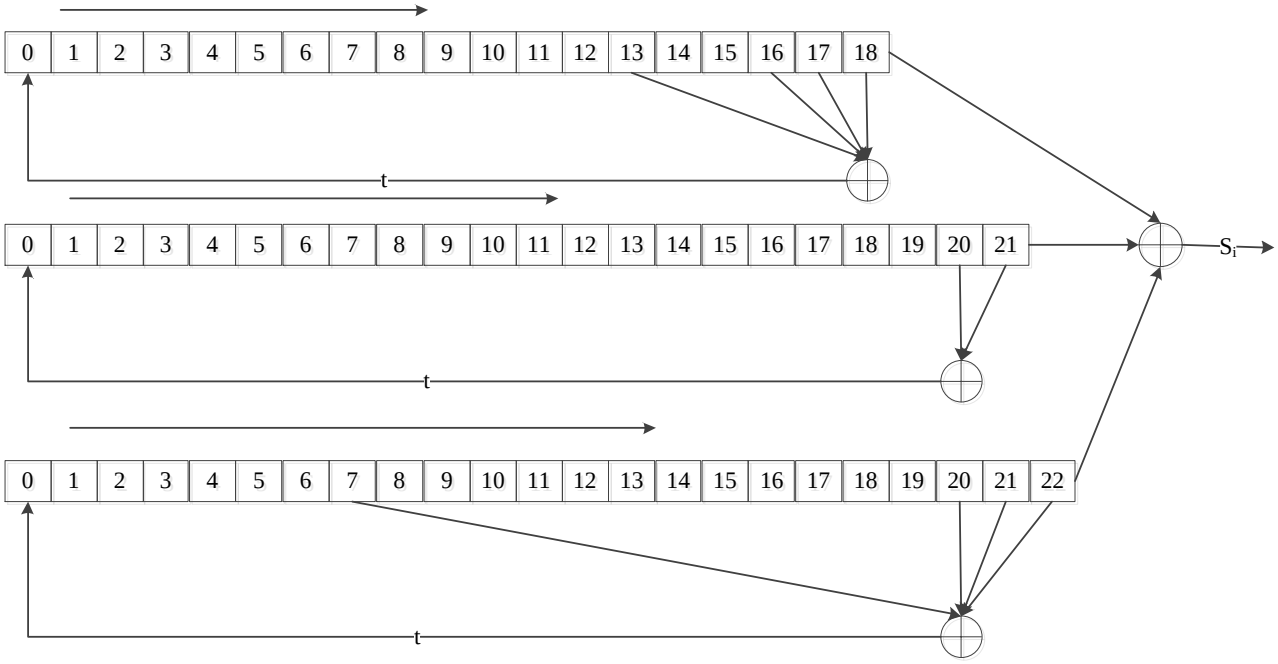
Vậy bản mã là $C = 111 \oplus 100 = 011$ (chữ *D)*

b. A5/1

Về nguyên tắc bộ sinh số *A5/1* hoạt động giống như *TinyA5/1*. Kích thước thanh ghi *X, Y, Z* lần lượt là 19, 22 và 23 bit. Các bước quay *X, Y, Z* cụ thể như sau:

- Quay *X* gồm các thao tác :
  - $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
  - $x_j = x_{j-1}$ *với* $j = 18, 17, 16, 15, \ldots 4, 3, 2, 1$
  - $x_0 = t$
- Quay *Y* gồm các thao tác:
  - $t = y_{20} \oplus y_{21}$
  - $y_j = y_{j-1}$ *với* $j = 21, 20, \ldots 4, 3, 2, 1$
  - $y_0 = t$
- Quay *Z* gồm các thao tác :
  - $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
  - $z_j = z_{j-1}$ *với* $j = 22, 21, \ldots 6, 5, 4, 3, 2, 1$
  - $z_0 = t$

Hàm maj tính được trên 3 bit $x_8; y_{10}; z_{10}$. Sau khi xoay xong bit sinh ra là $s_i = x_8 \oplus y_{10} \oplus z_{10}$. Toàn bộ quá trình sinh dãy số của A5/1 được minh họa qua hình 2.15. Mã hóa *A5/1* có thể được thực hiện dễ dàng bằng các thiết bị phần cứng, tốc độ nhanh. Do đó *A5/1* đã từng được sử dụng để mã hóa các dữ liệu trong thời gian thực, vì vậy ngày nay *A5/1* được sử dụng để mã hóa dữ liệu cuộc gọi trong mạng điện thoại *GSM*.



Hình 2. 17. Sơ đồ mã hóa giải mã A5/1

2.4.3. Thuật toán mã hóa RC4

Cũng giống như thuật toán mã dòng *A5/1*, *RC4* là một phương pháp mã hóa dòng với kích thước khóa không cố định, được phát triển vào năm 1987 bởi Ron Rivest cho *RSA DaSecurit*. Thuật toán *RC4* được dùng trong giao thức SSL để bảo mật dữ liệu trong quá trình truyền dữ liệu giữa Web Server và trình duyệt Web. Ngoài ra *RC4* còn được sử dụng trong mã hóa *WEP* của mạng *Wireless LAN*. Để hiểu rõ hơn về thuật toán mã hóa *RC4*, trước tiên hãy cùng xét một mô hình thu nhỏ của *RC4* gọi là *TinyRC4*:

a. TinyRC4

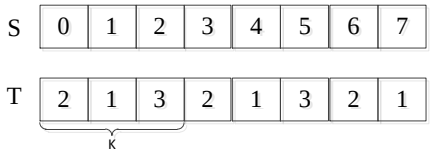
Khác với *TinyA5/1*, đơn vị mã hóa của *TinyRC4* là 3 bit. *TinyRC4* dùng 2 mảng *S* và *T* mỗi mảng gồm 8 số nguyên 3 bit (từ 0 đến 7). Khóa là một dãy gồm *N* số nguyên 3 bit với *N* có thể lấy giá trị từ 1 đến 8. Bộ sinh số mỗi lần sinh ra 3 bit để sử dụng trong phép *XOR*. Quá trình sinh số của *TinyRC4* gồm hai giai đoạn:

- Giai đoạn khởi tạo:

```
/* Khoi tao day so S va T */
        for  i = 0 to 7 do
        S[i] = i;
        T[i] = K[i mod N];
        next i
/* Hoan vi day S */
        j = 0;
        for  i = 0 to 7 do
        j = (j + S[i] + T[i]) mod 8;
        Swap(S[i], S[j]);
       next i
```

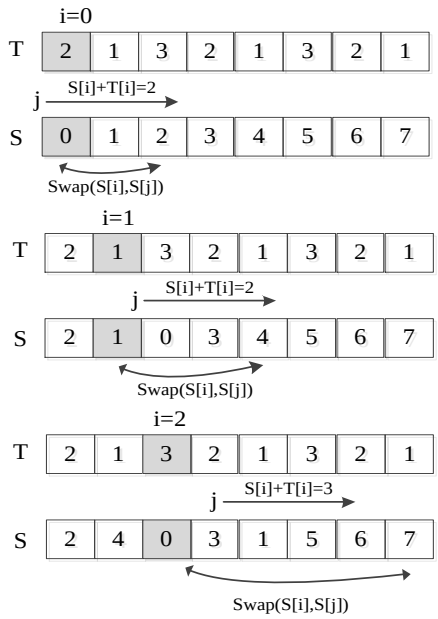
Dãy *S* gồm các số nguyên 3 bit từ 0 đến 7 được sắp xếp thứ tự tăng dần. Sau đó dựa trên các phần tử của khóa *K*, các phần tử của S được hoán vị lẫn nhau đến một mức độ ngẫu nhiên nào đó. Ví dụ: mã hóa thông điệp *P* = 001000110 (từ "bag") với khóa *K* gồm 3 số 2, 1, 3 (*N*=3):

- Khởi tạo S và T



Hình 2. 18. Khởi tạo S và T

- Hoán vị S



Hình 2. 159. Quá trình hoán vị S

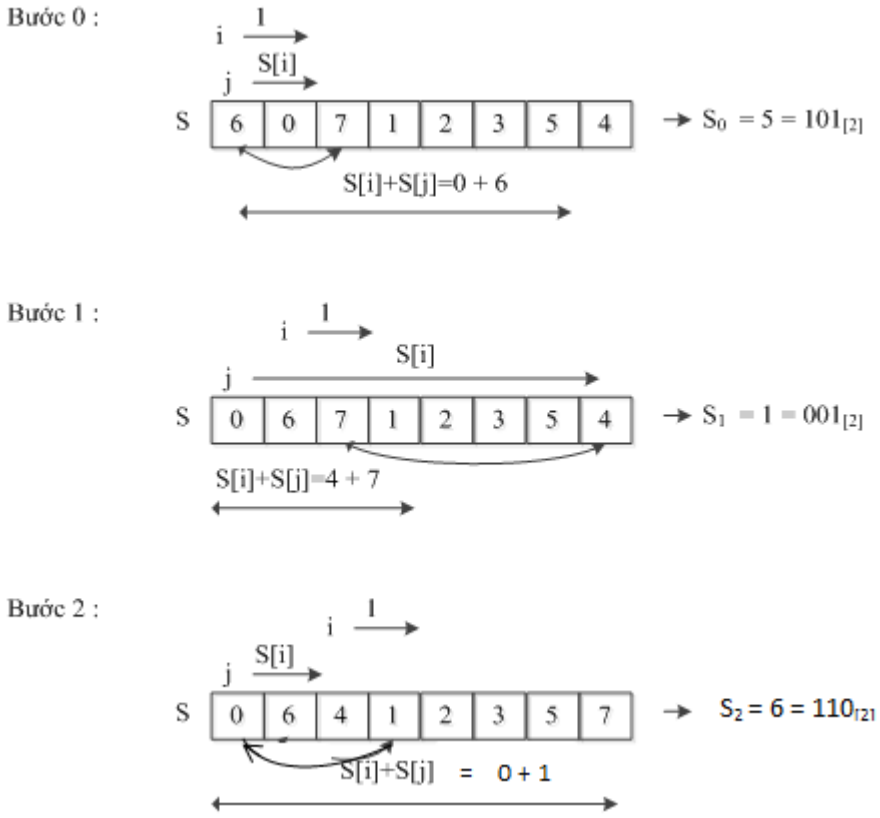
Quá trình thực hiện đến khi i=7 và lúc đó dãy S là 6 0 7 1 2 3 5 4

- Giai đoạn sinh số

```
i, j = 0;
while (true)
    i = (i + 1) mod 8;
    j = (j + S[i]) mod 8;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 8;
    k = S[t];
end while;
```

Trong giai đoạn này, các phần tử của S tiếp tục được hoán vị. Tại mỗi bước sinh số, hai phần tử của dãy *S* được chọn để tính ra số *k*, 3 bit là số được dùng để *XOR* với đơn vị mã hóa của thông điệp.

Tiếp tục ví dụ trên, quá trình sinh số mã hóa thông điệp "*bag*" thực hiện như sau:



Hình 2. 20. Quá trình sinh số mã hóa thông điệp

Vậy bản mã là C = 001.000.110 ⊕ 101.001.110 = 100.001.000 (từ EBA)

b. RC4

Cơ chế hoạt động của *RC4* cũng giống như *TinyRC4* với các đặc tính sau:

- Đơn vị mã hóa của *RC4* là một byte 8 bit.
- Mảng *S* và *T* gồm 256 số nguyên 8 bit.
- Khóa *K* là một dãy gồm *N* số nguyên 8 bit với *N* có thể lấy giá trị từ 1 đến 256.
- Bộ sinh số mỗi lần sinh ra một byte để sử dụng trong phép *XOR*. Hai giai đoạn của *RC4* là:

  - Giai đoạn khởi tạo:

```
/* Khoi tao day S va T*/
for  i = 0 to 255 do
    S[i] = i;
```

```
    T[i] = K[i mod N];
next i
/* Hoan vi day S */
j = 0;
for  i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap(S[i], S[j]);
next i
```

- Giai đoạn sinh số :

```
i, j = 0;
while (true)
    i = (i + 1) mod 256;
    j = (j + S[i]) mod 256;
    Swap (S[i], S[j]);
    t = (S[i] + S[j]) mod 256;
    k = S[t];
      end while;
```

Quá trình sinh số của *RC4* cũng sinh ra dãy số ngẫu nhiên, khó đoán trước, vì vậy *RC4* đạt được mức độ an toàn cao. Mã hóa *RC4* hoàn toàn được thực hiện trên các số nguyên một byte do đó tối ưu cho việc thiết lập bằng phần mềm và tốc độ thực hiện nhanh hơn so với mã khối.

2.5. ƯU ĐIỂM VÀ NHƯỢC ĐIỂM CỦA MÃ HÓA ĐỐI XỨNG

Từ quá trình nghiên cứu về các cơ chế hoạt động của các thuật toán mã hóa này, có thể rút ra một số đánh giá về ưu điểm và nhược điểm của mã hóa đối xứng như sau:

- Ưu điểm:
  - Đơn giản, tức là các yêu cầu về phần cứng không phức tạp;
  - Thời gian tính toán nhanh (tốc độ mã hóa và giải mã nhanh);
  - Có tính hiệu quả, thông thường hệ số mở rộng bản tin bằng 1 (số bit đầu ra mã hóa bằng với số bit đầu vào);

- Dễ sử dụng cho các dịch vụ nhạy cảm với độ trễ và các dịch vụ di động.

- Nhược điểm:
  - Yêu cầu phải dùng kênh an toàn để truyền khóa: chi phí cao và việc thiết lập kênh an toàn là khó khăn;
  - Việc tạo khóa và giữ bí mật khóa rất phức tạp;
  - Số lượng khóa cần tạo ra và lưu trữ lớn;
  - Khó khăn trong việc xây dựng các dịch vụ an toàn khác như các dịch vụ đảm bảo tính toàn vẹn dữ liệu, các dịch vụ xác thực và chữ ký số.

2.6. CÂU HỎI VÀ BÀI TẬP

1) Hãy trình bày đặc điểm chính của mã hóa khóa bí mật? Vẽ sơ đồ minh họa quy trình mã hóa và giải mã của kỹ thuật mã hóa khóa bí mật?

2) Hãy mô tả các bước mã hóa A5/1?

3) Hãy mô tả các bước mã hóa RC4?

4) Hãy mô tả cấu trúc khối Feistel ? Hãy vẽ sơ đồ mã hóa theo nguyên tắc cấu trúc khối Feistel ?

5) Hãy mô tả các bước chính của thuật toán mã hóa DES? Vẽ sơ đồ minh họa về thuật toán mã hóa DES? Giải thích tại sao thuật toán mã hóa DES lại yêu cầu 16 vòng lặp?

6) Hãy trình bày các ưu điểm và nhược điểm của mã hóa DES?

7) Hãy trình bày cơ sở toán học của thuật toán mã hóa AES?

8) Hãy mô tả về các bước chính của thuật toán mã hóa AES ? Hãy vẽ sơ đồ minh họa của thuật toán mã hóa AES?

9) Trong thuật toán DES với kích thước khóa đầu vào là 64 bit. Cho khóa sau khi nén xuống 48 bit ở vòng lặp đầu tiên là $K_1 = 194CD072DE8C_{16}$

Bảng nén khóa từ 56 bit → 48 bit sau khi tiến hành quá trình dịch bit :

14	17	11	24	1	5
3	25	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48

44	49	39	56	34	53
46	42	50	36	29	32

Tính giá trị khóa K sau vòng lặp thứ 2?

10) Tính khóa 48 bit sử dụng cho vòng lặp đầu tiên với dữ liệu đầu vào như sau :

- Khóa 64 bit dưới dạng hexa: **AABB09182736CCDD**.
- Bảng nén khóa từ 64bit → 56 bit :

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

- Bảng nén khóa từ 56 bit → 48 bit sau khi tiến hành quá trình dịch bit :

14	17	11	24	1	5
3	25	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

CHƯƠNG 3: MÃ HÓA KHÓA BẤT ĐỐI XỨNG

Chương 3 trình bày một số kiến thức liên quan đến kỹ thuật mã hóa khóa bất đối xứng bao gồm: khái niệm, nguyên tắc mã hóa, giải mã, ưu điểm và nhược điểm. Bên cạnh đó, chương 3 đề cập một số ứng dụng của mã hóa khóa bất đối xứng trong thực tế.

3.1. GIỚI THIỆU VỀ MÃ HÓA BẤT ĐỐI XỨNG

3.1.1. Giới thiệu chung

Mã hóa đối xứng dù rằng đã phát triển từ cổ điển đến hiện đại nhưng vẫn tồn tại một số điểm yếu sau:

- *Vấn đề trao đổi khóa giữa người gửi và người nhận*: Trong quá trình trao đổi khóa cần phải có một kênh an toàn để trao đổi sao cho khóa phải được giữ bí mật chỉ có người gửi và người nhận biết. Chính vì vậy, điều này gây ra một vấn đề rất khó khăn và cần phải có 1 kênh an toàn. Việc thiết lập một kênh an toàn như vậy sẽ tốn kém về mặt chi phí và chậm trễ về mặt thời gian.
- *Tính bí mật của khóa*: Việc khóa bí mật cần trao đổi giữa người gửi và người nhận sẽ dẫn đến trường hợp: khi có sự cố xảy ra thì không có cơ sở quy trách nhiệm nếu khóa bị tiết lộ.

Để khắc phục điểm yếu của mã hóa đối xứng, cần thực hiện 1 số phương án sau:

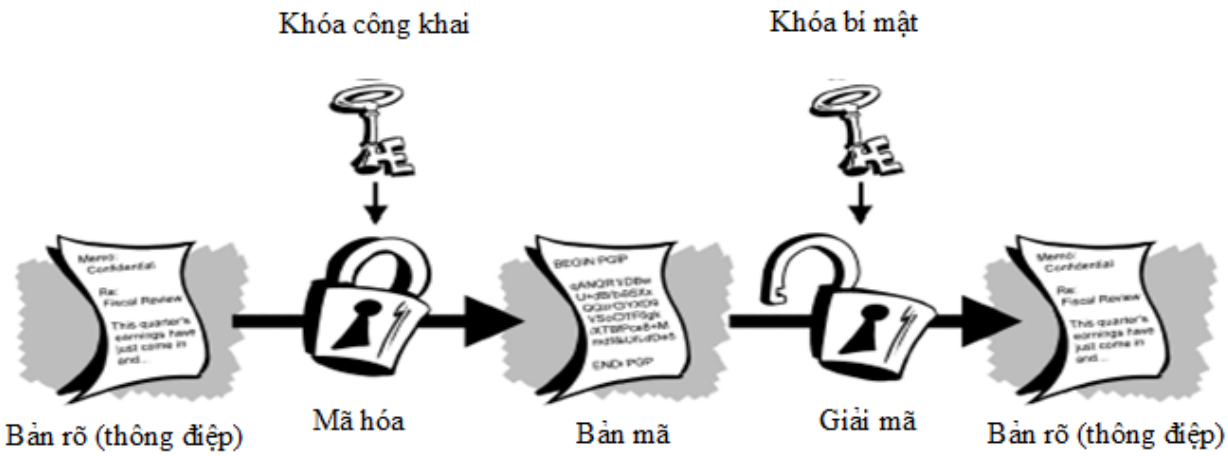
Phương án 1: Người nhận (*Bob*) giữ bí mật khóa K_2, còn khóa K_1 thì công khai cho tất cả mọi người biết. *Alice* muốn gửi dữ liệu cho *Bob* thì dùng khóa K_1 để mã hóa. *Bob* dùng K_2 để giải mã. Ở đây *Trudy* cũng biết khóa K_1, tuy nhiên không thể dùng chính K_1 để giải mã mà phải dùng K_2. Do đó chỉ có duy nhất *Bob* mới có thể giải mã được. Điều này bảo đảm tính bí mật của quá trình truyền dữ liệu. Ưu điểm của phương án này là không cần phải truyền khóa K_1 trên kênh an toàn.

Phương án 2: Người gửi (*Alice*) giữ bí mật khóa K_1, còn khóa K_2 thì công khai cho tất cả mọi người biết. Alice muốn gửi dữ liệu cho *Bob* thì dùng khóa K_1 để mã hóa. *Bob* dùng K_2 để giải mã. Ở đây *Trudy* cũng biết khóa K_2 nên *Trudy* cũng có thể giải mã được.

Do đó phương án này không đảm bảo tính bí mật. Vì vậy nếu kết hợp phương án 1 và phương án 2, thì mô hình đề xuất sẽ khắc phục được các nhược điểm của mã hóa đối xứng.

Vào năm 1976 *Whitfield Diffie* và *Martin Hellman* đã tìm ra một phương pháp mã hóa khác mà có thể giải quyết được hai vấn đề trên, đó là mã hóa khóa công khai hay còn gọi là mã hóa bất đối xứng. Đây có thể xem là một bước đột phá quan trọng nhất trong lĩnh vực mã hóa.

Mã hóa bất đối xứng là các phương pháp mà quy trình mã hóa và giải mã sử dụng các khóa khác nhau. Khóa dùng để mã hóa là khóa công khai và khóa dùng để giải mã là khóa bí mật. Hình 3.1 thể hiện quy trình mã hóa và giải mã của mô hình mã hóa khóa công khai.



Hình 3. 1. Mô hình mã hóa bất đối xứng

Trong cả hai phương án, một khóa được giữ bí mật và chỉ một người biết, còn khóa kia được công khai. Để thuận tiện, trong bài giảng quy ước lại các ký hiệu như sau:

- Khóa bí mật được gọi là khóa riêng và ký hiệu là K_R.
- Khóa công khai được ký hiệu là K_U.
- Thông điệp được ký hiệu là *M*, còn bản mã giữ nguyên ký hiệu là *C*
- Phương án 1 viết lại theo công thức 3.1 và 3.2 :

$$C = E(M, K_U) \qquad (3.1)$$

$$M = D(C, K_R) \qquad (3.2)$$

- Phương án 2 viết lại theo công thức 3.3 và 3.4:

$$C = E(M, K_R) \quad (3.3)$$

$$M = D(C, K_U) \quad (3.4)$$

Vậy giữa khóa riêng và khóa công khai có mối quan hệ với nhau như thế nào?. Chúng độc lập hay phụ thuộc vào nhau?. Trong thuật toán mã hóa khóa bất đối xứng thì dĩ nhiên là hai khóa K_U *và* K_R không thể nào hoàn toàn độc lập với nhau. Phải có một mối quan hệ nào đó giữa K_U và K_R thì mới có thể tiến hành mã hóa và giải mã được. Có nghĩa là $K_R = f(K_U)$. Tuy nhiên một yêu cầu rất quan trọng là việc tính $K_R = f(K_U)$ phải là bất khả thi về mặt thời gian. Nếu nguyên tắc này bị vi phạm thì việc giữ bí mật khóa K_R không còn ý nghĩa vì từ khóa công khai K_U có thể tính được khóa riêng K_R.

Để có được cặp khóa K_R và K_U như trên, người ta thường dùng các hàm một chiều (*oneway function*). Các hàm một chiều có tính chất là hàm phi nghịch đảo vì vậy rất khó thực hiện. Sau đây là ví dụ về hàm một chiều: việc sinh ra hai số nguyên tố lớn *p, q* và tính tích $N = pq$ thì thực hiện dễ dàng. Tuy nhiên nếu chỉ cho trước N và thực hiện phân tích N để tìm lại hai số nguyên tố *p, q* là việc hoàn toàn bất khả thi về mặt thời gian. Đây chính là cơ sở để xây dựng thuật toán mã hóa bất đối xứng *RSA*. Trong mục 3.2 sẽ trình bày chi tiết về vấn đề này.

3.1.2. Một số thuật toán mã hóa bất đối xứng

Các nghiên cứu trên thế giới từ năm 1976 cho đến nay đã đưa ra được một số bài toán một chiều như sau:

- Bài toán logarit rời rạc (Hệ mật Omura-Massey, Elgama…);
- Bài toán phân tích thừa số (gắn với hệ mật nổi tiếng RSA);
- Bài toán xếp ba lô (Hệ mật Merkle – Hellman; Hệ mật Chor-Rivest);
- Bài toán mã sửa sai (Hệ mật McEliece…);
- Bài toán xây dựng hệ mật trên đường cong Elliptic.

Trong bài giảng này, chỉ tập trung vào tìm hiểu thuật toán *RSA*. Các thuật toán khác có thể đọc trong phần tài liệu tham khảo.

3.2. THUẬT TOÁN MÃ HÓA RSA

Thuật toán mã hóa *RSA* là một thuật toán mã hóa khóa công khai. Thuật toán *RSA* được xây dựng bởi các tác giả *Ron Rivest, Adi Shamir* và *Len Adleman* tại học viện *MIT* vào năm 1977, và ngày nay đang được sử dụng rộng rãi. Về mặt tổng quát thuật toán *RSA* là một phương pháp mã hóa theo khối. Trong đó thông điệp *M* và bản mã *C* là các số nguyên từ 0 đến 2^i với *i* số bit của khối. Kích thước thường dùng của *i* là 1024 bit. Thuật toán *RSA* sử dụng hàm một chiều là vấn đề phân tích một số thành thừa số nguyên tố và bài toán Logarith rời rạc.

Nguyên tắc thực hiện quá trình mã hóa và giải mã thuật toán RSA :

✓ ***Quá trình mã hóa:***

Để thực hiện mã hóa và giải mã, thuật toán *RSA* dùng phép lũy thừa modulo của lý thuyết số. Các bước thực hiện như sau:

- Chọn hai số nguyên tố lớn *p* và *q* và tính $N = pq$. Cần chọn p và q sao cho:

  $M < 2^{i-1} < N < 2^i$ với *i* là chiều dài bản rõ.
- Tính $\Phi(n) = (p - 1)(q - 1)$
- Tìm một số *e* sao cho: {*e* và $\Phi(n)$ *là 2 số nguyên tố cùng nhau* và $0 < e < \Phi(n)$}
- Tìm một số *d* sao cho: $e.d \equiv 1 \bmod \Phi(n)$ (hay: $d = e^{-1} \bmod \Phi(n)$)

  (*d* là nghịch đảo của e trong phép modulo Φ(n))
- Chọn khóa công khai K_U là cặp (e, N), khóa riêng K_R là cặp (d, N)
- Việc mã hóa thực hiện theo công thức:

  - Theo phương án 1, mã hóa: $C = E(M, K_U) = M^e \bmod N$ *(3.5)*
  - Theo phương án 2, mã hóa chứng thực: $C = E(M, K_R) = M^d \bmod N$ *(3.6)*

✓ **Quá trình giải mã:** Việc giải mã thực hiện theo công thức:

- Theo phương án 1, giải mã: $M = D(C,K_R) = C^d \bmod N$ *(3.7)*
- Theo phương án 2, mã hóa chứng thực: $M = D(C,K_U) = C^e \bmod N$ *(3.8)*

Thông điệp *M* có kích thước *i-1* bit, bản mã *C* có kích thước *i* bit.

Tính đúng của thuật toán RSA:

Trên đây, bài giảng đã mô tả quy trình mã hóa và giải mã thuật toán RSA. Tiếp theo bài giảng sẽ trình bày tính đúng đắn của thuật toán RSA. Giả sử bản mã chính là thông điệp $\overline{M} = M$. Xét phương án 1:

- Từ bước 4 suy ra được $e.d = k.n + 1$ với k là một số nào đó.
- *Vậy* $\overline{M} = C^d \bmod N$

$= M^{ed} \bmod N$

$= M^{kn+1} \bmod N$

$= M^{k(p-1)(q-1)+1} \bmod N$

Điều này có nghĩa là *RSA* đúng khi và chỉ khi $M^{k(p-1)(q-1)+1} \bmod N \equiv M \bmod p$. Xét hai khả năng của M:

- M chia hết cho p khi đó $M \bmod p = 0$ do đó $M^{k(p-1)(q-1)+1} \equiv M \equiv 0 \bmod p$
- M không chia hết cho p: Vì p là số nguyên tố nên M và p là hai số nguyên tố cùng nhau. Vậy : $M^{k(p-1)(q-1)+1} \bmod p = M.(M^{p-1})^{(q-1)k} \bmod p = M.1^{k(q-1)} \bmod p$ *(theo định lý Fermat)* $= M \bmod p$

Vậy : $M^{k(p-1)(q-1)+1} - M \equiv 0 \bmod p \ \forall M$. Hay nói cách khác $M^{k(p-1)(q-1)+1} - M$ chia hết cho q. Vì q, p là hai số nguyên tố nên suy ra được $M^{k(p-1)(q-1)+1} - M$ chia hết cho $N = p.q$. Như vậy:

$M^{k(p-1)(q-1)+1} \equiv M \bmod N$

$\Rightarrow \quad \overline{M} = M^{k(p-1)(q-1)+1} \bmod N = M$ *(do* $M < N$ *)*

Vì e và d đối xứng nên có thể thấy trong phương án 2, cũng có $\overline{M} = M$.

- Không thể suy ra K_R từ K_U, nghĩa là tìm cặp (d, N) từ cặp (e, N): Có e và N, muốn tìm d, phải dựa vào công thức: $e.d \equiv 1 \bmod n$. Do đó phải tính được n.

Vì $\Phi(n) = (p-1)(q-1)$ nên suy ra phải tính được p và q. Vì $N = pq$ nên chỉ có thể tính được p và q từ N. Tuy nhiên điều này là bất khả thi vì $N = pq$ là hàm một chiều. Vậy không thể tính được K_R từ K_U

Ví dụ RSA :

- Chọn $p = 11$, $q = 3$ do đó $N = p.q = 33 = 100001_2$
- $\Phi(n) = (q-1)(p-1) = 20$
- Chọn $e = 3$ nguyên tố cùng nhau với $\Phi(n) = 20$.
- Tính nghịch đảo của e trong phép modulo n được $d = 7$ $(3.7 = 21)$
- Khóa công khai $K_U = (e,N) = (3,33)$.
- Khóa bí mật $K_R = (d,N) = (7,33)$.

Với mã hóa bảo mật:

- Mã hóa thông điệp $M = 17$

  $C = M^e \bmod N = 17^3 \bmod 33 = 29$

- Giải mã bản mã $C = 29$

  $\overline{M} = C^d \bmod N = 29^7 \bmod 33 = 17 = M$

Với mã hóa chứng thực:

- Mã hóa thông điệp $M = 17$

  $C = M^d \bmod N = 17^7 \bmod 33 = 8$

- Giải mã bản mã $C = 8$

  $\overline{M} = C^d \bmod N = 8^3 \bmod 33 = 17 = M$

Độ phức tạp trong tính toán RSA:

Có hai vấn đề về độ phức tạp tính toán trong phương pháp mã hóa *RSA*. Đó là các phép tính sinh khóa và các phép tính mã hóa/giải mã.

- *Phép tính mã hóa và giải mã:* Phép tính mã hóa và giải mã dùng phép lũy thừa modulo. Để an toàn, thường phải chọn *N, e, d, M* lớn. Tuy nhiên số học modulo có một tính chất sau:

$$(a \times b) \equiv \left[(a \bmod n) \times (b \bmod n)\right] \bmod n$$

Có thể sử dụng tính chất này để đơn giản phép tính lũy thừa modulo thông qua một phương pháp gọi là "bình phương liên tiếp".

Ví dụ cần tính $x^{16} \bmod n$ thì cần thực hiện các bước sau:

đầu tiên tính $a = x \bmod n$,

tiếp theo tính $b = x^2 \bmod n = a^2 \bmod n$,

tiếp theo tính $c = x^4 \bmod n = b^2 \bmod n$,

tiếp theo tính $d = x^8 \bmod n = c^2 \bmod n$,

cuối cùng có $x^{12} \bmod n = d^2 \bmod n$. Các số a, b, c, d luôn nhỏ hơn n do đó tránh được việc tính số lũy thừa lớn đồng thời nâng cao tốc độ tính toán.

Trong trường hợp tổng quát để tính $x^b \bmod n$, viết b dưới dạng số nhị phân:

$$b = b_k b_{k-1} \ldots b_2 b_1 b_0 \text{ trong đó bi là các bit 0, 1.}$$

Như vậy $b = \sum_{b_i \neq 0} 2^i$. Do đó $x^b = \prod_{b_i \neq 0} x^{2^i} \bmod n$

Như vậy ứng với các $b_i = 1$, dùng phương pháp bình phương liên tiếp để tính các x^{2^i}. Sau đó nhân các kết quả với nhau và rút gọn modulo để có kết quả cuối cùng. Để tăng tốc độ quá trình mã hóa dữ liệu, người ta thường chọn một khóa công khai e cụ thể nào đó. Thường thì e là $65537 (2^{16} + 1)$.

- *Phép tính sinh khóa:*

Phép tính sinh khóa là chọn p và q nguyên tố để tính N. Để phân tích số N thành tích hai thừa số nguyên tố p, q, chỉ có một cách duy nhất là thử từng số p và q. Do đó phải chọn p, q đủ lớn để việc thử là không khả thi. Dựa vào lý thuyết số nguyên tố, người ta ước tính rằng cần thử trung bình khoảng 70 số lẻ để tìm ra một số nguyên tố lớn chừng 2^{200}. Vì đã chọn e trước là 65537 (hay 3 hoặc 17 …), do đó cần kiểm tra xem e có nguyên tố cùng nhau với $\Phi(n) = (p-1)(q-1)$ hay không. Nếu không phải thử lại với p và q khác. Sau khi đã tìm p và q thích hợp, cần tìm d sao cho $e.d \equiv 1 \bmod \Phi(n)$.

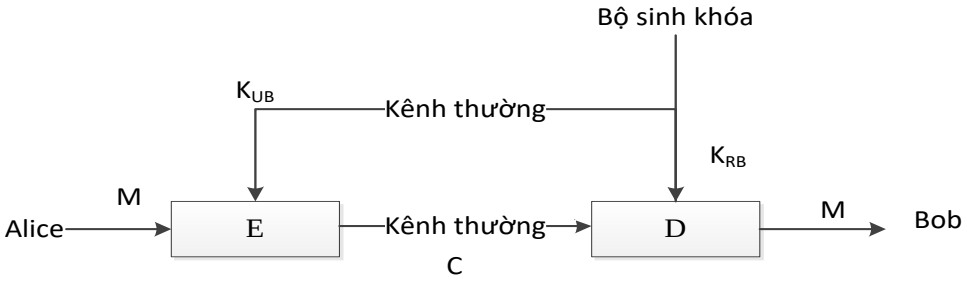
3.3. ƯU ĐIỂM, NHƯỢC ĐIỂM CỦA MÃ HÓA BẤT ĐỐI XỨNG

Xét tính Chứng thực trong mã hóa bất đối xứng: Giả sử *Alice* muốn gửi dữ liệu cho *Bob* dùng mã hóa khóa công khai, trước tiên *Alice* và *Bob* sẽ chọn cặp khóa riêng và khóa công khai. Ký hiệu khóa riêng và khóa công khai của *Alice* là K_{RA} và K_{UA}, của *Bob* là K_{RB}

và K_{UB}. Như vậy để gửi dữ liệu cho *Bob,* Alice sẽ dùng phương án 1: mã hóa dữ liệu bằng khóa công khai K_{UB} của *Bob,* và *Bob* dùng khóa riêng K_{RB} để giải mã :

$$C = E(M, K_{UB}) \qquad (3.9)$$

$$M = D(C, K_{RB})$$

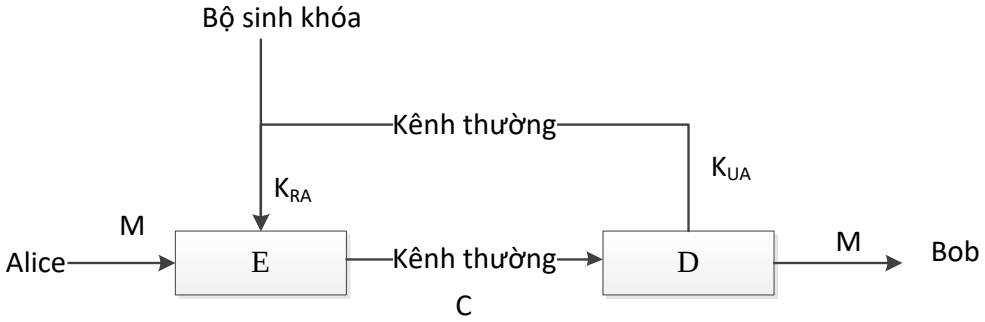


Hình 3. 2. Mô hình bảo mật

Để đảm bảo tính chứng thực và *Alice* không chối bỏ trách nhiệm gửi dữ liệu, *Alice* sẽ dùng phương án 2: *Alice* mã hóa dữ liệu bằng khóa riêng K_{RA}, và *Bob* dùng khóa công khai K_{UA} của *Alice* để giải mã.

$$C = E(M, K_{RA})$$

$$M = D(C, K_{UA}) \qquad (3.10)$$



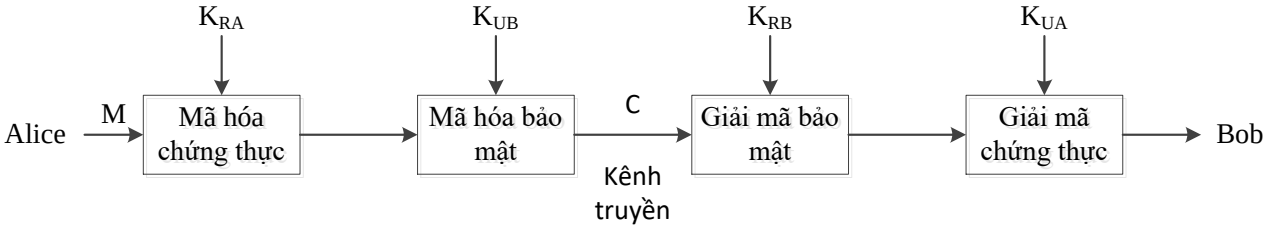
Hình 3. 3. Mô hình xác thực

Cũng với giả định rằng thông điệp có ý nghĩa là một dãy bit có cấu trúc, nếu bản giải mã của *Bob* là hợp lệ thì rõ ràng là *Alice* là người gửi vì chỉ có *Alice* mới có khóa riêng K_{RA}. Giống như mã hóa đối xứng, nếu Trudy (kẻ tấn công) can thiệp chỉnh sửa trên bản mã *C* thì *Bob* sẽ giải mã ra thông điệp là một dãy bit vô nghĩa. Còn nếu *Trudy* có được khóa K_{RA} thì *Alice* không thể thoái thác trách nhiệm làm lộ khóa.

Tuy nhiên mô hình như trên lại không đảm bảo tính bí mật. Vì không chỉ riêng *Bob, Trudy* cũng biết được khóa công khai K_{UA} của *Alice*. Do đó *Trudy* có thể giải mã bản mã *C* và biết được nội dung thông điệp *M*. Để giải quyết vấn đề trên, người kết hợp tính bí mật, chứng thực và không từ chối qua hình 3.4.

$$C = E(E(M, K_{RA}), K_{UB}) \qquad (3.11)$$

$$M = D(D(C, K_{RB}), K_{UA})$$



Hình 3. 4. Mô hình chứng thực và không chối bỏ

Từ những phân tích trên, cũng như kết hợp với đặc điểm của thuật toán mã hóa khóa bất đối xứng, có thể thấy một số ưu điểm và nhược điểm của mã hóa bất đối xứng như sau:

- Ưu điểm:
  - Kích thước khóa lớn, độ an toàn cao;
  - Việc quản lý và phân phối khóa dễ dàng hơn so với mã hóa đối xứng.
- Nhược điểm:
  - Tốc độ mã hóa, giải mã chậm hơn so với mã hóa đối xứng.

3.4. ỨNG DỤNG CỦA MÃ HÓA BẤT ĐỐI XỨNG

Ứng dụng cơ bản của các thuật toán mã hóa bất đối xứng nói chung bao gồm:

- Bí mật trong truyền tin (Confidentiality);
- Chứng thực;
- Kết hợp tính bí mật và tin cậy.

Ứng dụng thực tế của các thuật toán mã hóa bất đối xứng:

- Dùng để vận chuyển và bảo mật khóa bí mật cho mã hóa đối xứng;
- Mã hóa email hoặc xác thực người gửi email (OpenPGP hay S/MIME);
- Mã hóa hoặc nhận thực văn bản (Các tiêu chuẩn chữ ký XML* hoặc mã hóa XML* khi văn bản được thể hiện dưới dạng XML);

- Xác thực người dùng ứng dụng (Đăng nhập bằng thẻ thông minh; nhận thực người dùng trong SSL);
- Ứng dụng trong chữ ký số, chứng chỉ số.

3.5. CÂU HỎI VÀ BÀI TẬP

1) Hãy trình bày đặc điểm chính của thuật toán mã hóa khóa công khai?. Vẽ sơ đồ minh họa quy trình mã hóa và giải mã của thuật toán mã hóa khóa công khai?.
2) Hãy trình bày về chứng thực trong mô hình mã hóa bất đối xứng?.
3) Hãy nêu các yêu cầu cơ bản trong thủ tục sinh khóa trong thuật toán mã hóa RSA?. Hãy phân tích các yêu cầu cơ bản đó?.
4) Hãy trình bày về quá trình mã hóa và giải mã của thuật toán RSA?. Hãy lấy ví dụ minh họa về quá trình mã hóa và giải mã của thuật toán RSA?.
5) Hãy phân tích mức độ an toàn của thuật toán RSA?.
6) Hãy trình bày về ưu nhược điểm của mã hóa bất đối xứng?.
7) Hãy nêu một vài ứng dụng của mã hóa bất đối xứng?.

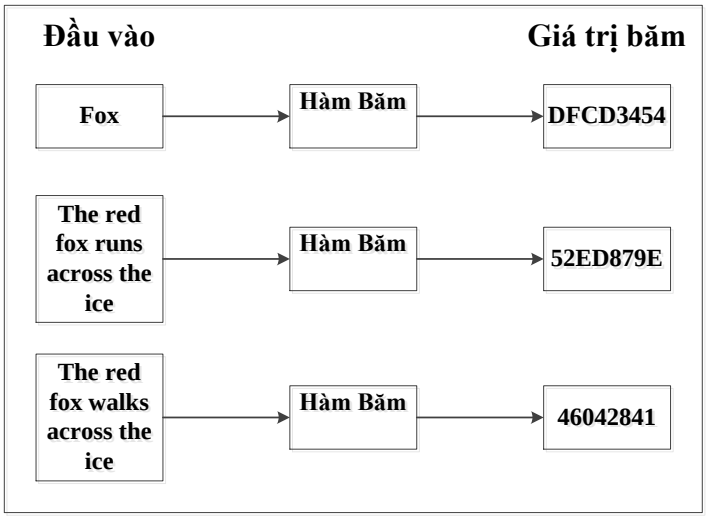
CHƯƠNG 4: CÁC HÀM BĂM

Chương này cung cấp các kiến thức liên quan đến hàm băm bao gồm: định nghĩa, khái niệm, tính chất, vai trò của hàm băm. Ngoài ra, chương 4 còn trình bày về một số hàm băm phổ biến hiện nay cũng như nguyên tắc làm việc của một số hàm băm đó.

4.1. GIỚI THIỆU VỀ HÀM BĂM

4.1.1. Định nghĩa

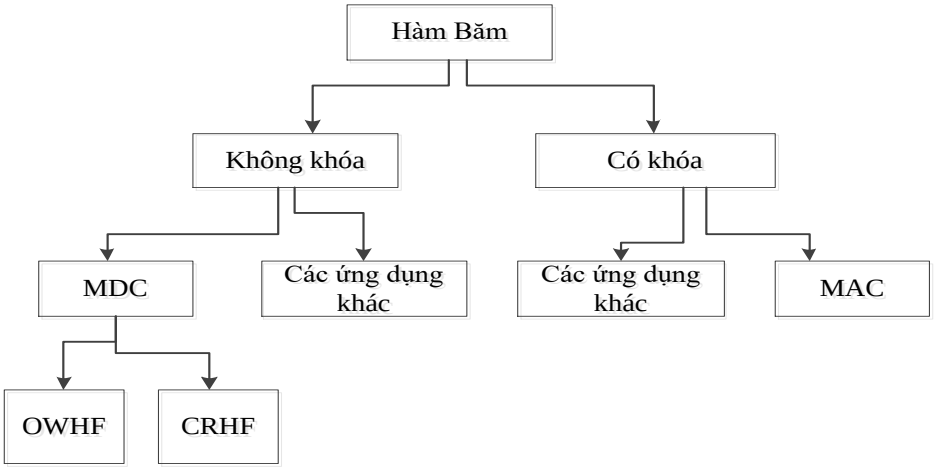
Hàm băm là hàm chuyển đổi một thông điệp (chuỗi bit) có độ dài bất kỳ hữu hạn thành một dãy bit có độ dài cố định *n* bit (*n > 0*). Dãy bit đầu ra của hàm băm được gọi là giá trị băm hay mã băm. Hình 4.1 dưới đây trình bày về ví dụ của hàm băm.



Hình 4. 1. Ví dụ về hàm băm

4.1.2. Phân loại

Cũng giống như mã hóa, việc phân loại hàm băm có nhiều cách khác nhau. Cơ chế của việc phân loại này dựa trên các tính chất hoặc theo khóa. Hình 4.2 trình bày một cách để phân loại hàm băm.



Hình 4. 2. Phân loại hàm băm

Theo đó, một số cách để phân loại hàm băm như sau:

Cách 1. Theo khóa sử dụng: hàm băm được chia thành 2 loại:

- Hàm băm không khóa (unkeyed): đầu vào chỉ là thông điệp *x* với hàm băm *h,*
- Hàm băm có khóa (keyed): đầu vào gồm thông điệp và khóa bí mật (theo dạng *h*(*x*, *K*), với hàm băm *h* và thông điệp *x* và *K* là khóa bí mật).

Cách 2. Theo chức năng: có thể chia các hàm băm thành 2 loại chính:

- Mã phát hiện sửa đổi (MDC - Modification Detection Code): MDC thường được sử dụng để tạo chuỗi đại diện cho thông điệp và dùng kết hợp với các kỹ thuật khác như chữ ký số để đảm bảo tính toàn vẹn của thông điệp. MDC thuộc loại hàm băm không khóa. MDC gồm 2 loại nhỏ:
    - Hàm băm một chiều (OWHF - One-way hash functions): Với hàm băm một chiều, việc tính giá trị băm là dễ dàng, nhưng việc khôi phục thông điệp từ giá trị băm là rất khó khăn;
    - Hàm băm chống đụng độ (CRHF - Collision resistant hash functions): Với hàm băm chống đụng độ, sẽ là rất khó để tìm được 2 thông điệp khác nhau nhưng có cùng giá trị băm.
- Mã xác thực thông điệp (MAC - Message Authentication Code): MAC cũng được dùng để đảm bảo tính toàn vẹn của thông điệp mà không cần một kỹ thuật bổ sung nào khác. MAC là loại hàm băm có khóa.

4.1.3. Các tính chất cơ bản

Như đã trình bày ở trên, hàm băm được phân thành 2 loại là hàm băm có khóa và hàm băm không có khóa. Như vậy, khi xét đến tính chất của hàm băm cũng cần phân loại giữa hàm băm có khóa và hàm băm không khóa. Dưới đây sẽ trình bày về các tính chất cơ bản của hàm băm có khóa và hàm băm không khóa

4.1.3.1. Tính chất của hàm băm không khóa

Ngoài hai tính chất cơ bản của hàm băm đã trình bày ở trên, hàm băm không khóa còn có các tính chất sau:

- ***Tính khó tính toán nghịch ảnh***: Với bất kỳ giá trị băm *h,* không thể tính được *x* sao cho *H(x)=h*. Hay *H* được gọi là ***hàm một chiều.***
- **Tính bền xung đột yếu** (*weak collision resistance*): với bất kỳ giá trị *x*, không thể tính được $y \neq x$ sao cho *H(y) = H(x).*
- **Tính bền xung đột mạnh** (*strong collision resistance*): Không thể tính được một cặp *(x, y)* sao cho *H(x) = H(y).*
- ***Kháng tiền ảnh (Pre-image resistance)*:** Với một mã băm *h* bất kỳ, khó tìm được một thông điệp *m* nào mà *h=hash(m)*. Trong góc độ hàm số toán học, mã băm là ảnh còn thông điệp là tạo ảnh của mã băm, hay gọi là tiền ảnh. Sức kháng cự tấn công từ ảnh ngược về tiền ảnh gọi là kháng tiền ảnh. Một hàm băm có kháng tiền ảnh yếu là lỗ hổng cho các cuộc tấn công tiền ảnh.
- ***Kháng tiền ảnh thứ hai (Second pre-image resistance)*** Với một thông điệp *m1* bất kỳ, khó tìm được một thông điệp thứ hai *m2* sao cho $m1 \neq m2$ và *hash(m1) = hash(m2)*. Xác suất xảy ra biến cố có thông điệp *m2* như thế tương tự biến cố "Cùng ngày sinh như bạn". Một hàm băm có kháng tiền ảnh thứ hai yếu là lỗ hổng cho các cuộc tấn công tiền ảnh thứ hai.

Lưu ý: trên đây bài giảng đã trình bày các tính chất cơ bản của hàm băm không khóa. Tuy nhiên, cần chú ý rằng: trong hàm băm không khóa tồn tại Hàm băm một chiều và Hàm băm chống đụng độ. Trong bài giảng đã tổng hợp các tính chất của cả 2 loại hàm băm này. Chi tiết hơn về các tính chất của hàm băm một chiều và hàm băm chống đụng độ có thể tham khảo trong tài liệu tham khảo của bài giảng.

4.1.3.2. *Tính chất của hàm băm có khóa*

Thuật toán MAC là một họ các hàm H_k (được tham số hoá bằng một khoá bí mật k) có các tính chất sau:

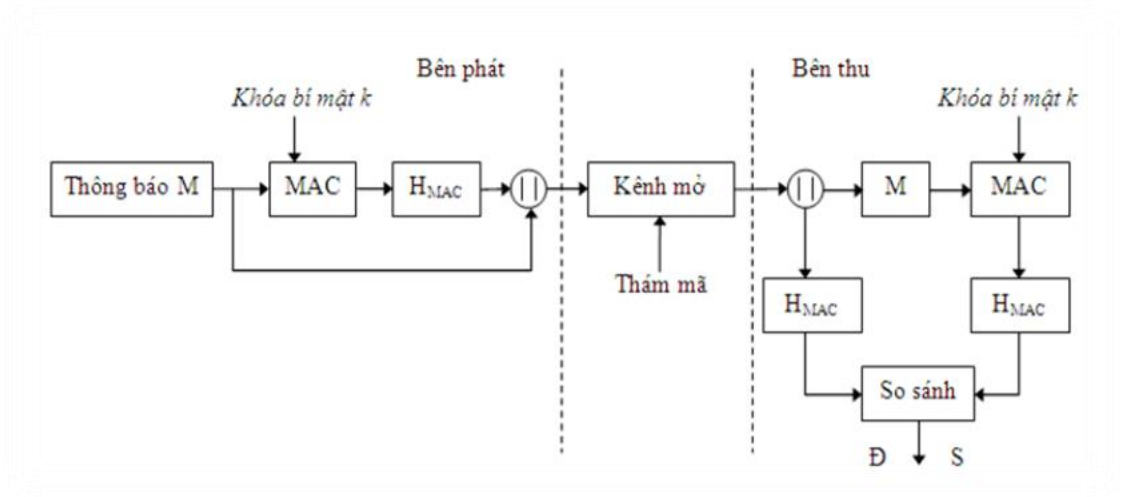
- *Tính chất nén*: với H_k đã biết và giá trị k cho trước và một đầu vào x, thì dễ dàng tính toán được $H_k(x)$ ($H_k(x)$ được gọi là giá trị MAC).
- *Tính chất dễ dàng tính toán*: H_k ánh xạ một đầu vào x có độ dài bit hữu hạn thì dễ dáng tính được đầu ra $H_k(x)$ có độ dài bit n cố định
- *Tính khó tính toán*: Với các cặp giá trị đầu vào là x và x_i với $x \neq x_i$ thì không có khả năng tìm được cặp $H_k(x)$ và $H_k(x_i)$ thỏa mãn $H_k(x) = H_k(x_i)$. Nếu tính chất này không được thỏa mãn thì thông điệp bị coi là giả mạo.

4.1.4. Vai trò và ứng dụng của hàm băm

4.1.4.1. Vai tro

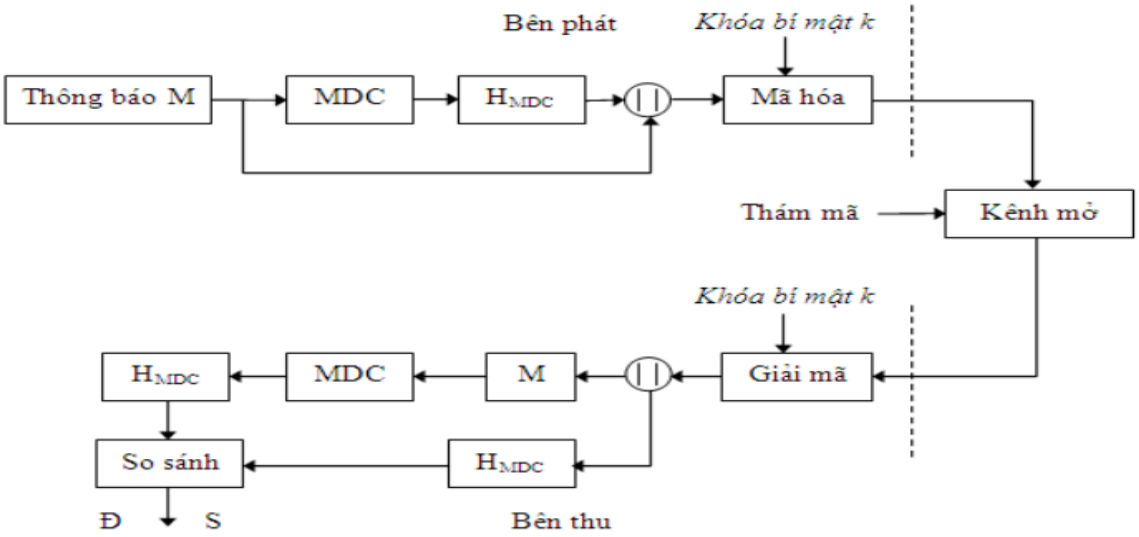
Vai trò và ứng dụng của hàm băm trong thực tế rất lớn. Ngoài việc ứng dụng của hàm băm nhằm đảm bảo các thuộc tính của an toàn thông tin thì hàm băm còn có vai trò là xác định tính toàn vẹn của dữ liệu và xác thực thông điệp. Tính toàn vẹn của dữ liệu và xác thực thông điệp là tính chất đảm bảo dữ liệu không bị sửa đổi một cách bất hợp pháp kể từ khi dữ liệu được tạo ra, được trao đổi hoặc được lưu giữ bởi một nguồn được xác định. Có ba phương pháp cung cấp tính toàn vẹn của dữ liệu bằng cách dùng các hàm băm.

- Chỉ dùng *MAC*



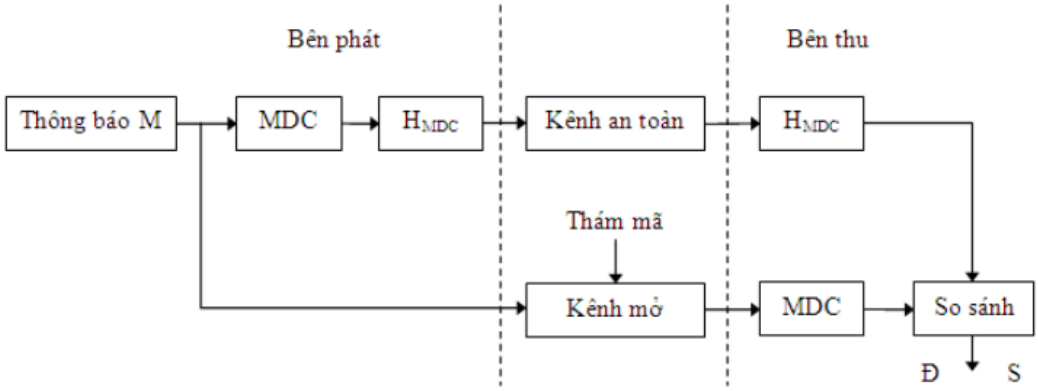
Hình 4.3. Sơ đồ kiểm tra toàn vẹn chỉ dùng *MAC*

- Dùng *MDC* và mã hóa



Hình 4. 4. Sơ đồ kiểm tra toàn vẹn dùng *MDC* và mã hóa

- Sử dụng *MDC* và kênh an toàn



Hình 4. 5. Sơ đồ kiểm tra tính toàn vẹn dùng *MDC* và kênh an toàn

4.1.4.2. Một số ứng dụng của hàm băm

Hàm băm có rất nhiều ứng dụng trong thực tế hiện nay. Một số ứng dụng điển hình của hàm băm bao gồm:

- Ứng dụng trong chữ ký số;

- Ứng dụng trong xác thực thông điệp;
- Trong dụng trong phát hiện sửa đổi trái phép.
- Ứng dụng trong bằng chứng công việc (Proof-of-Work) trong Blockchain....

4.2. MỘT SỐ HÀM BĂM THÔNG DỤNG

Bảng 4.1 liệt kê một số hàm băm thông dụng. Để tìm hiểu các hàm băm này, có thể nghiên cứu các tài liệu tham khảo. Trong bài giảng này, chỉ tập trung vào phân tích 2 hàm băm chính là hàm băm họ MD và SHA.

Bảng 4. 1. Các hàm băm thông dụng

Thuật toán	Kích thước đầu ra	Kích thước trạng thái trong	Kích thước khối	Độ dài	Kích thước world	Xung đột
HAVAL	256/224/192/160/128	256	1024	64	32	Có
MD2	128	384	128	Không	8	Khả năng lớn
MD4	128	128	512	64	32	Có
MD5	128	128	512	64	32	Có
PANAMA	256	8736	256	No	32	Có lỗi
RIPEMD	128	128	512	64	32	Có
RIPEMD-128/256	128/256	128/256	512	64	32	Không
RIPEMD-160/320	160/320	160/320	512	64	32	Không
SHA-0	160	160	512	64	32	Không
SHA-1	160	160	512	64	32	Có lỗi
SHA-256/224	256/224	256	512	64	32	Không
SHA-512/384	512/384	512	1024	128	64	Không
Tiger(2)-192/160/128	192/160/128	192	512	64	64	Không

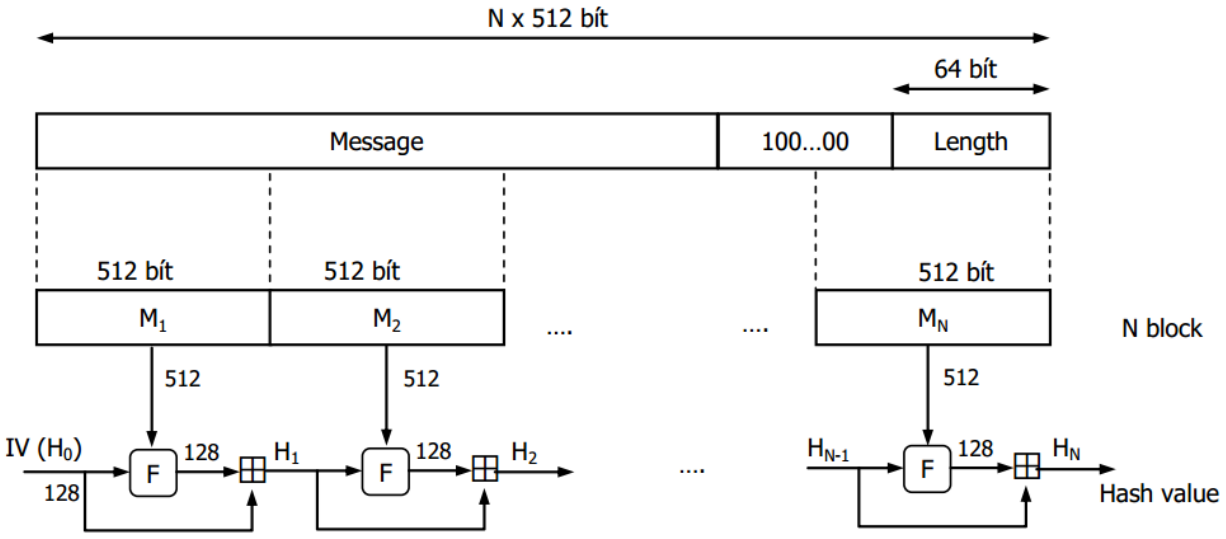
VEST-4/8 (hash mode)	160/256	256/384	8	80/128	1	Không
VEST-16/32 (hash mode)	320/512	512/768	8	160/256	1	Không
WHIRLPOOL	512	512	512	256	8	Không

4.2.1. Hàm băm họ MD - Message Digest (MD5)

MD5 được phát minh bởi *Ron Rivest*, người đã tham gia xây dựng *RSA*. *MD5*, viết tắt từ chữ *Message Digest*, được phát triển lên từ *MD4* và trước đó là *MD2*, do *MD2* và *MD4* không còn an toàn. Kích thước giá trị băm của *MD5* là 128 bit. Trong phần này sẽ trình bày về hàm băm *MD5* với kích thước giá trị băm là 128 bit, được sử dụng để tính giá trị của thông điệp có kích thước tối đa 2^{64} bit.

Thuật toán *MD5*:

- Input: xâu đầu vào x có độ dài tối đa 2^{64} bit.
- Output: chuỗi băm *128* bit.
- Sơ đồ thuật toán:



Hình 4. 6. Sơ đồ hàm băm *MD5*

Quy trình băm của hàm băm *MD5* theo hình 4.6 được trình bày như sau:

- Trước tiên thông điệp được đệm vào dãy padding 100...00. Chiều dài của dãy padding được chọn sao cho thông điệp cuối cùng có thể chia làm N block 512 bit M_1, M_2,..., M_N. Quá trình tính giá trị băm của thông điệp là quá trình lũy tiến. Trước tiên block M_1 kết hợp với giá trị khởi tạo H_0 thông qua hàm F để tính giá trị hash H_1. Sau đó block M_2 được kết hợp với H_1 để cho ra giá trị hash là H_2. Block M_3 kết hợp với H_2 cho ra giá trị H_3. Cứ như vậy cho đến block M_N thì có giá trị băm của toàn bộ thông điệp là H_N (xem hình 4.7).

- H_0 là một dãy 128 bit được chia thành 4 từ 32 bit, ký hiệu 4 từ 32 bit trên là *abcd*. Với *a, b, c, d* là các hằng số như sau (viết dưới dạng thập lục phân):

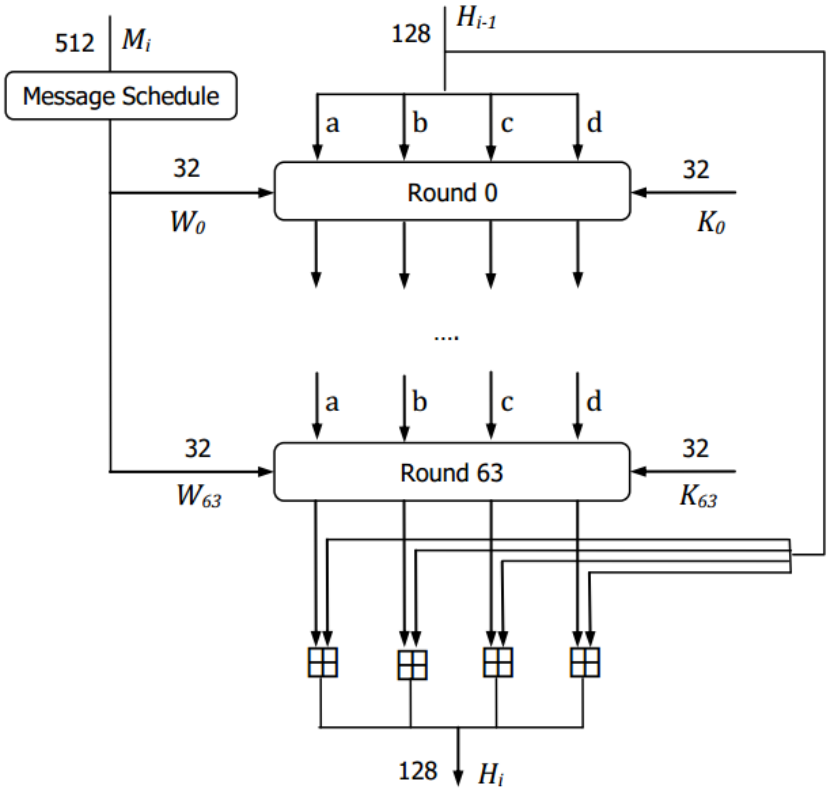
a = *01234567*

b = *89abcdef*

c = *fedbca98*

d = *76543210*

Cấu trúc của hàm F như sau:



Hình 4. 7. Sơ đồ cấu trúc hàm *F*

Tại mỗi bước lũy tiến, các giá trị *abcd* của giá trị hash H_{i-1} được biến đổi qua *64* vòng từ 0 đến *63*. Tại vòng thứ *j* sẽ có 2 tham số là K_j và W_j đều có kích thước 32 bit. Các tham số K_j được tính từ công thức: K_j là phần nguyên của số 2^{32} *abs(sin(i))* với *i* biểu diễn theo *rad*.

Giá trị block M_i 512 bit được biến đổi qua một hàm message schedule cho ra 64 giá trị W_0, W_1,…, W_{63} mỗi giá trị 32 bit. Block M_i 512 bit được chia thành 16 block 32 bit ứng với các giá trị W_0, W_1, …, W_{15} (16×32=512). Tiếp theo, 16 giá trị này được lặp lại 3 lần tạo thành dãy 64 giá trị.

Sau vòng cuối cùng, các giá trị *abcde* được cộng với các giá trị *abcd* của H_{i-1} để cho ra các giá trị *abcd* của H_i. Phép cộng ở đây là phép cộng modulo 2^{32}.

Tiếp theo tìm hiểu cấu trúc của một vòng. Việc biến đổi các giá trị *abcd* trong vòng thứ *i* được thể hiện trong hình bên dưới.

Ở đây $b \rightarrow c$, $c \rightarrow d$, $d \rightarrow a$.

Giá trị *b* được tính qua hàm:

$$t = a + f(b,c,d) + W_i + K_i$$

$$b = b + ROTL(t,s)$$

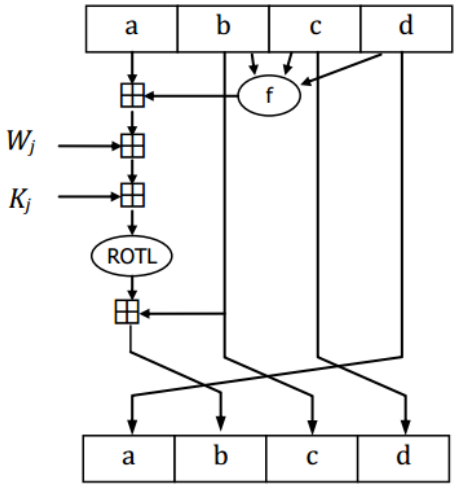
Trong đó : Hàm *f(x,y,z):*

$f(x,y,z) = (x \wedge y) \vee (\neg x \wedge z)$ nếu là vòng 0 đến 15

$f(x,y,z) = (z \wedge x) \vee (\neg z \wedge y)$ nếu là vòng 16 đến 32

$f(x,y,z) = x \oplus y \oplus z$ nếu là vòng 32 đến 48

$f(x,y,z) = y \oplus (x \vee \neg z)$ nếu là vòng 49 đến 63



Hình 4. 8. Cấu trúc biến đổi 1 vòng MD5

Hàm *ROTL*(t, *s*): t được dịch vòng trái s bit, với s là các hằng số cho vòng thứ *i* như sau:

Bảng 4. 2. Bảng tính giá trị ROTL(*t, s*)

i	*s*	*i*	*s*
0, 4, 8, 12	7	32, 36, 40, 44	4

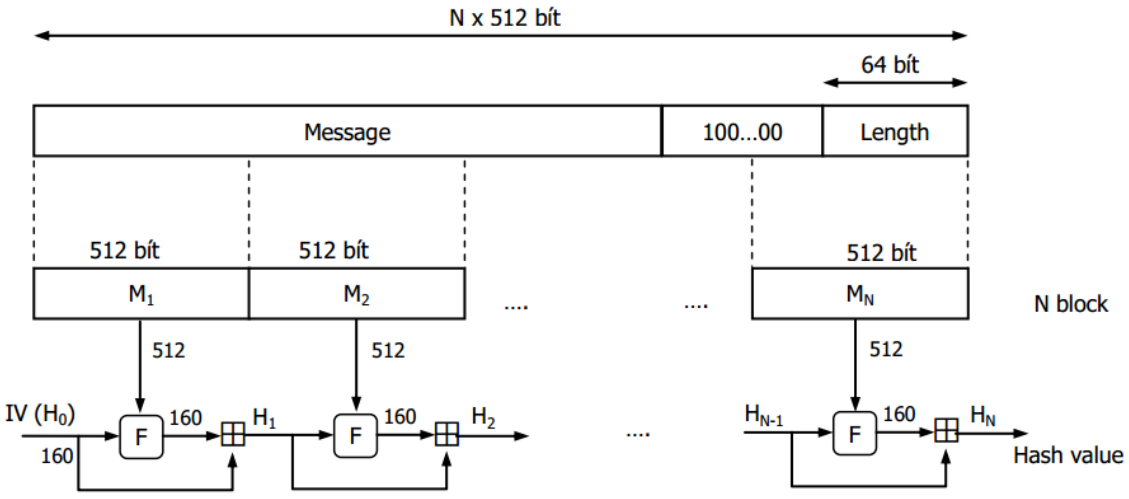
1, 5, 9, 13	12	33, 37, 41, 45	11
2, 6,10, 14	17	34, 38, 42, 46	16
3, 7, 11, 15	22	35, 39, 43, 47	23
16, 20, 24, 28	5	48, 52, 56, 60	6
17, 21, 25, 29	9	49, 53, 57, 61	10
18, 22, 26, 30	14	50, 54, 58, 62	15
19, 23, 27, 31	20	51, 55, 59, 63	21

Phép + (hay ⊞) là phép cộng modulo 2^{32}

4.2.2. Hàm băm họ SHA - Secure Hash Algorithm (SHA-1)

Hàm băm MD5 từ lúc ra đời đã có nhiều chuyên gia nhận định là không an toàn. Đặc biệt, vào năm 1994 và 1998, một phương pháp tấn công *MD5* đã được thực hiện thành công và một số thông điệp có cùng giá trị băm *MD5* được chỉ ra (2 thông điệp có cùng bản băm). Vì *MD5* không còn được xem là an toàn, nên người ta đã xây dựng thuật toán băm khác. Một trong những thuật toán đó là *SHA1* (*Secure Hash Algorithm*) mà đã được chính phủ Mỹ chọn làm chuẩn quốc gia.

Hàm băm *SHA*-1 tạo ra các giá trị băm có kích thước là *160* bit, kích thước đầu vào của *SHA*-1 là thông điệp có kích thước tối đa là 2^{64} bit. Sơ đồ tổng thể của *SHA-1* cũng giống như của *MD5* (xem hình 4.9).



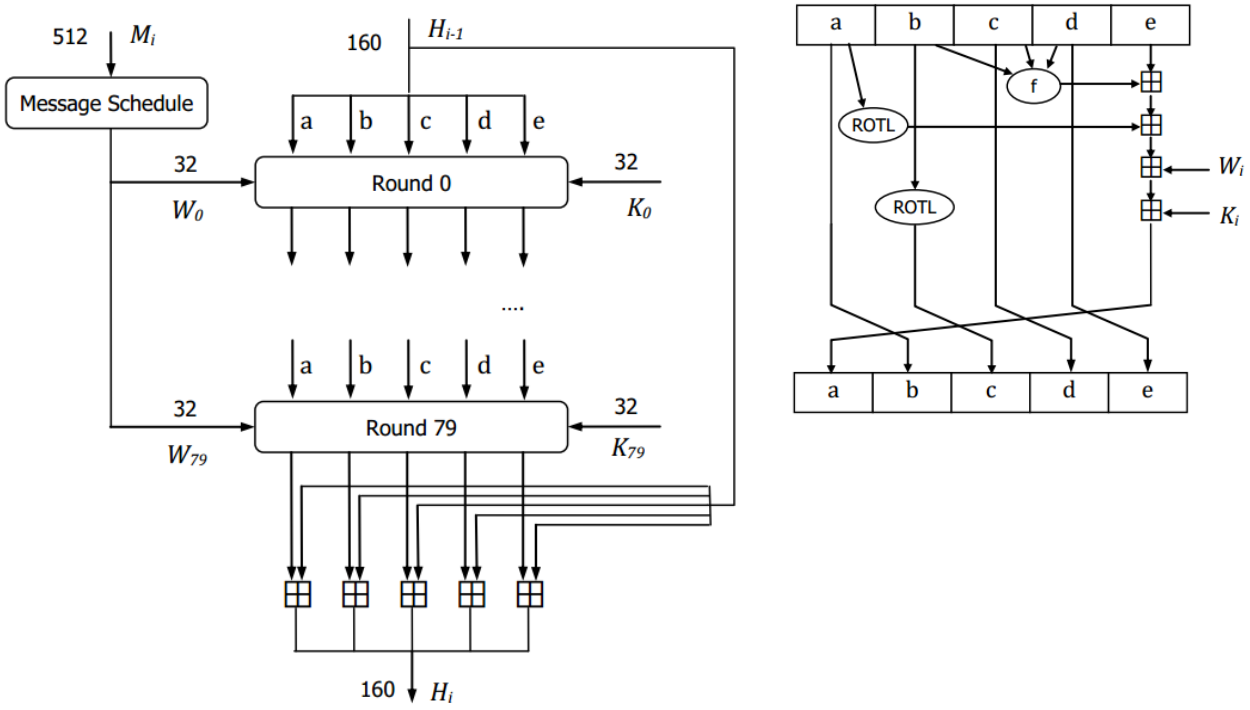
Hình 4. 9. Sơ đồ thuật toán băm SHA-1

Từ sơ đồ thuật toán 4.9 cho thấy quy trình làm việc của hàm băm *SHA1* như sau*:*

H_0 là một dãy 160 bit được chia thành 5 từ 32 bit, ký hiệu 5 từ 32 bit trên là *abcde*. Với *a, b, c, d, e* là các hằng số như sau:

a = 67452301 ; b = efcdab89 ; c = 98badcfe ; d = 10325476 ; e = c3d2e1f0

Cấu trúc của hàm *F* của *SHA* cũng tương tự như *MD5*, tuy nhiên được thực hiện trên 80 vòng (xem hình 4.6).



Hình 4. 10. Cấu trúc hàm F của thuật toán băm SHA-1

Giá trị $K_0, K_1, ..., K_{79}$ là các hằng số sau:

K_i = 5A827999 với $0 \le i \le 19$

K_i = 6ED9EBA1 với $20 \le i \le 39$

K_i = 8F1BBCDC với $40 \le i \le 59$

K_i = CA62C1D6 với $60 \le i \le 79$

Giá trị block M_i 512 bit được biến đổi qua một hàm message schedule cho ra 80 giá trị $W_0, W_1, \ldots, W_{80}$ mỗi giá trị 32 bit, theo quy tắc:

- Trước tiên block M_i 512 bit được chia thành 16 block 32 bit ứng với các giá trị $W_0, W_1, \ldots, W_{15}$ (16×32=512).

- Các giá trị W_t ($16 \le t \le 79$) được tính theo công thức:

  $W_t = ROTL\ (W_{t-3} + W_{t-8} + W_{t-14} + W_{t-16}\ ,\ 1)$ với phép cộng modulo 2^{32}.

- Việc biến đổi các giá trị *abcde* trong vòng thứ *i* được thể hiện trong hình bên trên.

Ở đây $a \to b$, $c \to d$, $d \to e$. Giá trị *a* và *c* được tính qua các hàm:

$a = ROTL(a,5) + f(b,c,d) + e + W_i + K_i$

$c = ROTL(b,30)$

Trong đó hàm *f(x,y,z)* được tính :

$f(x,y,z) = Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$ nếu là vòng 0 đến 19

$f(x,y,z) = Parity(x,y,z) = x \oplus y \oplus z$ nếu là vòng 20 đến 39

$f(x,y,z) = Maj(x,y,z) = (x \wedge y) \oplus (y \wedge z) \oplus (z \wedge x)$ nếu là vòng 40 đến 59

$f(x,y,z) = Parity(x,y,z) = x \oplus y \oplus z$ nếu là vòng 60 đến 79

Ý nghĩa của hàm Maj và hàm Ch:

- Hàm *Maj:* giả sử x_i, y_i, z_i là bit thứ *i* của *x*, *y*, *z*, thì bit thứ *i* của hàm *Maj* là giá trị nào chiếm đa số, 0 hay 1 (giống như hàm *maj* được định nghĩa trong phần thuật toán A5/1).
- Hàm *Ch*: bit thứ *i* của hàm *Ch là phép chọn*: *if* x_i *then* y_i *else* z_i.

4.3. CÂU HỎI VÀ BÀI TẬP

1) Hãy nêu định nghĩa về hàm băm ?
2) Hãy trình bày các tính chất của hàm băm có khóa và hàm băm không khóa ?
3) Hãy trình bày về các phương pháp phân loại hàm băm ?
4) Hãy mô tả thuật toán MDC-2 ?
5) Hãy mô tả thuật toán MDC-4 ?
6) Hãy mô tả thuật toán MAC ?
7) Hãy trình bày về hàm băm MD5 ? Hãy vẽ sơ đồ minh họa về hàm băm MD5 ?
8) Hãy trình bày về hàm băm SHA-1 ? Hãy vẽ sơ đồ minh họa về hàm băm SHA-1 ?

CHƯƠNG 5: THÁM MÃ

Chương này cung cấp các kiến thức cơ bản liên quan đến vấn đề thám mã bao gồm: định nghĩa, khái niệm, phân loại, cách thức tiến hành…. Ngoài ra, chương 5 còn trình bày một số kỹ thuật thám mã đối xứng và bất đối xứng đang được ứng dụng hiện nay như thám mã: vi sai, tuyến tính, nội suy, kênh bên, lựa chọn bản mã thích nghi...

5.1. Tổng quan về thám mã

Mật mã học từ lúc ra đời và được ứng dụng trong thực tế rộng rãi như ngày nay luôn mang trong mình quá trình đấu tranh để phát triển. Các thuật giải thuật mã hóa luôn bị thay thế bởi các giải thuật mã hóa mới nếu chúng được chứng minh không còn an toàn. Vậy làm sao biết được giải thuật mã hóa còn an toàn hay không? Đó chính là vấn đề của thám mã hay còn gọi là phân tích mã (cryptanalysis). Có nhiều quan điểm và định nghĩa khác nhau về thám mã. Tuy nhiên, theo quan điểm của tác giả có thể hiểu đơn giản về thám mã như sau: *Thám mã: là ngành khoa học chuyên nghiên cứu, áp dụng các phương pháp, kỹ thuật, thuật toán nhằm tìm ra những thông tin ý nghĩa và cần thiết cho người thám mã.* Những thông tin ý nghĩa ở đây có thể hiểu là: thông điệp (bản rõ) hoặc một phần thông điệp hoặc khóa bí mật.

Người thám mã: *là người thực hiện các hành vi, hành động nhằm khôi phục lại thông điệp hoặc khóa bí mật mà không có khóa*. Trong thực tế, người thực hiện thám mã có thể là các nhà mật mã, nhà phân tích mã, các chuyên gia trong lĩnh vực an toàn thông tin và những kẻ tấn công, nghe lén….. *Để thực hiện được hành động thám mã, người thám mã thường phải là người có trình độ, kinh nghiệm, hiểu biết về các giải thuật mã hóa và khả năng phân tích toàn học.*

Phương pháp thám mã: *là các thuật toán, kỹ thuật nhằm phân tích bản mã tìm ra khóa, hoặc bản rõ, hoặc cả khóa và bản rõ*. Như vậy, phương pháp thám mã cũng giống như phương pháp mã hóa là đều có những phương pháp và kỹ thuật chung. Tuy nhiên, sự khác biệt ở chỗ: phương pháp mã hóa sử dụng thuật toán, kỹ thuật nhằm biến đổi thông điệp thành bản mã, còn phương pháp thám mã dùng phương pháp, kỹ thuật, thuật toán nhằm tìm ra bản rõ hoặc quy luật mã hóa, hoặc khóa bí mật. Trong phần sau, bài giảng sẽ trình bày chi tiết hơn về vấn đề này.

Mục đích của thám mã: mục đích cao nhất của thám mã là nắm được nội dung thông điệp ban đầu hoặc đánh giá được mức độ an toàn của hệ thống hoặc của giải thuật mã hóa. Tùy theo từng phương pháp, kỹ thuật thám mã mà có thể thu được các kết quả khác nhau.

Vai trò của thám mã: có nhiều ý kiến khác nhau về vai trò của thám mã trong an toàn thông tin. Trên quan điểm tích cực, thám mã đã tìm ra được các điểm yếu của các giải thuật mã hóa. Điểm yếu này có thể được sử dụng để tấn công hệ mật, nhưng cũng có thể được sử dụng để cải tiến hệ mật cho tốt hơn. Chính vì vậy, thám mã giúp cho các nhà mật

mã và quản lý an toàn thông tin có lựa chọn giải pháp thay thế cho phù hợp. Tuy nhiên, trên quan điểm tiêu cực, thám mã đang tấn công vào các tính chất cơ bản của an toàn thông tin gây tổn thất về tài sản cho các tổ chức. Từ những phân tích trên, có thể thấy thám mã là tốt hay xấu còn tùy thuộc vào hành vi, mục đích của người thám mã. Nhưng rõ ràng, dù là mục đích tốt hay xấu, sự phát triển của thám mã làm thúc đẩy sự phát triển mật mã nói riêng và phát triển an toàn thông tin nói chung.

Phương pháp thám mã: Có nhiều phương pháp thám mã khác nhau, nhưng có thể tựu chung các phương pháp thám mã thành 3 loại chính như sau:

- Phương pháp vét cạn: đây là phương pháp thám mã sơ đẳng nhất, phương pháp này không đòi hỏi người thám mã phải có trình độ và hiểu biết về kỹ thuật mã hóa cũng như kinh nghiệm phân tích mã. Phương pháp này yêu cầu công cụ tính toán mạnh nhưng thời gian xử lý chậm. Đối với các giải thuật mã hóa hiện đại thì phương pháp này không còn khả thi vì thời gian vét cạn để tìm được khóa quá lâu.
- Phương pháp thống kê: đây là phương pháp thám mã đòi hỏi người thám mã phải có nhiều kinh nghiệm và kỹ năng bao gồm: kinh nghiệm và hiểu biết về mật mã, hiểu biết về một số phương pháp phân tích đặc trưng cơ bản của thông điệp như: Tần số (Frequency); Sự trùng lặp; Văn phong (Quy luật hành văn trong văn bản); *Quy luật tình huống… Đối với các giải thuật mã hóa hiện đại thì phương pháp thám mã dựa vào thống kê thường không mang lại hiệu quả cao do các giải thuật mã hóa hiện đại thường đảm bảo các tính chất là lộn xộn và rườm rà.*
- Phương pháp phân tích toán học: đây là phương pháp được áp dụng nhiều hiện nay. Các giải thuật mã hóa hiện đại thường không giấu phương pháp mã hóa mà chỉ giấu khóa. Chính vì vậy, người thám mã sử dụng phương pháp phân tích toán học để phân tích và tìm ra điểm yếu của thuật toán hoặc tìm ra khóa. Phương pháp phân tích toán học yêu cầu các nhà thám mã cần phải có kiến thức về toán học, đặc biệt là sự hiểu biết về các giải thuật mã hóa.

Chú ý: với mỗi giải thuật mã hóa đều có cơ chế và cơ sở toán học riêng để mã hóa thông tin đồng thời chống lại được các kỹ thuật thám mã đã biết trước. Chính vì vậy, để thám mã thành công người thám mã thường sử dụng kết hợp nhiều phương pháp và kỹ thuật. Mỗi phương pháp đều có ưu điểm và nhược điểm, sự kết hợp các ưu điểm của các phương pháp này sẽ mang lại hiệu quả cao nhất. Để nắm rõ hơn về sự kết hợp này, tiếp theo bài giảng sẽ trình bày về quy trình thám mã. Quy trình thám mã thể hiện các bước, giai đoạn thực hiện, từ quá trình tiền xử lý đến quá trình phân tích, đánh giá và cuối cùng là xây dựng phương pháp thám mã.

Quy trình thám mã: Có nhiều cách thức cũng như các phương pháp khác nhau để thực hiện một quy trình thám mã. Trong bài giảng sẽ trình bày một quy trình [2] để thấy được các bước, giai đoạn sẽ thực hiện trong phân tích mã. Sau khi nhận được các bản mã,

để tìm được ra nội dung thông điệp hoặc khóa hay bất cứ thông tin gì, các nhà thám mã thực hiện các bước sau:

- **Phân loại bản mã: ở bước này, từ những bản mã thu thập được, các nhà phân tích mã phân loại chúng thành các cụm và các lớp.** Đây là một bước quan trọng quyết định sự thành công hay thất bại của mã thám nên rất mất nhiều thời gian. Có nhiều phương pháp để phân cụm và phân lớp các bản mã mà không cần biết bản mã này là kết quả của giải thuật mã hóa nào.
- Xác định phương pháp mã hóa: ở bước này, các nhà thám mã đánh giá xem bản mã thu được thuộc phương pháp mã hóa nào. Các phương pháp chính được áp dụng trong việc xác định phương pháp mã hóa đã thực hiện như sau [2]: **Tần số (Frequency); Tính trùng mã; Tần số định kỳ; Phân tích kết quả tính các tần số và trùng mã…..** Việc này rất cần thiết và quan trọng vì nếu xác định được đúng phương pháp mã hóa sẽ giúp cho quá trình thám mã trở lên đơn giản hơn và có phương pháp tiếp cận chính xác hơn. Tuy nhiên, với các phương pháp mã hóa hiện đại, bước này có thể bỏ qua vì hầu hết các phương pháp mã hóa hiện đại đều công khai kỹ thuật mã hóa và chỉ giữ bí mật cho khóa.
- **Thám mã**: bước này thực hiện phân tích bản mã. Trong bước này gồm 2 công việc chính:
  - T**hám mã trực tiếp**: Nếu bản mã thuộc loại mã hóa truyền thống như các mã hóa thủ công hoặc được mã bằng một máy mã cụ thể nào đó mà có thuật toán thám mã thì có thể tiến hành thám trực tiếp (thực hiện thủ công và sau đó có thể tự động hoá bằng lập trình trên máy tính).
  - **Xây dựng phương pháp thám mã:** Nếu giải thuật mã hóa thuộc loại mới, công việc yêu cầu phức tạp cần phải xây dựng phương pháp thám mã cho phù hợp. Trong trường hợp này có thể áp dụng 2 kỹ thuật:
    - Phương pháp phân tích: được sử dụng trong trường hợp các nhà thám mã đã biết được cấu trúc khoá mã đã được sử dụng làm "mầm khoá" (key seed) để mã hoá bản mã này. Khi đó có nhiều kiểu để xác định khoá: phương pháp "thử - sai", phương pháp "lượng sai", phương pháp "những phần tử tách biệt", phương pháp "tuyến tính". Tóm lại tuỳ theo thuật toán mã hoá của bản mã như thế nào mà chọn phương pháp phân tích nào cho hợp lý.
    - Phương pháp dự đoán: Phương pháp này chủ yếu là dựa vào thông tin tiên nghiệm về khoá và thông tin về bản rõ mà đối tượng sử dụng (quy luật ngôn ngữ) để dự đoán khoá được sử dụng. Nội dung của phương pháp này là dự đoán cụm từ có thể xuất hiện trong bản rõ gốc ứng với bản mã, sau đó tìm cách xác định khoá đúng. Nếu khoá là đúng thì có thể dịch bản mã để cho ra bản rõ.

Từ quá trình phân tích trên có thể thấy rằng: mật mã và thám mã luôn tồn tại song song với nhau. Thám mã càng phát triển thì các yêu cầu đối với các giải thuật mã hóa càng

cao. Với sự phát triển của khoa học công nghệ và sự hỗ trợ đắc lực của các công cụ tính toán, thám mã đang thu được những kết quả rất to lớn và đang làm cho các nhà mật mã, chuyên gia an toàn thông tin phải đau đầu để đối phó.

5.2. Phân loại thám mã

Như đã trình bày ở trên, để thám mã thành công người thám mã cần có phương pháp tiếp cận, đánh giá phù hợp. Thực tế chỉ ra rằng: với mỗi giải thuật mã hóa thường các nhà thám mã sẽ có những biện pháp thám mã tương ứng nhằm mang lại hiệu quả cao nhất. Dưới đây, bài giảng sẽ trình bày một số phương pháp thám mã phổ biến hiện nay.

- Thám mã khi chỉ biết bản mã (ciphertext-only attack): Người thám mã chỉ có bản mã của một số thông điệp (*C1; C2; C3…Cn*), tất cả đều đã được mã hóa bằng các thuật toán mã hóa (*C1= E(P1, K); C2= E(P2, K); C3= E(P3, K); Cn= E(Pn, K)*). Mục tiêu của người thám mã là tìm cách lấy được nội dung một hoặc nhiều bản rõ hoặc lấy được khóa mật (*P1; P2…Pn* hoặc *K*). Đây là mô hình thám mã căn bản nhất trong đó người thám mã chỉ tiếp cận thông tin là các bản mã. Rõ ràng nếu một hệ mã mà không đứng vững được trong mô hình này thì phải đánh giá là không đáng tin cậy.

- Thám mã khi đã biết bản rõ (known-plaintext attack): Ở đây người thám mã chỉ có thể biết một số cặp bản rõ và bản mã tương ứng nào đó (*C1; P1); (C2; P2); (C3; P3)… (Cn; Pn)*. Mục tiêu của người thám mã là khám phá nội dung các bản rõ quan trọng khác và/hoặc lấy được khóa mật (*P* hoặc *K*) . Rõ ràng mô hình thám mã này mạnh hơn so với thám mã chỉ qua bản mã: Việc biết một số cặp bản mã/ bản rõ sẽ làm bổ sung thêm đầu mối phân tích. Đặc biệt người thám mã có thể dùng phép thử loại trừ để vét cạn không gian khóa và tìm ra khóa đúng.

- Thám mã bản rõ chọn sẵn (chosen plaintext attack): Trong mô hình này, không những người thám mã có được một số cặp bản rõ/ bản mã mà còn một số bản rõ mà do bản thân người thám mã soạn ra (chosen plaintext). Điều này nghe có vẻ không khả thi thực tế, tuy nhiên có thể hiểu là người thám mã bằng cách nào đó có được rất nhiều các bản rõ và bản mã của một tổ chức, từ đó người thám mã sẽ thêm nhiều cơ sở để phân tích quan hệ giữa bản mã và bản rõ để từ đó lần tìm giá trị khóa. Ví dụ: Người thám mã có qua hệ với người văn thư trong công ty. Từ người văn thư này, người thám mã biết được các văn bản trước khi gửi đi đều được mã hóa bằng kỹ thuật mã hóa nào đó. Chính vì vậy, để thám mã và tìm được thông tin cần tìm, người thám mã có thể lựa chọn các bản rõ và bản mã trước đó và từ đó tìm được quy luật mã hóa.

- Thám mã chọn bản mã thích nghi (adaptive-chosen-ciphertext attack): ở phương pháp thám mã này, người thám mã có thể thu thập được một số cặp bản rõ/bản mã mà bản mã là giá trị được thiết kế sẵn. Trong thực tế điều này có thể xảy ra nếu như người thám mã có thể truy nhập được vào máy mật mã 2 chiều (có thể sử dụng với cả 2 chức năng là sinh mã và giải mã). Để thám mã đạt kết quả tốt nhất, người thám mã thường sử dụng một chiến

thuật là thiết kế bản rõ hay bản mã chọn sẵn theo kiểu thích nghi (adaptive), tức là các bản rõ chọn sau có thể thiết kế dựa vào kiến thức phân tích các cặp bản rõ/bản mã đã thu thập từ trước.

Như vậy, có thể thấy rằng: mỗi phương pháp, kỹ thuật thám mã trên đều bao hàm rất nhiều thuật toán, giải thuật thám mã. Trong phần sau, bài giảng sẽ trình bày sâu hơn về vấn đề này.

Một điều cần chú ý: những kỹ thuật thám mã trình bày ở trên chỉ là những kỹ thuật thám mã phổ biến và không thể khẳng định là chỉ có những phương pháp thám mã này bởi vì do đặc thù của ngành thám mã, những người thám mã làm trong lĩnh vực bí mật, hoặc những kẻ tấn công thường giữ bí mật những kết quả thu được trong lĩnh vực thám mã bao gồm cả phương pháp thám mã và kết quả đánh giá. Chính vì vậy, khi ứng dụng và triển khai các giải thuật mã hóa trong thực tế, các chuyên gia an toàn thông tin cần đánh giá lại mức độ an toàn của các giải thuật mã hóa. Các giải thuật mã khóa không bị công khai các điểm yếu không có nghĩa là chúng đang an toàn mà có thể kẻ tấn công đã thám mã thành công và đang chờ đợi để khai thác các điểm yếu của các giải thuật này khi các giải thuật được triển khai trong thực tế. Tiếp theo, bài giảng sẽ trình bày một số kỹ thuật thám mã lên giải thuật mã hóa đối xứng và bất đối xứng.

5.3. Một số kỹ thuật thám mã hệ mật khóa đối xứng

5.3.1. Phương pháp thám mã tuyến tính

Thám mã tuyến tính là một phương pháp thám mã đã biết bản rõ. Kỹ thuật thám mã này lợi dụng mối quan hệ tuyến tính giữa đầu vào và đầu ra của một hệ mã để thực hiện thám mã.

Cách tiếp cận của kỹ thuật này là phân tích các thành phần phi tuyến tính và ước lượng chúng bằng các biểu diễn xấp xỉ thành phần. Những biểu diễn tuyến tính thành phần này sau đó được kết hợp lại để tạo thành một biểu diễn xấp xỉ tuyến tính của toàn bộ hệ mã. Biểu thức xấp xỉ của toàn bộ hệ mã sẽ bao gồm các bit bản rõ, các bit bản mã và các bit khóa.

Trong thực tế, kỹ thuật thám mã tuyến tính đã áp dụng thành công lên giải thuật mã hóa DES. Tiếp theo, để giúp sinh viên hiểu được cơ chế của kỹ thuật thám mã tuyến tính, bài giảng sẽ trình bày chi tiết hơn về cách thức tấn công của kỹ thuật tuyến tính lên giải thuật mã hóa DES.

Như đã trình bày ở trên, để thám mã một giải thuật mã hóa, kỹ thuật thám mã tuyến tính thường phân tích các thành phần phi tuyến tính và đưa chúng về biểu thức thành phần xấp xỉ. Đồng thời, khi đã có các biểu thức thành phần xấp xỉ, kỹ thuật thám mã tuyến tính sẽ tìm cách xác định biểu thức xấp xỉ của một hệ mã hoàn chỉnh. Như đã biết, trong giải thuật mã hóa DES chỉ có các hộp *S - box* là các phép biến đổi phi tuyến tính, các phép biến đổi còn lại chỉ sử dụng các phép toán hoán vị và XOR nên các giá trị giữa bản mã và bản

rõ vẫn liên hệ tuyến tính với nhau. Lúc này với ẩn số là khóa *K* cho trước cố định, bằng công cụ mô phỏng trên máy tính và sử dụng các cặp bản rõ/ bản mã tương ứng kẻ tấn công có thể thiết lập được hệ thống phương trình tuyến tính để tìm lại được các bit của khóa *K* trong thời gian đa thức

Mục đích của phương pháp thám mã tuyến tính trên giải thuật mã hóa 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 cạn . Để thực hiện được mục đích này thì cần khai thác các nhược điểm của hộp *S – box* và đồng thời xác định các biểu thức xấp xỉ thành phần và biểu thức xấp xỉ hoàn chỉnh.

- **Xác định biểu thức xấp xỉ của các thành phần:** Phép toán phi tuyến tính trong giải thuật mã hóa DES là sự thay thế được thực hiện bởi các *S-box*. Trong mỗi vòng của giải thuật mã hóa DES sẽ có 8 *S-box*. Mỗi *S-box* có thể được biểu diễn xấp xỉ với biểu thức có dạng:

$$X_{i1} \oplus X_{i2} \oplus \ldots \oplus X_{iu} \oplus Y_{j1} \oplus Y_{j2} \ldots \oplus Y_{jv} = 0 \qquad (5.1)$$

Trong đó X_i , Y_i là các bit đầu vào và đầu ra của hộp *S-box* tương ứng.

Mỗi một *S-box* với *u* bit đầu vào và *v* bit đầu ra có $(2^u - 1) \times (2^v - 1)$ xấp xỉ tuyến tính có thể có. Như vậy, với 6 bit đầu vào và 4 bit đầu ra của mỗi *S –box*, giải thuật mã hóa DES sẽ có $(2^6 - 1) * (2^4 - 1)$ xấp xỉ tuyến tính có thể có.

Do mỗi *S-box* có thể có nhiều xấp xỉ nên cần lựa chọn giá trị xấp xỉ tốt nhất để phục vụ cho việc xác định giá trị xấp xỉ của toàn hệ mã. Xác suất của xấp xỉ nên được giới hạn là ½ để đảm bảo nó cho một xấp xỉ tốt, đảm bảo giá trị xấp xỉ trong biểu thức 4.1 có nghĩa. Trong tài liệu [9] đã chứng minh các cuộc thám mã sẽ không có hiệu quả nếu hệ mã là một hàm ngẫu nhiên với tất cả các xấp xỉ đều có một xác suất là ½. Nhận định này là chính xác bởi vì nếu p=½t hì ε=0 khi đó phương trình 5.1 sẽ không còn phù hợp nữa. Để tiến hành thám mã hiệu quả, giá trị xấp xỉ với sai số (độ lệch) cao nhất sẽ được lựa chọn. Giá trị sai số (độ lệch - ε) của xấp xỉ được xác định theo công thức:

$$\varepsilon = \left| p - \frac{1}{2} \right| \qquad (5.2)$$

Trong đó: *p* là xác suất mà xấp xỉ tuyến tính này đúng (tức là *p* là xác suất để giá trị XOR của các bit đầu vào và các bit đầu ra bằng giá trị 0 – thỏa mãn phương trình (*5.1)*. *p* được xác định bằng cách tính số lần xuất hiện kết quả đúng tương ứng với đầu vào và đầu ra của một *S-box* và sau đó chia cho số lượng đầu vào có thể của *S-box* tương ứng.

Nhận xét: Như vậy kẻ tấn công sau khi xác định được độ hiệu quả ε từ xác suất *p* thì đánh giá mức độ an toàn của hệ mã và từ đó việc tìm các bit khóa rồi thực hiện thám mã toàn bộ hệ mã. Kết quả của việc xác định các biểu thức xấp xỉ thành phần sẽ giúp cho kẻ tấn công thực hiện tấn công hiệu quả hơn. Tiếp theo, kẻ tấn công có trong tay các xấp xỉ

thành phần thì sẽ tiếp tục tìm kiếm và xác định các biểu thức và giá trị của biểu thức xấp xỉ hoàn chỉnh của toàn hệ mã để từ đó thực hiện tìm khóa.

- **Xác định biểu thức xấp xỉ của một hệ mã hoàn chỉnh:** Khi các *S-box* được tính xấp xỉ, các xấp xỉ này được kẻ tấn công kết hợp từ vòng này sang vòng khác nhằm lấy được một biểu thức xấp xỉ tuyến tính của hệ mã hoàn chỉnh. Xấp xỉ này phải có một sai số ε cao để thực hiện thám mã hiệu quả. Khi sai số của biểu thức xấp xỉ hoàn chỉnh được tính toán thì người thám mã có thể thực hiện việc thám mã. Để xác định giá trị sai số của xấp xỉ cho một hệ mã hoàn chỉnh, kẻ tấn công có thể sử dụng bổ đề Piling-Up. Trong công bố [9], các tác giả đã trình bày chi tiết về bổ đề Piling-Up để xác định giá trị sai số của xấp xỉ cho một hệ mã hoàn chỉnh.

Lưu ý: Các xấp xỉ của *S-box* trong giải thuật mã hóa DES không phải độc lập. Tuy nhiên, trong thực tế khi thực nghiệm thám mã giải thuật mã hóa DES các nhà thám mã thường giả định là các xấp xỉ của *S-box* độc lập.

Cách thức tiến hành thám mã: Như vậy, sau quá trình tính toán, kẻ tấn công đã tính toán được các xấp xỉ thành phần và xấp xỉ hoàn chỉnh của các hàm *S-box* trong giải thuật mã mã DES. Tiếp theo, kẻ tấn công sẽ thực hiện các bước sau để thực hiện thám mã các vòng của giải thuật mã hóa DES.

Bước 1: Lựa chọn số lượng lớn các bản rõ (N_p) và các cặp bản rõ/bản mã được yêu cầu. Các cặp bản rõ/ bản mã càng nhiều thì kẻ tấn công càng thuận lợi trong việc tính toán các biểu thức xấp xỉ.

Bước 2: Loại bỏ vòng lặp cuối cùng của hệ mã. Để tấn công hệ mã DES với *16* vòng lặp, kẻ tấn công bắt đầu bằng cách loại bỏ vòng lặp cuối cùng của hệ mã bằng cách tìm các bit khóa kết hợp của nó. Sau đó tấn công làm giảm hệ mã với *15* vòng lặp và loại bỏ các vòng tiếp theo. Quá trình này được lặp lại như vậy cho đến khi thu được các bit khóa kết hợp với tất cả các vòng của hệ mã.

Bước 3: Tấn công vòng cuối cùng: Để tấn công đến vòng cuối cùng, kẻ tấn công xây dựng một biểu thức tuyến tính bao gồm các bit bản rõ và các bit đi vào vòng cuối cùng của hệ mã:

$$X_{i1} \oplus X_{i2} \oplus \ldots \oplus U_{j1} \oplus U_{j2} \oplus \ldots = 0$$

Trong đó:

X_i là các bit bản rõ

U_j là các bit bản rõ đi vào vòng cuối cùng của hệ mã tương ứng.

Cuộc tấn công được lặp lại để thu hồi khóa kết hợp với các vòng còn lại. Nếu khóa được lựa chọn không hợp lệ thì các khóa khác với sai số cao hơn được thử.

Trong tài liệu [9, 10, 11] đã trình bày một số ví dụ cụ thể về thực hiện thám mã trên một số vòng của thuật mã hóa DES. Trong thực tế, với việc áp dụng phương pháp thám mã tuyến tính, các chuyên gia bảo mật đã thực hiện thám mã thành công đến vòng thứ 7 của giải thuật mã hóa DES. Tuy nhiên, phương pháp thám mã tuyến tính cần biết trước 2^{43} cặp bản rõ/ bản mã. Đây là một con số lớn nên thám mã tuyến tính cũng không phải là một phương pháp khả thi.

5.3.2. Phương pháp thám mã vi sai

Phương pháp thám mã vi sai do Biham và Shamir đưa ra. Thám mã vi sai là một phương pháp thám mã giải thuật mã hóa DES rất nổi tiếng. Đây là một phương pháp thám mã với bản rõ chọn lọc. Nó phân tích sự ảnh hưởng của sự khác nhau trên các cặp bản rõ lên các cặp bản mã tương ứng. Sự sai khác này có thể được dùng để xác định xác suất của các khả năng khóa và tìm ra khóa có xác suất cao nhất. Phương pháp này thường hoạt động trên nhiều cặp bản rõ với vi sai nhất định và sử dụng các cặp bản mã tương ứng. Đối với các hệ mật giống với DES vi sai được chọn bằng giá trị XOR cố định của 2 bản rõ. Tiếp theo, để hiểu hơn về cách thức và quy trình thám mã của kỹ thuật thám mã vi sai, hãy cùng xét ví dụ về thám mã vi sai trên giải thuật mã hóa DES.

Các ký hiệu được sử dụng**:**

n_x : một số hệ hexa được kí hiệu bằng chỉ số x (ví dụ: $10_x = 16$)

X^*, X' : Ở bất kì vị trí trung gian nào trong quá trình mã hóa của cặp bản rõ, X, X^* lần lượt là giá trị trung gian của 2 quá trình thực hiện của thuật toán, và X' được định nghĩa bằng $X' = X \oplus X^*$

$P(X)$: Hoán vị P được kí hiệu bằng $P(X)$

$E(X)$: Mở rộng E được kí hiệu bằng $E(X)$

$IP(X)$: Hoán vị khởi tạo. Trong tài liệu này bỏ qua IP và IP^{-1}.

P : Bản rõ (sau khi được hoán vị khởi tạo IP) được kí hiệu bằng P. P^* là một bản rõ khác trong cặp và $P' = P \oplus P^*$ là giá trị XOR của cặp bản rõ.

T : Bản mã tương ứng của P, P^* (trước khi được hoán vị IP^{-1}) được kí hiệu bằng T, T^* $T' = T \oplus T^*$ là giá trị XOR của cặp bản mã.

(L, R) : Nửa trái và nửa phải của bản rõ P được kí hiệu bằng L và R.

(l, r) : Nửa trái và nửa phải của bản mã T được kí hiệu bằng l và r.

$a, ..., j$: 32 bit đầu vào của hàm F.

$A, ..., J$: 32 bit đầu ra của hàm F.

Si : Hộp S-box S1,S2,…,S8

$Si_{\mathrm{EX}}, Si_{\mathrm{KX}}, Si_{\mathrm{IX}}, Si_{\mathrm{OX}}$: Đầu vào của Si trong vòng X được kí hiệu là Si_{IX} với $X \in \{a,..., j\}$. Đầu ra của Si trong vòng X được kí hiệu là Si_{OX}. Giá trị của 6 bit khóa của hộp *S-box* Si được kí hiệu là Si_{KX} và giá trị của 6 bit đầu vào từ $E(X)$ được XOR với Si_{KX} để tạo nên Si_{IX} được kí hiệu là Si_{EX}.

Lưu ý rằng: với ví dụ về thám mã vi sai trên giải thuật mã hóa DES chỉ xét hạn chế DES *n* vòng với $n \leq 16$. Bởi vậy, với các điều kiện trên, có thể coi L_0R_0 là bản rõ và L_nR_n là bản mã trong DES *n* vòng (cần chú ý rằng không cần đảo L_nR_n).

Cách thức tiến hành thám mã như sau:

Bước 1: Xác định giá trị XOR của đầu vào và đầu ra của hàm *F*

***Lưu ý*: *Khóa độc lập là danh sách các khóa con mà không nhất thiết được tạo ra từ thuật toán sinh khóa*.** Để đơn giản hóa quá trình phân tích xác suất, các nhà thám mã thống nhất là mọi khóa con đều độc lập.

Xác định trước giá trị XOR của cặp đầu vào của hàm F dễ dàng xác định giá trị XOR sau khi được qua hàm mở rộng bởi công thức sau:

$$E(X) \oplus E(X^*) = E(X \oplus X^*)$$

Phép XOR với khóa không làm thay đổi giá trị XOR của cặp đầu vào.

$$(X \oplus K) \oplus (X^* \oplus K) = X \oplus X^*$$

Đầu ra của *S-box* được xáo trộn bởi phép hoán vị P vì vậy giá trị XOR của cặp sau khi đi qua phép hoán P là hoán vị của giá trị XOR đầu ra của *S-box*:

$$P(X) \oplus P(X^*) = P(X \oplus X^*)$$

Giá trị XOR đầu ra của hàm F là tuyến tính trong phép XOR kết nối các vòng khác nhau:

$$(X \oplus Y) \oplus (X^* \oplus Y^*) = (X \oplus X^*) \oplus (Y \oplus Y^*)$$

Như vậy, rõ ràng giá trị XOR sẽ không đổi đối với khóa và tuyến tính trong hàm E, P và phép XOR.

Mặt khác, hộp *S-box* có tính phi tuyến, biết giá trị XOR của cặp đầu vào không bảo đảm biết được giá trị XOR đầu ra. Bởi vì một cặp đầu sẽ có một vài khả năng cho giá trị XOR đầu ra. Bên cạnh đó, nhận thấy rằng với một giá trị XOR đầu vào không phải tất cả tất cả các khả năng giá trị XOR đầu ra xuất hiện đều nhau, vì một số giá trị XOR sẽ xuất hiện với tần số cao hơn giá trị khác. Kết quả thu được tại bước 1 xác định được giá trị XOR (vi sai) của cặp đầu vào.

Bước 2: Tạo bộ đầu vào *S-box*

Như vậy, kết quả bước 1 kẻ tấn công sẽ tìm được giá trị vi sai của cặp đầu vào của hàm *F*. Từ kết quả này, kẻ tấn công lựa chọn các bản rõ thỏa mãn giá trị vi sai vừa tìm được. Để thực hiện điều đó, kẻ tấn công làm như sau:

- Tạo một bộ 64 cặp (*Si, Si**) sao cho giá trị XOR từng cặp bằng với giá trị XOR của đầu vào cho ở bước 1.

- Tính bộ 64 cặp đầu vào *S-box* (Si_{IX}, Si^*_{IX}) thỏa mãn $Si_{IX} \oplus Si^*_{IX}$ = E(Si)⊕E(Si*).

Giả sử ta chọn giá trị XOR = 34_x việc tính bộ 64 cặp đầu vào sẽ diễn ra như sau:

- Vì $X \oplus X^*$ = 34 nên ta sẽ chọn lần lượt các giá trị của X để suy ra giá trị X^* tương ứng. Sau bước 2 ta thu được các cặp đầu vào tương ứng giống như phần Possible Input trong Bảng 4.2 .

Giả sử xor đầu vào ở bước 1 A=01100..00 độ dài 32 bit là nửa phải bản rõ.

Ta có S_i xor S*i =A. Khi đó S_i=01100..00 độ dài 32 bit

S*i =0000…00 độ dài 32 bit

Ta tính cặp đầu vào S-box(S_{iIX},S^*_{iIX})

S_{iIX} xor S^*_{iIX}=E(Si) xor E(S*i)

=001100…0 xor 000000…0

=001100…00 độ dài 48 bit

Ta xét đi qua hộp S_1 nên có đầu vào là 6 bit đầu

Ta có cặp đầu vào S-box 1 là (S_{iIX},S^*_{iIX}) có giá trị xor là 001100 và bằng 0Cx;

Vì có 6 bit đầu vào nên sẽ có 2^6 =64 giá trị. Ta lấy 0Cx xor với 64 giá trị ta sẽ có được 64 cặp đầu vào S-box

(0,C);(1,D),(2,E),(3,F),(4,8),(5,9),(6,a),(7,b),

(10,1c),(11,1d),(12,1e),(13,1f),(14,18),(15,19),(16,1a),(17,1b),

(20,2c),(21,2d),(22,2e),(23,2f),(24,28),(25,29),(26,2a),(27,2b),

(30,3c),(31,3d),(32,3e),(33,3f),(34,38),(35,39),(36,3a),(37,3b),

Đổi vị trí S1 và S*1 ta được thêm một cặp.

Bước 3: Đối chiếu với bảng phân bố cặp giá trị XOR của *S-box*

Sau khi thực hiện thành công bước 2, kẻ tấn công sẽ có trong tay danh sách các cặp bản rõ thỏa mãn giá trị vi sai. Tiếp theo, kẻ tấn công cần tính xác suất của vi sai đầu ra tương ứng với vi sai đầu vào. Do biết giá trị XOR của cặp đầu vào không bảo đảm biết được giá trị XOR đầu ra. Chính vì vậy, trong bước này, kẻ tấn công sẽ thực hiện đối chiếu các kết quả thu được ở bước trên với bảng phân bố cặp giá trị XOR của *S-box (xem bảng 5.1).* Xây dụng bảng phân bố cặp giá trị XOR của *S-box* dựa trên một số quy tắc và định nghĩa sau:

Định nghĩa 1: *Một bảng biểu diễn sự phân bố của giá trị XOR đầu vào và đầu ra của tất cả các cặp của S-box được gọi là bảng phân bố cặp giá trị XOR của S-box. Trong bảng*

này mỗi hàng ứng với một giá trị XOR đầu vào xác định, mỗi cột ứng với giá trị XOR đầu ra và mỗi mục trong bảng là số các cặp với giá trị XOR đầu ra và đầu vào tương ứng. Mỗi dòng trong bảng chứ 64 cặp trong 16 mục khác nhau. Chính vì vậy trong mỗi dòng trong bảng giá trị trung bình của mỗi mục là 4.

Để áp dụng quy tắc này ta xét ví dụ sau:

Như chúng ta thấy ở bước 2 với mỗi giá trị xor đầu vào sẽ có 64 cặp đầu vào xor với nhau thỏa mãn giá trị đó nên tổng của mỗi hàng là 64. Sau đó chúng ta cho từng đầu vào đi qua s-box rồi sau đó xor chúng với nhau. Ta có 64 cặp giá trị (S_{1I},S^*_{1I}) xor với nhau cho ra đầu vào và đầu ra của phép xor có giá trị bằng 0_x. Do đó giá trị trong bảng đầu vào của phép xor bằng 0_x và đầu ra của phép xor bằng 0_x bằng 64.

Ví dụ:Ta có 1 cặp đầu vào:

S_{1I}=12x=010010

S^*_{1I}=1Ex=011110

Sử dụng hộp S_1 để tìm cặp đầu ra:

Nguyên tắc thực hiện: Ta bỏ bit đầu và bit cuối và áp dụng hàm S-boxes tìm được cặp đầu ra là :

S_{1O}=1001=A_x

S^*_{1O}=0111=7_x

Input xor=$S_{1I} \oplus S^*_{1I}$=$0C_x$

Output xor=$S_{1O} \oplus S^*_{1O}$ =$0D_x$

Từ đây ta có 1 cặp giá trị đầu vào(S_{1I},S^*_{1I})=(12_x,$1E_x$) có input xor là $0C_x$ và đầu ra của phép xor là $0D_x$. Tương tự ta tìm thêm được 5 cặp có giá trị input xor là $0C_x$ và đầu vào của phép xor tương ứng là $0D_x$. Do đó giá trị trong bảng đầu vào của phép xor bằng $0C_x$ và đầu ra của phép xor bằng $0D_x$ bằng 6.

Tương tự, ta có bảng phân bố cặp giá trị vi sai vào-ra của hộp S1 như trong bảng 5.1.

Định nghĩa 2: *Với X là số 6 bit và Y là số 4 bit. Khi đó X có thể tạo ra Y thông qua S-box nếu có một cặp mà giá trị XOR đầu vào của S-box bằng với X và giá trị XOR đầu ra của S-box bằng với Y. Nếu có cặp như vậy sẽ biểu diễn $X \rightarrow Y$, và nếu không có cặp nào như vậy thì nói X không tạo ra Y thông qua S-box.*

Bảng 5. 1. Bảng phân bố cặp giá trị XOR của S_1

Input XOR	Output XOR															
	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2_x	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3_x	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4_x	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5_x	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6_x	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7_x	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
8_x	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
9_x	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
A_x	0	8	6	2	2	8	6	0	6	4	6	0	4	0	2	10
B_x	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
C_x	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
D_x	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
E_x	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
F_x	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
									⋮							
30_x	0	4	6	0	12	6	2	2	8	2	4	4	6	2	2	4
31_x	4	8	2	10	2	2	2	2	6	0	0	2	2	4	10	8
32_x	4	2	6	4	4	2	2	4	6	6	4	8	2	2	8	0
33_x	4	4	6	2	10	8	4	2	4	0	2	2	4	6	2	4
34_x	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6
35_x	2	2	4	0	8	0	0	0	14	4	6	8	0	2	14	0
36_x	2	6	2	2	8	0	2	2	4	2	6	8	6	4	10	0
37_x	2	2	12	4	2	4	4	10	4	4	2	6	0	2	2	4
38_x	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39_x	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
$3A_x$	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
$3B_x$	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
$3C_x$	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
$3D_x$	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
$3E_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
$3F_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

Định nghĩa 3: *X có thể tạo ra Y với xác suất p của hộp S-box với p là tỷ lệ số cặp với giá trị XOR đầu vào bằng X và giá trị XOR đầu ra bằng Y.*

Ví dụ 1 : $34_x \rightarrow 2_x$ đối chiếu bảng 5.1 ta sẽ có được 16 cặp. $34_x \rightarrow 7_x$ đối chiếu bảng 5.1 ta sẽ có được 12 cặp.

Xác suất p của $34_x \rightarrow 2_x$ là 16/64=1/4;

Xác suất p của $34_x \rightarrow 7_x$ là 12/64=3/16;

Từ ví dụ $34_x \rightarrow 2_x$ có được từ trong 16 cặp từ 64 cặp của S_1, với xác suất $\frac{1}{4}$. Sự phân bố xuất hiện khác nhau ở các hàng khác nhau trong bảng. Có tổng số 70% đến 80% các mục là có khả năng xảy ra, còn 30% và 20% là không thể xảy ra. Bảng phân bố cặp XOR sẽ giúp tìm được các khả năng giá trị đầu vào và đầu ra với cặp XOR cho trước. Để hiểu hơn về quy tắc này, cùng xét ví dụ 2:

Bước 4: Tạo bảng giá trị XOR đầu vào đầu ra tương ứng

Kết quả ở bước 3, kẻ tấn công đã thu được danh sách các vi sai đầu ra với xác suất xuất hiện tương ứng. Tiếp theo, kẻ tấn công cần tính các giá trị vi sai đầu ra của 64 cặp đầu vào *S – box* đã lựa chọn ở bước 2. Để thực hiện điều đó, kẻ tấn công sẽ làm như sau:

- Xác định giá trị XOR sau khi qua *S-box* và hàm *P* của từng cặp trong bộ 64 cặp (Si_{IX}, Si^*_{IX}) .
- Thiết lập bảng gồm 2 trường: XOR đầu ra sau khi qua *S-box* và hàm *P* và các cặp giá trị (Si_{IX}, Si^*_{IX}) tương ứng.

Ví dụ 2: Xét mục $34_x \rightarrow 4_x$ trong bảng phân bố cặp XOR của S_1. Vì mục $34_x \rightarrow 4_x$ có giá trị 2, chỉ có 2 cặp thỏa mãn cặp XOR này. 2 cặp này là đối ngẫu. Nếu cặp đầu tiên là $S1_I, S1^*_I$ thì cặp còn lại là $S1^*_I, S1_I$. Bảng dưới đây cho thấy 2 giá trị đầu vào này phải là $13_x, 27_x$ với giá trị đầu ra tương ứng là $6_x, 2_x$.

Bảng 5. 2. Ví dụ về các giá trị đầu vào ứng với vi sai $S1'_I = 34_x$

Output XOR $(S1'_O)$	Possible Inputs $(S1_I)$
1	03, 0F, 1E, 1F, 2A, 2B, 37, 3B
2	04, 05, 0E, 11, 12, 14, 1A, 1B, 20, 25, 26, 2E, 2F, 30, 31, 3A
3	01, 02, 15, 21, 35, 36
4	13, 27
7	00, 08, 0D, 17, 18, 1D, 23, 29, 2C, 34, 39, 3C
8	09, 0C, 19, 2D, 38, 3D
D	06, 10, 16, 1C, 22, 24, 28, 32
F	07, 0A, 0B, 33, 3E, 3F

Ví dụ thêm về XOR = 35_x

Bảng 5. 3. Ví dụ về các giá trị đầu vào ứng với vi sai $S1'_I = 35_x$

Đầu ra XOR	Các khả năng của đầu vào
0	23, 16
1	39, C
2	3F, A, 3E, B
4	24, 11, 29, 1C, 2D, 18, 3D, 8
8	20, 15, 27, 12, 32, 7, 31, 4, 30, 5, 36, 3, 3B, E
9	21, 14, 34, 1
A	33, 6, 37, 2, 22, 17
B	25, 10, 3A, F, 3C, 9, 2C, 19
D	35, 0
E	26, 13, 2B, 1E, 2A, 15, 28, 1D, 2F, 1A, 2E, 38, 1B, D

Ví dụ: Với 64 cặp đầu vào S-box đã chọn được từ bước 2, ta thực hiện tính toán giá trị xor đầu ra sau khi qua hàm S-box 1 cho từng cặp.

Bảng 5. 4. Kết quả các phép xor của từng cặp

Cặp đầu vào 16 bit	Cặp đầu vào 2 bit	Giá trị xor đầu ra 2 bit	Giá trị xor đầu ra 16 bit
0, C	000000, 001100	1110 xor 1011 = 0101	5
1, D	000001, 001101	0000 xor 1101 = 1101	D
2, E	000010, 001110	0100 xor 1000 = 1100	C
3, F	000011, 001111	1111 xor 0001 = 1110	E
4, 8	000100, 001000	1101 xor 0010 = 1111	F
5, 9	000101, 001001	0111 xor 1110 = 1001	9
6, A	000110, 001010	0001 xor 1111 = 1110	E
7, B	000111, 001011	0100 xor 0010 = 0110	6
10, 1C	010000, 011100	0011 xor 0000 = 0011	3
11, 1D	010001, 011101	1010 xor 0011 = 1001	9
12, 1E	010010, 011110	1010 xor 0111 = 1101	D
13, 1F	010011, 011111	0110 xor 1000 = 1110	E
14, 18	010100, 011000	0110 xor 0101 = 0011	3
15, 19	010101, 011001	1100 xor 1001 = 0101	5
16, 1A	010110, 011010	1100 xor 1001 = 0101	5
17, 1B	010111, 011011	1011 xor 0101 = 1110	E
20, 2C	100000, 101100	0100 xor 0010 = 0110	6
21, 2D	100001, 101101	1111 xor 0001 = 1110	E
22, 2E	100010, 101110	0001 xor 1011 = 1010	A
23, 2F	100011, 101111	1100 xor 0111 = 1011	B
24, 28	100100, 101000	1110 xor 1101 = 0011	3
25, 29	100101, 101001	1000 xor 0100 = 1100	C
26, 2A	100110 ,101010	1000 xor 0110 = 1110	E
27, 2B	100111, 101011	0010 xor 1001 = 1011	B
30, 3C	110000, 111100	1111 xor 0101 = 1010	A
31, 3D	110001, 111101	0101 xor 0110 = 0011	3
32, 3E	110010, 111110	1100 xor 0000 = 1100	C
33, 3F	110011, 111111	1011 xor 1101 = 0110	6
34, 38	110100, 111000	1001 xor 0011 = 1010	A
35, 39	110101, 111001	0011 xor 1010 = 1001	9
36, 3A	110110, 111010	0111 xor 1010 = 1101	D
37, 3B	110111, 111011	1110 xor 0000 = 1110	E

Thiết lập bảng tổng kết gồm 2 trường, giá trị XOR đầu ra sau khi qua hàm S-box và hàm P và các cặp giá trị đầu vào ($SiIX$, $SiIX*$) tương ứng. Do đang sử dụng giá trị XOR đầu ra của hàm S-box 1 nên ở cột giá trị sau khi qua hàm P (cột thứ 3) chỉ hiển thị vị trí được cập nhật của 4 bit đầu tiên

Bảng 5. 5. Kết quả thực nghiệm các cặp thỏa mãn

Giá trị XOR đầu ra	Các cặp đầu vào thỏa mãn	Giá trị đầu ra hàm P (32 bit) P (32 bit) (o1o2o3…o31o32)
3 = **0011**	(10, 1C); (14, 18); (24; 28); (31, 3D)	$o_1o_2\ldots o_8\mathbf{0}..o_{16}\mathbf{0}..o_{22}\mathbf{1}..o_{30}\mathbf{1}o_{32}$
5 = **0101**	(0, C); (15, 19); (16, 1A)	$o_1o_2\ldots o_8\mathbf{0}..o_{16}\mathbf{1}..o_{22}\mathbf{0}..o_{30}\mathbf{1}o_{32}$
6 = **0110**	(7, B); (20, 2C); (33, 3F)	$o_1o_2\ldots o_8\mathbf{0}..o_{16}\mathbf{1}..o_{22}\mathbf{1}..o_{30}\mathbf{0}o_{32}$
9 = **1001**	(5, 9); (11, 1D); (35, 39)	$o_1o_2\ldots o_8\mathbf{1}..o_{16}\mathbf{0}..o_{22}\mathbf{0}..o_{30}\mathbf{1}o_{32}$
A = **1010**	(22, 2E); (30, 3C); (34, 38)	$o_1o_2\ldots o_8\mathbf{1}..o_{16}\mathbf{0}..o_{22}\mathbf{1}..o_{30}\mathbf{0}o_{32}$
B = **1011**	(23, 2F); (27, 2B)	$o_1o_2\ldots o_8\mathbf{1}..o_{16}\mathbf{0}..o_{22}\mathbf{1}..o_{30}\mathbf{1}o_{32}$
C = **1100**	(2, E); (25, 29); (32, 3E)	$o_1o_2\ldots o_8\mathbf{1}..o_{16}\mathbf{1}..o_{22}\mathbf{0}..o_{30}\mathbf{0}o_{32}$
D = **1101**	(1, D); (12, 1E); (36, 3A)	$o_1o_2\ldots o_8\mathbf{1}..o_{16}\mathbf{1}..o_{22}\mathbf{0}..o_{30}\mathbf{1}o_{32}$
E = **1110**	(3, F); (6, A); (13, 1F); (17, 1B); (21, 2D); (26, 2A); (37, 3B)	$o_1o_2\ldots o_8\mathbf{1}..o_{16}\mathbf{1}..o_{22}\mathbf{1}..o_{30}\mathbf{0}o_{32}$
F = **1111**	(4, 8)	$o_1o_2\ldots o_8\mathbf{1}..o_{16}\mathbf{1}..o_{22}\mathbf{1}..o_{30}\mathbf{1}o_{32}$

Bước 5: Xác định các giá trị của khóa của 1 *S-box*

Sau khi thực hiện thành công kết quả ở bước 4, kẻ tấn công sẽ tạo được một bảng các giá trị đầu vào tương ứng với vi sai đầu ra (xem 1 ví dụ ở bảng 5.2). Tiếp theo, kẻ tấn công tính các khả năng khóa. Kẻ tấn công làm như sau: Xác định các giá trị khóa có thể có ứng với các cặp (Si_{IX}, Si^*_{IX}). Công thức như sau:

$$Si_{KX} = Si_{IX} \oplus Si_{EX} \text{ hoặc } Si_{KX} = Si^*_{IX} \oplus Si_{EX} \ (Si_{EX} = \mathrm{E(Si)})$$

Lưu ý: cặp (Si_{IX}, Si^*_{IX}) đối ngẫu.

Từ công thức trên chúng ta có thể thấy để tính được Si_{KX} cần 2 đầu vào là Si_{EX} và (Si_{IX}, Si^*_{IX}).

Với thám mã vi sai, chúng ta sẽ có thể có trong tay các cặp bản rõ tùy chọn và các cặp bản mã tương ứng.Giả sử trong thực tế chúng ta các cặp bản rõ tùy chọn và chúng ta có các cặp bản mã tương ứng.Chúng ta có thể suy ra Si_{EX} một cách khá đơn giản từ bản rõ. Đối với (Si_{IX}, Si^*_{IX}) tuy chúng ta đã có 64 cặp tìm được ở bước 2 nhưng chúng ta không nhất thiết phải thử hết 64 cặp này. Chúng ta có thể rút ngắn số cặp (Si_{IX}, Si^*_{IX}) cần thử bằng cách suy ngược từ XOR của bản mã. Để suy ngược từ XOR của bản mã chúng ta cần chọn các cặp bản rõ thích hợp. DES với số vòng khác nhau thì sẽ có cách chọn các cặp bản rõ khác nhau.

Xét cụ thể với trường hợp DES 1 vòng. Chúng ta có thể suy ra các cặp (Si_{IX}, Si^*_{IX}) dựa vào xor đầu ra của s-box. Xor đầu ra của s-box có thể được suy ra từ giá trị xor sau khi đi qua P. Giá trị xor sau khi đi qua P có thể được suy ra từ XOR của bản mã. Để làm rõ hơn chúng ta sẽ cùng đi tới 1 số phép biến đổi:

- Giả sử có bản rõ X(L,R) và X*(L*,R*)
- Giả sử $L \oplus L^* = 0$, $R \oplus R^* = R'$
- $E(R) \oplus E(R^*) = E(R \oplus R^*) = E(R')$

- S(E(R) ⊕ K) = s1os2os3os4os5os6os7os8o = S
- S(E(R*) ⊕ K) = s1*os2*os3*os4*os5*os6*os7*os8*o = S*
- XOR sau khi qua S-box = S ⊕ S* = s1'os2'os3'os4'os5'os6'os7'os8'o
- XOR sau khi đi qua P: P' = P(S) ⊕ P(S*) = P (S ⊕ S*) = P(s1'os2'os3'os4'os5'os6'os7'os8'o)
- Bản mã của X (L,R) là (R, L ⊕ P(S))
- Bản mã của X* (L*,R*) là (R*,L* ⊕ P(S*))
- XOR của 2 bản mã: (R',P')

Với cặp bản rõ thỏa mã các điều kiện L ⊕ L* = 0, R ⊕ R* = R' thì chúng ta có XOR của bản mã tương ứng là (R',giá trị XOR sau khi qua hàm F). Vậy từ XOR của bản mã chúng ta có thể suy ra ngược lại giá trị XOR sau khi đi qua hàm F. Từ Xor sau khi đi qua hàm F có thể suy ra XOR đầu ra của bất kì hộp S-box nào.Cụ thể ở đây là xét với S-box1. Ta có: P' = P(s1'os2'os3'os4'os5'os6'os7'os8'o). Bản chất của phép hoán vị là đổi chỗ, nghĩa là XOR đầu ra của các hộp S-box sẽ được hoán vị tới 1 vị trí cố định trong đầu ra (4 bit của s1'o sẽ được hoán vị vào các vị trí 9,17,23,31) => Từ P' chúng ta có thể đi tới các vị trí 9, 17, 23, 31 để tìm các bit trong XOR đầu ra của S-box1. Từ XOR đầu ra của S-box1 chúng ta có thể suy ra các cặp (Si_{IX},Si^*_{IX}) dựa vào bảng ở bước 4. Sau khi đã có đầy đủ các đầu vào chúng ta sẽ có thể tìm được các khả năng của khóa

Để hiểu rõ hơn về quy trình thực hiện của kẻ tấn công, hãy xem các ví dụ bên dưới:

Ví dụ 3: Xét S_1 và giả thiết cặp đầu vào là $S1_E = 1_x, S1^*_E = 35_x$ và giá trị 6 bit khóa tương ứng là $S1_K = 23_x$. Giá trị đầu vào của *S1* là $S1_I = 22_x, S1^*_I = 16_x$ và đầu ra là $S1_O = 1_x, S1^*_O = C_x$. Giá trị XOR đầu ra là $S1'_O = D_x$.Giả thiết biết $S1_E = 1_x, S1^*_E = 35_x$ và $S1'_O = D_x$ và cần tìm $S1_K$. Giá trị XOR đầu vào bằng $S1'_E = S1'_I = 34_x$ không để đến giá trị của $S1_K$. Tra cứu bảng 5.1 thấy đầu vào của *S- box* có 8 khả năng. Tám khả năng này tạo ra 8 giá trị của khóa ($S_K = S_E \oplus S_I$) được liệt kê ở bảng 5.2. Mỗi dòng trong bảng mô tả 2 cặp với với 2 giá trị đầu vào giống nhau nhưng thứ tự được hoán đổi. Mỗi cặp có một giá trị khóa, nên mỗi dòng sẽ có 2 giá trị khóa (là $S_E \oplus S_I$ và $S_E \oplus S^*_I$). Giá trị đúng của $S1_K$ xuất hiện trong bảng này.

Bảng 5. 6. Các trường hợp khóa cho $34_x \to D_x$ bởi S_1 với đầu vào là $1_x, 35_x$

S box input	Possible Keys
06, 32	07, 33
10, 24	11, 25
16, 22	17, 23
1C, 28	1D, 29

Sử dụng thêm một cặp nữa sẽ có thêm những khả năng cho khóa $S1_K$. Xem xét cặp giá trị đầu vào $S1_E = 21_x, S1^*_E = 15_x$ (với giá trị $S1_K = 23_x$ tương tự như trường hợp trên). Giá trị đầu vào của *S-box* là $S1_I = 2_x, S1^*_I = 36_x$ và giá trị đầu ra là $S1_O = 4_x, S1^*_O = 7_x$. Giá trị XOR đầu ra là $S1'_O = 3_x$. Các khả năng đầu vào *S-box* với $34_x \rightarrow 3_x$ và các trường hợp khóa tương ứng được mô tả ở bảng 5.6. Khóa đúng phải xuất hiện ở cả 2 bảng 3 và 4. Thấy giá trị khóa chung của 2 bảng là 17_x và 23_x. Hai giá trị này không thể phân biệt với giá trị XOR đầu vào vì $17_x \oplus 23_x = 34_x = S1'_E$, nhưng có thể phân biệt bằng cách sử dụng một cặp với giá trị XOR đầu vào khác 34_x.

Bảng 5. 7. Các trường hợp khóa cho $34_x \rightarrow 3_x$ bởi S_1 với đầu vào là 2_x,36_x

S box input	Khóa có thể
01, 35	20, 14
02, 36	**23, 17**
15, 21	34, 0

Nhận thấy rằng, nếu vẫn chọn giá trị vi sai = 34_x thì sẽ chỉ thu hẹp được khả năng khóa là 23 hoặc 17. Để hiểu rõ hơn vấn đề này xét ví dụ 4 dưới đây.

Ví dụ 4: Giả sử biết $S1'_E = 0C_x$, $S1_E = 23_x$, $S1_{E^*} = 2F_x$, $S1_{O'} = E_x$

Bảng 5. 8. Các trường hợp khóa cho $0C_x$ -> E_x bởi S_1 với đầu vào là 23_x,$2F_x$

Đầu vào S-box	Khóa có thể
3, F	20, 2C
6, A	25, 29
13, 1F	30, 3C
17, 1B	34, 38
21, 2D	2, E
26, 2A	5, C
37, 3B	14, 18

Để giới hạn khóa chúng ta cần thêm cặp đầu vào. Giả sử biết $S1'_E = 0C_x$, $S1_E = 7_x$, $S1_{E^*} = 0B_x$, $S1_{O'} = E_x$

Bảng 5. 9. Các trường hợp khóa cho $0C_x$ -> E_x bởi S_1 với đầu vào là 07_x,$0B_x$

Đầu vào S-box	Khóa có thể
3, F	4, 8
6, A	1, D
13, 1F	14, 18
17, 1B	10, 1C
21, 2D	26, 2A
26, 2A	21, 2D
37, 3B	30, 3C

Từ bằng 5.9 nhận thấy rằng: có 2 khóa chung: 14_x , 18_x . Tuy nhiên, nếu vẫn chọn giá trị vi sai = $0C_x$ thì chúng ta sẽ không thể tìm được khóa duy nhất. Vì vậy chúng ta sẽ xét thêm 1 cặp có giá trị XOR đầu vào S-box1 khác $0C_x$. Giả sử biết $S1'_E = 34_x$, $S1_E = 35_x$, $S1^*_E = 1_x$, $S1'_{O=} 8_x$ (xem bảng 5.10)

Bảng 5. 10. Các trường hợp khóa cho 34_x -> 8_x bởi S_1 với đầu vào là 35_x,01_x

Đầu vào S-box	Khóa có thể
19, 2D	2C, 18
38, 0C	0D, 39
09, 3D	3C, 08

Sau khi xét kết hợp giữa 3 bảng 5.8, 5.9, 5.10 thì thấy khóa chung là 18_x => $S1_k = 18_x$

Bước 6: Tìm các bit của khóa trong các hộp *S-box* có thể mở rộng để tìm khóa con của hàm *F*

Như vậy, sau các bước ở trên, kẻ tấn công đã thành công trong việc tìm các bit khoa của một *S – box*. Tiếp theo, kẻ tấn công cần xác định xác suất để đầu vào có thể tạo ra đầu ra của hàm *F* ở vòng tiếp theo. Phương pháp tìm các bit của khóa trong các hộp *S-box* có thể mở rộng để tìm khóa con của hàm F bằng phương pháp sau đây:

1. Chọn một giá trị XOR của thông điệp thích hợp.

2. Tạo các cặp bản rõ với số lượng thích hợp với giá trị XOR đã được định trước. Mã hóa chúng và lưu lại các cặp bản mã tương ứng.

3. Với mỗi cặp rút ra các giá trị XOR đầu ra mong muốn trong các hộp S-box ở vòng cuối từ giá trị XOR bản rõ và các cặp bản mã.

4. Với mỗi giá trị khóa, đếm số cặp mà có giá trị XOR đầu ra bằng với giá trị XOR mong muốn trong vòng cuối.

5. Giá trị khóa đúng là khóa được đề xuất bởi tất cả các cặp.

Như vậy, sau sáu bước của thám mã vi sai, kẻ tấn công có khả năng tìm được khóa con của một vòng của giải thuật mã hóa DES. Để có thể tấn công tất cả các vòng của giải thuật mã hóa DES kẻ tấn công phải xác định được các bộ đặc trưng qua từng vòng. Trong các tài liệu [9, 11] đã trình bày chi tiết quy trình, kỹ thuật xác định các bộ đặc trưng qua từng vòng của giải thuật mã hóa DES.

Từ các ví dụ được phân tích ở trên, chúng ta xây dựng được bảng tổng hợp như bảng 5.11.

Bảng 5. 11. Tổng hợp kết quả

$S1'_E = S1_I'$	$S1_E$	$S1^*_E$	$S1_O$	$S1^*_O$	$S1'_O$	Khóa có thể $S1_k$)
34	1	35	1	C	D	7, 33, 11, 25, **17**, **23**, 1D, 29
34	21	15	4	7	3	20, 14, **23**, **17**, 34, 0
34	20	14	F	E	1	**23**, 2F, 3E, 3F, A, B, **17**, 1B
34	4	30	2	6	4	**17**, **23**
35	1	34	1	B	A	32, 7, 36, 3, **23**, **16**
35	2	37	38	31	9	**23**, **16**, 36, 3
35	16	23	3	E	D	**23**, **16**

Từ bảng 5.11 có thể nhận thấy rằng giá trị 23 được dùng chung nên chúng ta kết luận $S1_k$ là 23. Ta tìm được 1 bit của khóa con, tiếp tục lặp lại các bước trên ta thu được 6 bit đầu của khóa con $S1_k$.

5.3.3. Thám mã nội suy

Thám mã nội suy là phương pháp xây dựng một đa thức biểu diễn mối quan hệ giữa bản rõ và bản mã hay nói cách khác là xây dựng đa thức từ bản rõ và bản mã. Nếu như cả bộ mã được biểu diễn như 1 đa thức với bậc nhỏ thì hệ số của đa thức có thể tìm thấy từ các cặp bản rõ/bản mã sử dụng phương pháp nội suy Lagrange.

Kịch bản thám mã: như đã biết, thám mã nội suy sẽ tìm cách biểu diễn các bản mã bằng một đa thức của bản rõ. Nếu đa thức này có một số ít các hệ số chưa biết tương đối thấp, thì sau đó với một tập hợp các cặp *bản rõ/bản mã*, đa thức có thể được tái tạo. Với đa thức tái tạo lại kẻ tấn công sau đó có thể tính toán để tìm được một đại diện của bản mã từ đó biết được phương thức mã hóa làm giảm tính bảo mật của lược đồ mã hóa, mà không

cần có khóa bí mật. Cuộc tấn công nội suy cũng có thể được sử dụng để khôi phục khoá bí mật.

Quy trình thực hiện

Bước 1: **Xây dựng đa thức biểu diễn bản mã thông qua bản rõ**

Ký hiệu:

$x = (x_L, x_R)$: bản rõ với x_L là nửa trái và x_R là nửa phải.

$y = (y_L, y_R)$: tương tự kí hiệu cho bản mã.

$y^\tau = (y^\tau{}_L, y^\tau{}_R)$: là đầu ra của bộ mã khi bỏ đi vòng cuối cùng.

Xét công thức nội suy Lagrange:

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

Trong đó:

- R là một trường có $2n$ phần tử $x_1, ..., x_n, y_1, ..., y_n \in R$, với các x khác nhau đôi một.
- $f(x)$ là đa thức duy nhất trên trường R với bậc cao nhất là $n-1$ thỏa mãn $f(x_i) = y_i$ với $i = 1 \ldots n$.

Xét bộ mã trên với r vòng. Lợi dụng đặc điểm là phép XOR sử dụng trong bộ mã thì tương đương với phép cộng trong trường Z^2. Kết quả là bộ mã bao gồm những phép toán đại số cơ bản, vì thế mỗi nửa của bản mã y có thể được mô tả bằng một đa thức của bản rõ. Đa thức này có dạng như sau: $p(x_L, x_R) \in GF(2^{32})[x_L, x_R]$. Đa thức này có nhiều nhất là $3^{2r-1} + 3^r + 3^{r-1} + 1$ hệ số. Chú ý rằng bậc của x_R và x_L nhiều nhất lần lượt là 3^r và 3^{r-1}. Từ đây, kẻ tấn công có thể xây dựng đa thức này bằng $3^{2r-1} + 3^r + 3^{r-1} + 1$cặp bản rõ/ bản mã sử dụng nội suy Lagrange. Ví dụ với với $r=6$ kẻ tấn công cần nhiều nhất 2^{18} cặp bản rõ/ bản mã. Trong tài liệu [2, 11] đã trình bày một số định lý về độ phức tạp của tấn công nội suy.

Để hiểu rõ hơn về cách thức để xây dựng đa thức biểu diễn bản mã thông qua bản rõ bằng cách sử dụng phương thức tấn công lựa chọn bản rõ và công thức nội suy Lagrange, hãy cùng xem bảng 5.12.

Bảng 5. 12. Các bước tấn công nội suy

Bước 1: Kẻ tấn công	Kẻ tấn công gửi đi thông tin *x* cho hệ thống mã hóa. Thông tin *x* này được kẻ tấn công lựa chọn tùy ý theo phương pháp lựa chọn bản rõ.

Bước 2: Tại hệ thống mã hóa	Hệ thống nhận được thông tin *x* từ kẻ tấn công sau đó mã hóa thông tin *x* bằng 1 quy trình mã hóa lặp đi lặp lại: $c_i = (c_{i-1} + k_i)^3$ Với: c_0 là bản rõ (thông tin *x*); $k_i \in K$: khóa bí mật; *r*: số vòng lặp; c_r : bản mã sau *r* vòng; *c*: biểu thị bản mã cuối cùng. Xét mật mã 2 vòng [11]: Sau vòng 1 đầu ra có được: $c_1 = (x+k_1)^3 = (x^2+k_1^2)(x+k_1) = x^3 + k_1^2 x + x^2 k_1 + k_1^3$ Sau vòng 2 có được : $c_2 = c = (x+k_2)^3 = (x^3 + k_1^2 x + x^2 k_1 + k_1^3 + k_2)^3 =$ $x^9 + x^8 k_1 + x^6 k_2 + x^4 k_1^2 k_2 + x^3 k_2^2 + x^2(k_2 k_1^4 + k_1 k_2^2) + x(k_2^2 k_1^2 + k_1^8) + k_2^2 k_1^3 + k_1^9 + k_2^3$ Thể hiện bản mã như đa thức của bản rõ $p(x) = a_1 x^9 + a_2 x^8 + a_3 x^6 + a_4 x^4 + a_5 x^3 + a_6 x^2 + a_7 x + a_8$ Gửi trả kết quả về cho kẻ tấn công
Bước 3: Kẻ tấn công	Kẻ tấn công nhận được: $p(x) = a_1 x^9 + a_2 x^8 + a_3 x^6 + a_4 x^4 + a_5 x^3 + a_6 x^2 + a_7 x + a_8$ là một phương trình bậc 9 có 8 hệ số (a_1,…,a_8). Theo [2, 11] sẽ có với *n=8* sẽ chọn 8 cặp rõ/mã tùy ý *(x,p(x))*. Sau đó xây dựng đa thức tái tạo đại diện cho phép mã hóa bằng công thức Lagrange: L(x) = $\sum_{i=1}^{8} p(x_i) \prod_{\substack{1 \le j \le 8 \\ i \ne j}} \frac{x-x_j}{x_i-x_j}$ Áp dụng và có được: *L(x)* = [p(x_1)($\frac{x-x_2}{x_1-x_2}$)($\frac{x-x_3}{x_1-x_3}$)…($\frac{x-x_8}{x_1-x_8}$)] + [p(x_2)($\frac{x-x_1}{x_2-x_1}$)($\frac{x-x_3}{x_2-x_3}$)…($\frac{x-x_8}{x_2-x_8}$)] + … + [p(x_8)($\frac{x-x_1}{x_8-x_1}$)($\frac{x-x_2}{x_8-x_2}$)…($\frac{x-x_7}{x_8-x_7}$)] (*) ⇨ Đa thức tái tạo: đa thức đại diện cho phép mã hóa, ở đây khi triển khai đa thức (*) trên thu được một phương trình bậc 7 với các hệ số được xác định. ⇨ Có được biểu diễn *p(x)* của phép mã hóa mà không cần có thông tin về khóa bí mật *K*.

Bước 2: **Xây dựng không gian *bản rõ/ bản mã* từ đa thức thu được**

Mục đích của bước này chính là: từ kết quả đạt được của bước 1, kẻ tấn công sẽ thực hiện xây dựng một đồ thị chứa tập hợp các điểm (*bản rõ/ bản mã*).

Như vậy, qua ví dụ thể hiện ở bảng 5.12 có thể nhận thấy rằng: kẻ tấn công có thể biểu diễn được bản mã bằng một đa thức của bản rõ. Ở ví dụ trên thì đa thức tìm được là đa thức (*). Từ đa thức này, có thể biểu diễn được tập hợp các điểm (các cặp *bản rõ/ bản mã*) nằm trên đồ thị của đa thức đó. Từ đó, kẻ tấn công sẽ giới hạn được không gian *bản rõ/bản mã* và rút ngắn được thời gian tìm kiếm vì kẻ tấn công biết được các cặp *bản rõ/bản mã* chỉ nằm trên đồ thị của đa thức vừa xây dựng được.

Bước 3: **Đoán khóa vòng cuối**:

Để đoán khóa vòng cuối kẻ tấn công sử dụng kỹ thuật tấn công Meet-in-the-middle. Tấn công meet-in-the-middle là cuộc tấn công làm suy yếu hệ thống bảo mật bằng cách lưu trữ giá trị trung gian từ mã hóa hoặc giải mã và sử dụng chúng để giảm thiểu thời gian tìm khóa.

Mục đích: tìm ra giá trị khóa của vòng cuối. Cách thức thực hiện như sau*:*

- Xét bộ lặp với r vòng: Giả sử kẻ tấn công dự đoán được khóa của vòng cuối: k_{r-1}. Với k_{r-1} đã dự đoán và bản rõ đã chọn sẽ thu được y^τ.
- Gọi: z là đầu ra của vòng s với $s \le r-1$.

Như vậy: z được biểu diễn thông qua một hàm đa thức $h(y^\tau) \in GF(2^m)[y^\tau]$ của đầu ra y^τ với m là độ dài khối, đa thức $h(y^\tau) \in GF(2^m)[y^\tau]$ có bậc là d_h.

Mặt khác z có thể biểu diễn thông qua bản rõ x như một đa thức $g(x) \in GF(2^m)[x]$, với m là độ dài khối, đa thức $h(y^\tau) \in GF(2^m)[y^\tau]$ có bậc là d_g.

Từ những biện luận trên, kẻ tấn công sẽ xây dựng phương trình sau: $g(x) = h(y^\tau)$. Phương trình này có nhiều nhất $d_{gh}+2$ ẩn, với $d_{gh} = d_g + d_h$. Phương trình có thể giải với phép nhân và phép cộng cả g và h với hằng số. Chính vì vậy, để đảm bảo chỉ thu được nghiệm duy nhất, kẻ tấn công thể đặt hệ số của bậc cao nhất bằng 1 và hằng số bằng 0 (khi đó luôn luôn tìm ra được một nghiệm). Sau đó, giải phương trình bằng d_{gh} cặp *bản rõ/bản mã* kẻ tấn công kiểm tra xem một cặp *bản rõ/bản mã* bất kỳ (x, y^τ) có thỏa mãn $g(x) = h(y^\tau)$ không. Nếu đúng, có thể cho rằng đã đoán đúng giá trị khóa của vòng cuối.

5.4. Một số kỹ thuật thám mã hệ mật mã bất đối xứng

Các giải thuật mã hóa bất đối xứng nói chung đều có các điểm yếu sẽ bị khai thác nếu không có những lựa chọn hợp lý các tham số đầu vào. Trong thực tế, từ khi ra đời đến nay, có rất nhiều phương pháp và kỹ thuật đã được xử dụng để tấn công lên các giải thuật mã hóa bất đối xứng. Trong các tài liệu [1, 2, 4, 5] đã trình bày chi tiết về cơ sở toán học công một số kỹ thuật tấn công lên giải thuật mã hóa bất đối xứng. Tiếp theo, để giúp sinh

viên hiểu rõ hơn về quy trình, các bước tiến hành của một số kỹ thuật thám mã bất đối xứng, bài giảng sẽ trình bày 3 kỹ thuật và phương pháp tấn công phổ biến hiện nay.

5.4.1. Thám mã dựa vào các sơ hở toán học

5.4.1.1. Tấn công module chung

Với mục đích giảm chi phí sinh khóa cho Trung tâm xác thực, người quản lý thường dùng chung một module *N* cho tất cả người xử dụng, mỗi người sử dụng có khóa bí mật riêng. Đây là điểm yếu mà người thám mã có thể lợi dụng để lấy được các thông điệp quan trọng mà trung tâm xác thực muốn gửi đến một số người dùng khác. Tấn công modulo dùng chung là kỹ thuật tấn công chỉ biết bản mã.

Mục đích của kỹ thuật tấn công module dùng chung:

- Tìm được khóa bí mật (phá vỡ toàn bộ hệ thống)
- Tìm được bản rõ

Tiếp theo, bài giảng sẽ trình bày về cả 2 cách thức tiến hành đạt được hai mục đích trên.

Cách thức tấn công đạt được mục đích 1: Phá vỡ toàn bộ hệ thống

Phương pháp: Sử dụng khóa công khai và bí mật của mình để sinh ra khóa bí mật của người khác. Nghĩa là căn cứ vào khóa công khai của người bị tấn công, kẻ tấn công có thể tìm được khóa bí mật của người bị tấn công mà không cần biết chính xác các tham số khác $\Phi(n)$ với *N* là modulo chung.

Cách thức thực hiện: Để tìm được khóa bí mật này, cần phải tìm một số sao cho thỏa mãn 2 điều kiện:

- là bội của $\Phi(n)$ (ĐK1).

- Nguyên tố cùng nhau với khóa công khai (ĐK2).

Mô tả yêu cầu**:**

Đầu vào: kẻ tấn công có cặp khóa công khai/bí mật của kẻ tấn công (e_2, d_2), khóa công khai của người bị tấn công e_1.

Đầu ra: cần tìm khóa bí mật của người bị tấn công d_1

Các bước tiến hành như sau:

Bước 1: Đặt $t = e_2.d_2 - 1$

Bước 2: Kẻ tấn công tiếp tục thực hiện ĐK2. Sử dụng thuật toán Euclid mở rộng để tìm $f = UCLN(t, e1)$ có cặp số (r, s) thỏa mãn công thức $r*t + s*e_1 = f$.

Bước 3: Nếu $f = 1$ đặt $d_1' = s$ và trả về kết quả d_1'.

Bước 4: Nếu f khác 1 thì đặt $t = \frac{t}{f}$ rồi quay lại bước 2.

Với việc thực hiện các bước trên, kẻ tấn công hoàn toàn có thể tìm được khóa bí mật của người bị tấn công mà không mất quá nhiều thời gian và các bước tính toán. Cách thức thực hiện và tính đúng đắn của phương pháp này đã được trình bày trong tài liệu [2]. Để hiểu rõ hơn về kỹ thuật này, hãy xem ví dụ dưới đây:

Ví dụ về tấn công module dùng chung để tìm khóa bí mật người dùng

Gọi khóa $d_1 = 17$, bản rõ $M = 25$ gửi đi (2 yếu tố cần tìm)

Kẻ tấn công có:

$n = 253$ (số module dùng chung)

$e_1 = 13$ (khóa công khai của người bị tấn công),

$e_2 = 23$ (khóa công khai của kẻ tấn công)

$d_2 = 67$ (khóa bí mật của kẻ tấn công),

Bản mã $C = 27$

Kẻ tấn công cần tìm khóa d_1'

Cách thức kẻ tấn công tiến hành thám mã như sau:

Bước 1: Đặt $t = e_2 * d_2 - 1 = 23*67 - 1 = 1540$

Bước 2: Có $UCLN\ (t, e_1)$ = UCLN (1540,13) = 1

Sử dụng giải thuật Euclid mở rộng tìm được $r = -2$ và $s = 237$. Từ đây tìm được $d_1' = 237$.

Có $M = C^{d_1'} mod\ n = 27^{237} \bmod 253 = 25$

Như vậy, khóa bí mật mới d_1' không bằng khóa bí mật ban đầu d_1 nhưng nó vẫn giải mã chính xác (do d_1' vẫn thỏa mãn điều kiện $d_1'.e_1 \equiv 1 \bmod (\Phi(n))$.

Cách thức tấn công đạt được mục đích 2: Tấn công trong trường hợp khóa công khai của kẻ tấn công và người bị tấn công là các số nguyên tố cùng nhau:

Phương pháp: Nếu một thông điệp được gửi tới nhiều người dùng có các khóa công khai e nguyên tố cùng nhau thì kẻ tấn công có thể giải mã được thông điệp mà không cần tìm khóa bí mật. Trong trường hợp này, kẻ tấn công phải tìm được một cặp khóa công khai nguyên tố cùng nhau. Do các khóa được sinh ra ngẫu nhiên bởi hệ thống nên điều này hoàn toàn có thể xảy ra.

Ý tưởng: Thông điệp M được gửi tới hai người có khóa công khai tương ứng là e_1 và e_2. Hai khóa này nguyên tố cùng nhau. Hai người lần lượt nhận được bản mã:

$C_1 = M^{e1} \bmod n$ (Công thức tạo bản mã của *RSA*)

$C_2 = M^{e2} \bmod n$

Trong đó $ƯCLN\ (e_1, e_2) = 1$

Dựa vào C_1, C_2, e_1, e_2 thì kẻ tấn công có thể giải mã văn bản mà không cần biết đến các khóa bí mật d_1, d_2.

Thuật toán:

Đầu vào: kẻ tấn công có : module chung n; các khóa công khai e_1, e_2; các bản mã C_1, C_2.

Đầu ra: kẻ tấn công muốn tìm Thông điệp M.

Bước 1: Tính $r_1 = e_1^{-1} \, mod \;\; e_2$

Bước 2: Tính $r_2 = \frac{r_1.e_1 - 1}{e_2}$

Bước 3: Tính $M = \; C_1^{r_1}.\left(C_2^{r_2}\right)^{-1} mod \; n$

Công thức tính M ở trong bước 3 đúng bởi vì:

Có $C_1^{r_1}.\left(C_2^{r_2}\right)^{-1} mod \; n = \; M^{r_1.e_1}.(M^{r_2.e_2})^{-1} mod \; n$

Mặt khác:

$$C_1 = M^{e1} \, mod \; n) = \; M^{r_1.e_1}.\left(M^{-\frac{r_1.e_1-1}{e_2}.e_2}\right) mod \; n = \; M^{r_1.e_1}.(M^{1-r_1.e_1}) \; mod \; n$$

$$= \; M^{r_1.e_1+1-r_1.e_1} \; mod \; n = \; M^1 \; mod \; n = \; M \; mod \; n = \; M$$

Để hiểu rõ hơn về kỹ thuật tấn công module dùng chung để tìm được bản rõ, hãy xem ví dụ dưới đây:

Ví dụ về tấn công module dùng chung để tìm bản rõ:

Thông điệp được gửi đi M = 30 (yếu tố cần tìm)

Kẻ tấn công có: n = 247, e_1= 5, e_2= 7, C_1 = 140, C_2 = 30

Cách thức thực hiện:

Bước 1: Tìm giá trị r_1

Có $r_1 = e_1^{-1} \, mod \;\; e_2 = \; 5^{-1} \; \text{mod} \;\; 7 = 3$

Bước 2: Tìm giá trị r_2

$$r_2 = \frac{r_1.e_1 - 1}{e_2} = \frac{3 * 5 - 1}{7} = 2$$

Bước 3: Tìm bản rõ M:

$$M = \; C_1^{r_1}.\left(C_2^{r_2}\right)^{-1} mod \; n = \; 140^3.(30^2)^{-1} \; \text{mod} \; 247$$

$$= [140^3 \; \text{mod} \; 247].[(30^2)^{-1} \text{mod} \; 247] \; \text{mod} \; 247 = 77.87 \; \text{mod} \; 247$$

$$= 6699 \; \text{mod} \; 247 = 30$$

Như vậy, kẻ tấn công mặc dù không có khóa giải mã nhưng vẫn tìm ra được bản rõ M= 30.

5.4.1.2. Tấn công với khóa bí mật nhỏ (Tấn công Wiener)

Như đã biết, để giảm thời gian giải mã một trong những cách được dùng đó là sử dụng giá trị *d* nhỏ. M. Wiener đã chỉ ra rằng với *d* nhỏ hệ mật có thể bị phá vỡ như sau:

Định lý 1 (M. Wiener): Cho $n = p*q$ với $q < p < 2q$. Giả sử $d < x = \frac{\sqrt[4]{n}}{3}$ cho trước (n, e) với $ed = 1 \bmod (\varphi(n))$, có thể tìm được *d* hiệu quả.

Ví dụ, nếu *n* có kích thước là 1024 bit thì *d* được chọn ít nhất là 256 bit thì mới có thể tránh được tấn công Wiener.

Mục đích của kỹ thuật tấn công Wiener: tìm được khóa bí mật.

Cách thức tiến hành: kẻ tấn công áp dụng định lý Wiener và phân số liên tục.

Đầu vào: kẻ tấn công có khóa công khai *e, n.*

Cần tìm: kẻ tấn công muốn tìm ra khóa bí mật *d.*

Các bước tiến hành như sau:

- Bước 1: Tìm phân số liên tục của $\frac{e}{n}$
- Bước 2: Liệt kê các phân số hội tụ của $\frac{e}{n}$
- Bước 3: Gán $\frac{k}{d}$ là phân số hội tụ đầu tiên
- Bước 4: Tính $\Phi(n) = \frac{ed-1}{k}$
- Bước 5: Nếu $\Phi(n)$ nguyên thì in ra *d* và kết thúc
- Bước 6: Nếu $\Phi(n)$ không nguyên thì gán $\frac{k}{d}$= phân số hội tụ tiếp theo rồi quay về bước 4.

Trên đây, bài giảng đã trình bày các bước thực hiện thuật toán Wiener để tìm ra khóa bí mật, để làm rõ hơn bài giảng sẽ trình bày ví dụ cụ thể:

Ví dụ: $(n, e) = (90581, 17993)$. Tìm d= ?

Theo định lí Weiner:$d <= \frac{\sqrt[4]{n}}{3}. \approx 5.7828$

Biểu diễn liên phân số của $\frac{e}{n}$ có dạng:

$$\frac{e}{n} = \frac{17993}{90581} = 0 + \cfrac{1}{5 + \cfrac{1}{29 + \cfrac{1}{4 + \cfrac{1}{1 + \cfrac{1}{3 + \cfrac{1}{2 + \cfrac{1}{4 + \frac{1}{3}}}}}}}} = [0, 5, 29, 4, 1, 3, 2, 4, 3]$$

Danh sách dãy hội tụ từ phân số trên như sau:

Phân số hội tụ thứ nhất: 0

Phân số hội tụ thứ hai: $\frac{1}{5}$

Phân số hội tụ thứ ba: $0 + \frac{1}{5+\frac{1}{29}} = \frac{29}{146}$

Phân số hội tụ thứ tư: $0 + \frac{1}{5+\frac{1}{29+\frac{1}{4}}} = \frac{117}{589}$

……

$$\Rightarrow \frac{k}{d} = 0, \frac{1}{5}, \frac{29}{146}, \frac{117}{589}, \frac{146}{735}, \frac{555}{2794}, \frac{1256}{6323}, \frac{5579}{28086}, \frac{17993}{90581}$$

* Có $ed \equiv 1 \bmod \Phi(n) \Rightarrow ed = k.\Phi(n) + 1 \Rightarrow \Phi(n) = \frac{ed-1}{k}$. Thử lần lượt các giá trị $\frac{k}{d}$ để tìm $\Phi(n)$ nguyên.

+ Với $\frac{k}{d} = 0$ thì không thể sinh ra các yếu tố của *n*.

+ Với $\frac{k}{d} = \frac{1}{5}$ thì $\Phi(n) = \frac{ed-1}{k} = \frac{17993.5-1}{1} = 89964$ (Theo chứng minh của Wiener, $\frac{k}{d}$ là phân số tối giản, vì vậy khi $\frac{k}{d} = \frac{1}{5} \Rightarrow k = 1, d = 5$).

Từ kết quả trên nhận thấy rằng: chỉ có $d = 5$ cho $\Phi(n)$ giá trị nguyên. Vậy khóa bí mật cần tìm là $d = 5$.

5.4.1.3. Tấn công với số mũ công khai nhỏ

Tấn công số mũ công khai nhỏ là dạng tấn công chỉ biết bản mã, dùng trong trường hợp khóa công khai nhỏ. Nếu người gửi mã hóa một thông điệp *M* bằng cùng một khóa công khai nhỏ và gửi cho nhiều người thì có thể tìm *M* mà không cần biết các khóa bí mật.

Mục đích của kỹ thuật tấn công này là tìm ra bản rõ *M*

Các tiến hành như sau:

Bước 1: Kẻ tấn công biết được người gửi sẽ gửi một bản mã cho nhiều người khác nhau và biết được khóa công khai, đồng thời bắt được toàn bộ các bản mã.

Bước 2: Sử dụng định lý số dư Trung Hoa để tìm được bản rõ M^n chung từ các bản mã.

Bước 3: Tính ra bản rõ gốc: $M = (M^n)^{-n}$.

Ví dụ: người gửi dùng khóa công khai $e = 3$ mã hóa thông điệp M rồi gửi cho 3 người khác. Khi đó, kẻ tấn công sẽ có các bản mã tương ứng:

$C_1 = M^3 \bmod n_1$

$C_2 = M^3 \bmod n_2$

$C_3 = M^3 \bmod n_3$

Từ các bản mã $C_1; C_2; C_3$ mà kẻ tấn công có thể tìm ra được bản rõ M mà không cần biết khóa bí mật.

Quy trình giải mã như sau:

Nếu n_1, n_2, n_3 là nguyên tố cùng nhau, thì áp dụng định lý phần dư Trung Hoa sẽ tính được M^3 duy nhất thỏa mãn điều kiện. Lấy căn bậc 3 thông thường của M^3 sẽ tìm được M. Kẻ tấn công có thể mò ra M mà không cần biết các khóa bí mật.

Nếu ngược lại n_1 và n_2 không là nguyên tố cùng nhau thì *ƯCLN* (n_1, n_2) sẽ là p hoặc q chung của hai người. Khi đó người này có thể phá vỡ được hệ mật của người kia.

Ví dụ về kỹ thuật tấn công lợi dụng số mũ công khai nhỏ:

Có $C_1 = 2, C_2 = 3, C_3 = 5, n_1 = 3, n_2 = 5, n_3 = 7$. Tìm M

Giải quyết :

Đặt $M^3 = x$

$$\begin{cases} x \equiv 2 (mod\, 3) \\ x \equiv 3 (mod\, 5) \\ x \equiv 5 (mod\, 7) \end{cases}$$

Có:

$$Z = 3.5.7 = 105; Z_1 = 5.7 = 35, Z_2 = 3.7 = 21, Z_3 = 3.5 = 15.$$

$$y_1 = 35^{-1} (\mathrm{mod}\, 3) = 2^{-1} (\mathrm{mod}\, 3) = 2;$$

$$y_2 = 21^{-1} (\mathrm{mod}\, 5) = 1^{-1} (\mathrm{mod}\, 5) = 1;$$

$$y_3 = 15^{-1} (\mathrm{mod}\, 7) = 1^{-1} (\mathrm{mod}\, 7) = 1.$$

Từ đó tính được:

$$x \equiv 2 \times 35 \times 2 + 3 \times 21 \times 1 + 5 \times 15 \times 1 (\mathrm{mod}\, 105)$$

$$x \equiv 140 + 63 + 75 (\mathrm{mod}\, 105) \equiv 278 (\mathrm{mod}\, 105)$$

$$x \equiv 68 (\mathrm{mod}\, 105).$$

$=> M^3 \equiv 68 \bmod 105 => M^3 = 68 + k.105$

5.4.1.4. Các tham số p-1 và q-1 có các ước nguyên tố nhỏ

Trong khi xây dựng hệ RSA, nếu bất cẩn trong việc chọn các tham số *p* và *q* dẫn đến *p-1* hoặc *q-1* có các ước nguyên tố nhỏ thì hệ mật trở nên mất an toàn. Khi đó, kẻ tấn công có thể dùng thuật toán Pollar để phân tích *N* một cách hiệu quả và tìm được các thám số *p, q.* Các yêu cầu đối với kỹ thuật này như sau: *module N* và một cận *B.*

Cách thức thực hiện như sau*:*

- Bước 1: Chọn một cận *B* bất kì

-Bước 2:

$$\text{Tính } M = \prod_{\substack{\text{số nguyên} \\ \text{tố } q \le B}} q^{\lfloor log_q n \rfloor}$$

q là tập các số nguyên tố và *q*< = *B*

- Bước 3: Chọn ngẫu nhiên một số *a* sao cho *a* là số nguyên tố cùng nhau với *n*

- Bước 4: Tính $g = ƯCLN(a^M - 1, n)$

- Bước 5:

+ Nếu $1 < g < n$ thì nhận *g*, lúc này $p = g$ và $q = \frac{n}{p}$

+ Nếu $g = 1$, chọn 1 số *B* khác lớn hơn.

+ Nếu $g = n$, chọn 1 số *B* khác nhỏ hơn.

Ví dụ: về phương pháp tấn công lợi dụng vào *p-1* và *q-1* có các ước nguyên tố nhỏ như sau:

Gọi $p= 13, q = 23$ (yếu tố cần tìm)

Cách thức để kẻ tấn công tìm ra 2 số *p, q* như sau*:*

Bướ 1: Chọn $B = 5$

Bước 2: Tính $M = 2^8 * 3^5 * 5^3$

(*q* lúc này = 2, 3, 5 là các số nguyên tố thỏa mãn điều kiện $q \le 5$)

- Có $n = 299$, chọn $a = 2$

- $g = gcd\ (a^M - 1, n) = 13$

Do $1 < 13 < 299$, thì trả về $g = 13$

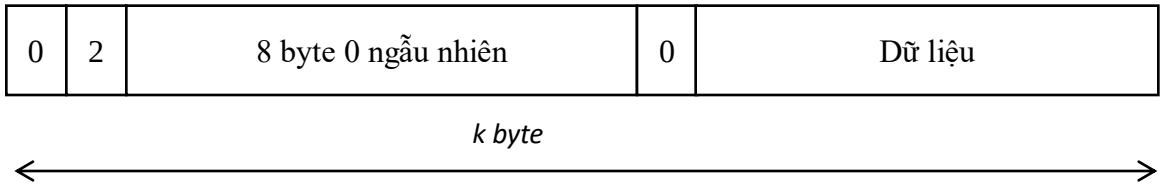
=> $p = 13, q = \frac{n}{p} = 23$

5.4.2. Thám mã lựa chọn bản mã thích nghi

5.4.2.1. Giới thiệu về chuẩn PKCS

Trước khi thực hiện mã hóa, phải thực hiện việc chuyển đổi văn bản rõ (chuyển đổi từ *M* sang *m*) sao cho không có giá trị nào của *M* tạo ra văn bản mã không an toàn.

Trong chuẩn PKCS#1 phiên bản 1.5, trước khi mã hóa, thông điệp *M* được đệm thông báo vào trước nó thành *m = PKCS(M) = 00 02 R1 R2 … Rs 00 M.*



Trong đó:

- *k* là số byte của *n*
- byte đầu tiên có giá trị 0, byte thứ hai có giá trị 2
- R_i là các byte ngẫu nhiên khác 0
- *s = k – 3 – số byte của data* và s ≥ 8 để đáp ứng yêu cầu về an toàn
- byte ngay sau R_s cũng có giá trị 0
- Cuối cùng là thông điệp *M*.

Khi đó bản mã được tính là $C = m^e \bmod N$. Khi giải mã, đầu tiên tính $m = C^d \bmod N$ rồi sau đó kiểm tra xem *m* có tuân theo chuẩn PKCS hay không tức là *m* có dạng *00 02 R1 R2 … Rs 00 M* hay không.

Nếu chuẩn PKCS#1 phiên bản 1.5 được cài đặt cùng với hàm tiên tri (Oracle Functions) thì mỗi khi giải mã xong hàm tiên tri này sẽ kiểm tra bản rõ *PKCS(m)* và gửi phúc đáp ngược trở lại nơi yêu cầu giải mã là bản rõ *m* này có tuân theo chuẩn PKCS hay không. Chính phúc đáp này giúp cho kẻ tấn công tấn công tính ra được bản rõ trên một bản mã mà mình muốn giải mã mà không cần phải biết khóa giải mã. Hàm tiên tri (Oracle Functions) – một phần mềm dùng để kiểm tra xem bản rõ có tuân theo PKCS #1 v1.5 hay không.

5.4.2.2. Cách thức tấn công lựa chọn thích nghi bản mã

Loại tấn công này không cần phải biết khóa bí mật mà vẫn giải mã được các thông điệp dựa vào việc khai thác các kết quả trả về của hàm tiên tri. Trong bài giảng này, tác giả sẽ trình bày cách thức tấn công lựa chọn thích nghi bản mã để tấn công lên giải thuật mã hóa RSA khi giải thuật mã hóa RSA sử dụng tiêu chuẩn chuyển đổi bản rõ PKCS #1 v1.5. Đầu vào của hàm tiên tri là bản mã *C*, đầu ra là kết quả kiểm tra bản rõ *M*.

Hàm tiên tri hoạt động như sau: nhận bản mã *C*, giải mã *C* thu được *M*, kiểm tra *M* có tuân theo PKCS hay không, trả về kết quả kiểm tra cho người yêu cầu. Nghĩa là người yêu cầu chỉ có được kết quả kiểm tra PKCS của bản rõ chứ không có được bản rõ. Kẻ tấn công có thể gửi nhiều bản mã tùy thích đến các hàm tiên tri và nhận về kết quả kiểm tra xem các bản rõ được giải mã có tuân theo PKCS hay không và căn cứ vào các kết quả này để tìm ra bản rõ tương ứng với một bản mã mong muốn giải mã.

Kịch bản của cuộc tấn công như sau: Giả sử kẻ tấn công muốn tìm bản rõ *M* từ bản mã $C = M^e \bmod n$. Bản rõ *M* trong trường hợp này không nhất thiết phải tuân theo PKCS. Tuy nhiên, trong thực tế để tấn công được hiệu quả, thì bản rõ *M* này phải được phân tích về dạng tuân theo PKCS.

Bước 1: kẻ tấn công kiểm tra xem *M* có tuân theo PKCS không (bằng cách gửi *C* đến hàm tiên tri, giải mã được *M*, hàm tiên tri trả về kết quả kiểm tra PKCS của *M*). Lúc này có 2 khả năng xảy ra:

Khả năng 1: Nếu *M* tuân theo PKCS thì $m = M$

Khả năng 2: Nếu *M* không tuân theo PKCS, thì kẻ tấn công chọn 1 bản rõ *m* tuân theo PKCS sao cho từ *m* kẻ tấn công tính được *M*.

Bước 2: tìm được *m* thuộc 1 miền xác định

$$m \in M_0 = [2B, 3B - 1]$$

Trong đó:

M_0 là miền giá trị chứa *m*, $B = 2^{8(k-2)}$

k là là độ dài của module *n* tính theo byte.

Lúc này, kẻ tấn công chưa tìm được *m* cụ thể vì vậy chưa tính được *M*. Kẻ tấn công sẽ tìm cách thu hẹp miền giá trị chứa *m*. Để làm được điều đó, kẻ tấn công chọn số s_i ngẫu nhiên sao cho: Nếu $m*s_i$ tuân theo PKCS, khi s_i tăng lên thì miền giá trị của *m* sẽ thu hẹp lại. Công thức tổng quát như sau:

$$\frac{2B+r_i.n}{s_i} \leq m \leq \frac{3B-1+r_i.n}{s_i} \qquad \text{với } r_i = \left\lfloor \frac{m.s_i}{n} \right\rfloor$$

Trong thực tế để thu hẹp miền chứa *m*, kẻ tấn công phải chọn các s_i tăng dần, sao cho $m*s_i$ tuân theo PKCS. Kết quả cuối cùng là $m \in M_i = [a,a]$ (miền chỉ có 1 giá trị *a*).

Tiếp theo, để hiểu rõ hơn về kỹ thuật này, bài giảng sẽ trình bày về chi tiết về cách thức tiến hành thám mã.

Gọi *C* là bản mã mà kẻ tấn công thu được.

Gọi hàm *ktra_PKCS(a)* là hàm kiểm tra xem bản rõ *b* của bản mã *a* có tuân theo PKCS không.

Kết quả hàm *ktra_PKCS (a)* là hàm do hàm tiên tri gửi về.

$$ktra\_PKCS\ (a) = \begin{cases} 1\ khi\ b\ tuân\ theo\ PKCS \\ 0\ khi\ b\ không\ tuân\ theo\ PKCS \end{cases}$$

với *a* là bản mã, *b* là bản rõ của *a*.

Gọi M_i là miền giá trị chứa *m* (với $i \geq 0, i \in Z$)

***Cách thức thực hiện**:*

- **<u>Bước 1</u>**: Phân tích bản rõ *M* về dạng tuân theo PKCS.

+ Nếu *ktra_PKCS (C) = 0* thì

- Chọn s_0 một cách ngẫu nhiên (đặt $m = s_0.M \ mod \ n \Rightarrow c = m^e \ mod \ n = M^e(s_0)^e \ mod \ n = C.(s_0)^e \ mod \ n$).
- Gửi bản mã $c = C.(s_0)^e \ mod \ n$ đến hàm tiên tri.
- Nếu *ktra_PKCS* $(c) = 0$ thì tiếp tục chọn s_0 mới khác giá trị đã chọn.

+ Nếu *ktra_PKCS (C) = 1* thì *m = M, c = C.*

Sau bước 1, thu được bản mã của bản rõ tuân theo PKCS, bản rõ này có quan hệ với bản rõ ban đầu.

- **<u>Bước 2</u>**: Khởi tạo miền giá trị chứa *m*

Do *m* luôn tuân theo PKCS (kết quả của bước 1) nên có

- $m \in M_0 = [2B, 3B - 1]$ với $B = 2^{8(k-2)}$ và *k* là là độ dài của module n tính theo byte.

Để thu nhỏ miền M_0, tiến hành chọn tham số s_1 trong bước 3.

<u>Bước 3</u>: Bắt đầu tìm kiếm bản mã mới phù hợp với PKCS

- Tìm s_1 nhỏ nhất có thể với điều kiện $s_1 \geq \frac{n}{3B}$ sao cho

ktra_PKCS($c.(s_1)^e \ \text{mod } n$) = 1.

- Tiến hành thu hẹp miền M_i

<u>Bước 4</u>: Tìm kiếm bản mã mới phù hợp với PKCS, sử dụng tham số s_i để thu nhỏ các miền $[a, b]$ chứa *m*, sử dụng tham số *r* để băm $[a, b]$ thành các đoạn không giao nhau có độ dài nhỏ hơn.

- Nếu M_{i-1} chứa ít nhất 2 đoạn thì tìm s_i nhỏ nhất có thể với điều kiện $s_i \geq s_{i-1}$ sao cho *ktra_PKCS*($c.(s_i)^e \ \text{mod } n$) = 1.
- Nếu M_{i-1} chứa đúng một đoạn, tức là $M_{i-1} = [a, b]$ thì chọn các giá trị nhỏ r_i và s_i sao cho: $r_i \geq 2\frac{b.s_{i-1}-2B}{n}$ và $\frac{2B+r_i.n}{b} \leq s_i \leq \frac{3B+r_i.n}{a}$ và sao cho *ktra_PKCS*($c.(s_i)^e \ \text{mod } n$) = 1.

<u>Bước 5</u>: Thu hẹp miền giá trị chứa *m*

$$M_i = \bigcup_{(a,b,r)} \left\{ \left[max\left(a, \left\lceil \frac{2B + r.n}{s_i} \right\rceil\right), min\left(b, \left\lfloor \frac{3B - 1 + r.n}{s_i} \right\rfloor\right)\right]\right\}$$

Đối với tất cả đoạn $[a, b] \in M_{i-1}$ và $\frac{a.s_i - 3B + 1}{n} \leq r \leq \frac{b.s_i - 2B}{n}$

Nếu với 1 đoạn $[a,b] \in M_{i-1}$ mà $max\left(a, \left\lceil \frac{2B+r.n}{s_i} \right\rceil\right) \geq min\left(b, \left\lfloor \frac{3B-1+r.n}{s_i} \right\rfloor\right)$ thì loại luôn đoạn $[a,b]$ này (vì đầu mút trái lớn hơn đầu mút phải).

Bước 6: Sau bước 5, miền *Mi* đã được thu hẹp lại và chia nhỏ dựa vào tham số r, s_i. Lúc này *Mi* có thể chỉ còn 1 giá trị, kẻ tấn công cần kiểm tra điều này.

+ Nếu M_i chứa đúng một phần tử, tức là $M_i = [a, a]$

- $m = a$
- Nếu *ktra_PKCS (C) = 0* thì $M = m.(s_0)^{-1} \ mod \ n$
- Nếu *ktra_PKCS (C) = 1* thì $M = m$
- Trả về giá trị *M* và kết thúc.

+ Nếu M_i chứa nhiều hơn 1 phần tử, đặt $i = i + 1$ và quay về bước 4 để tiếp tục thu hẹp và chia nhỏ miền M_i.

Trong kỹ thuật tấn công lựa chọn bản mã thích nghi có điểm cần lưu ý là tại sao $m \in M_0 = [2B, 3B - 1]$. Để giúp sinh viên hiểu rõ hơn về vấn đề này, bài giảng sẽ tiến hành chứng minh công thức trên.

Do *m* tuân theo PKCS nên hai byte đầu tiên của *m* là 00 và 02. Để thuận tiện, đặt: $B = 2^{8(k-2)}$ với *k* là là độ dài của module *n* tính theo byte

=> *k - 2* là kích thước của module *n* trừ đi 2 byte đầu tiên 00 và 02.

Khi đó $B = 01\underbrace{00\ 00\ \dots\ 00\ 00}_{k-2 \text{ byte toàn 0}}$

Do *m* tuân theo PKCS nên giá trị *m* phải nằm giữa *00 02 00 00… 00* (tức *2B*) và *00 02 FF FF … FF* (tức *3B – 1* do *3B = 00 03 00 00 .. 00*). Như vậy $m \in [2B, 3B - 1]$

Kết luận: Trong thuật toán này nhận thấy rằng tham số s_i sẽ điều khiển quá trình nhốt m_0 vào các khoảng $[a, b]$. Đặc biệt độ lớn của các khoảng $[a, b]$ này sẽ giảm dần. Tựu chung là m_0 chỉ có thể thuộc về một trong các khoảng đóng thuộc tập hợp M_i đối với tất cả các bước trung gian i. Số lượng các khoảng đóng trong tập hợp M_i sẽ giảm dần do xuất hiện các khoảng trống. Một khi M_i chỉ chứa đúng một khoảng đóng thì độ dài của khoảng này giảm đi một nửa sau mỗi bước. Điều kiện để thuật toán nhanh thành công là nhờ kết quả của $2B \leq ms \ mod \ n < 3B$ cho phép nhanh chóng tìm được bản mã tuân theo PKCS khi tìm các s_i trung gian. Ngoài ra thuật toán còn nhanh thành công khi số lượng các khoảng đóng và độ dài của chúng giảm nhanh theo chiều hướng sau mỗi bước lại giảm đi một nửa. Cuối cùng là dẫn đến tập hợp $M_k = [m_0, m_0]$ với duy nhất một khoảng đóng có độ dài bằng 0. Trên thực tế về mặt trung bình thì cứ sau 2^{20} bản mã lựa chọn cần thiết đưa vào hàm tiên tri sẽ giúp cho kẻ tấn công khám phá ra được hoàn toàn m_0 và từ đó tính ra *m*. Chính vì vậy mà tấn công này còn được gọi là "Tấn công triệu thông báo". Gần đây vào năm 2012 các chuyên gia đã hiện thực hóa tấn công trên khi làm giảm số lượng bản mã lựa chọn cần thiết đưa vào hàm tiên tri chỉ còn dưới 50 ngàn bản mã. Kỹ thuật tấn công này đặc biệt hiệu quả đối với các thiết bị SmartCard kiểu USB Token và HSM có hỗ trợ chức năng nhập khẩu

khóa bí mật có lập mã. Tấn công giải mã bản mã RSA kích thước 1024 bit theo chuẩn PKCS#1 phiên bản 1.5 chỉ tốn vài chục phút cho đến vài giờ.

Để phòng tránh kỹ thuật tấn công này thì cần phải thực hiện một số thay đổi sau: 1) sẽ không đưa ra câu phúc đáp ngay hoặc không đưa ra thông báo. Như vậy dẫn đến việc kiểm tra điều kiện tuân theo PKCS sẽ không bình thường và gặp khó khan. 2) áp dụng một số mô hình chuyển đổi an toàn hơn như chuyển đổi mã hóa bất đối xứng tối ưu (Optimal Asymmetric Encryption Padding).

5.5. Câu hỏi ôn tập

Câu 1. Hãy trình bày khái niện về thám mã?

Câu 2. Hãy trình bày một số kỹ thuât thám mã?

Cấu 3. Hãy trình bày về quy trình thám mã?

Cấu 4. Hãy trình bày về phương pháp thám mã vi sai?

Câu 5. Hãy trình bày về phương pháp thám mã tuyến tính?

Câu 6. Hãy trình bày về phương pháp thám mã nội suy?

Câu 7. Hãy trình bày về một số kỹ thuật thám mã dựa trên cơ sở toán học của các giải thuật mã hóa khóa bất đối xứng?

Câu 8. Hãy trình bày về thám mã lựa chọn bản mã thích nghi?

Câu 9. Hãy trình bày về thám mã kênh bên trên giải thuật mã hóa khóa bất đối xứng ?

TÀI LIỆU THAM KHẢO

[1] Nguyễn Bình, Ngô Đức Thiện. *Cơ sở mật mã học*. Học Viện Công Nghệ Bưu Chính Viễn Thông, 2013. 237 trang.

[2] Sunil Gupta. *Introduction to cryptography and network security*. ISBN-10: 819069197X 2015, 784p.

[3] William Stallings, *Cryptography and Network Security 7th Edition*, Prentice Hall, 2021.

[4] Jonathan Katz; Yehuda Lindell, *Introduction to Modern Cryptography 2nd Edition*, Chapman & Hall/ CRC, 2017.

[5] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C 2nd Edition*, Wiley, 2015.

[6] Edward Schaefer. *An introduction to cryptography and cryptanalysis*. Santa Clara University, 2009, 120p.

[7] Eli Biham, Adi Shamir, *Diffirential Cryptanalysis of DES-like Cryptosystems.*

[8] Mitsuru Matsui. *Linear Cryptanalysis of DES-Cipher*.

[9] Thomas Jakobsen, Lars R. Knudsen. *The Interpolation Attack on Block Ciphers*.

[10] Mark Stamp, Richard M. Low. *Applied Cryptanalysis: Breaking Ciphers in the Real World*. April 2007. 424 pages. ISBN: 978-0-470-11486-5.

[11] Dan Boneh (1999), *Twenty Years of Attacks on the RSA cryptosystem*