**BỘ TÀI NGUYÊN VÀ MÔI TRƯỜNG**

**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM**

**KHOA: HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**

****

**ĐỒ ÁN MÔN HỌC**

**MẠNG MÁY TÍNH CƠ BẢN**

**NGHIÊN CỨU VÀ ỨNG DỤNG MÃ HÓA**

Giảng viên hướng dẫn: **Ths. Từ Thanh Trí**

Sinh viên thực hiện: **Nguyễn Hoàng Tân**

**Đặng Thành Đạt**

**Phạm Quang Hùng**

Lớp : **CNTT\_01**

Khóa : **04**

***TP. Hồ Chí Minh, tháng 2 năm 2017***

**MỞ ĐẦU**

Trong nhưng năm gần đây, mạng Internet đã trở thành nền tảng chính cho sự trao đổi thông tin trên toàn cầu. Có thể thấy một cách rõ ràng là Internet đã và đang tác động lên nhiều mặt của đời sống chúng ta từ việc tìm kiếm thông tin, trao đổi dữ liệu đến việc hoạt động thương mại, học tập nghiên cứu và làm việc trực tuyến,… Nhờ Internet mà việc trao đổi thông tin cũng ngày càng tiện lợi, nhanh chóng hơn, khái niệm như thư điện tử (email) cũng không còn mấy xa lạ với mọi người.

Tuy nhiên trên môi trường truyền thông này, ngoài mặt tích cực Internet cũng tiềm ẩn những tiêu cực của nó đối với vấn đề bảo vệ thông tin.

Do đó, những yêu cầu được đặt ra đối với việc trao đổi thông tin trên mạng:

○ Bảo mật tuyệt đối thông tin trong giao dịch.

○ Đảm bảo tính toàn vẹn của thông tin.

○ Chứng thực được tính đúng đắn về pháp lí của thực thể tham gia trao đổi thông tin.

○ Đảm bảo thực thể không thể phủ nhận hay chối bỏ trách nhiệm của họ về những hoạt động giao dịch trên Internet.

Mã hóa thông tin là một ngành quan trọng và có nhiều ứng dụng trong đời sống xã hội. Ngày nay các ứng dụng mã hóa và bảo mật thông tin đang được sử dụng ngày càng phổ biến hơn trong các lĩnh vực khác nhau trên Thế giới, từ các lĩnh vực an ninh, quân sự, quốc phòng,… Cho đến các lĩnh vực dân sự như thương mại điện tử, ngân hang,…

Trong đề tài này chúng em tìm hiểu một số phương pháp mã hóa đang được sử dụng rộng rãi hiện nay và ứng dụng của nó trong đời sống thực tế.

**LỜI CẢM ƠN**

Chúng em xin gửi lời cảm ơn chân thành tới quý thầy cô giáo khoa Hệ Thống Thông Tin – Viễn Thám, trường Đại Học Tài Nguyên và Môi Trường nói chung và thầy cô giáo trong bộ môn Công Nghê Thông Tin nói riêng, đã giúp đỡ, hỗ trợ và tạo điều kiện cho chúng em trong suốt quá trình làm đề tài.

Chúng em xin chân thành cảm ơn thầy giáo ThS. Từ Thanh Trí, người đã giúp đỡ, chỉ bảo, hướng dẫn tận tình cho chúng em trong quá trình làm đề tài. Trong thời gian làm việc với thầy, chúng em không những học được nhiều kiến thức bổ ích về các phương pháp mã hóa và tầm quan trọng trong mã hóa dữ liệu ngày nay mà còn học được tinh thần làm việc, thái độ nghiên cứu khoa học ngiêm túc và đáng khâm phục của thầy.

Mặc dù chúng em đã cố gắng hoàn thành đề tài với tất cả nổ lực của mỗi người nhưng chắc chắn sẽ không tránh khỏi thiếu sót. Em kính mong nhận được sự thong cảm và tận tình chỉ bảo của thầy và các bạn.

Một lần nữa, chúng em xin chân thành cảm ơn và luôn mong nhận được sự đóng góp quý báu của tất cả mọi người.

**NHẬN XÉT**

(Của giáo viên, nếu có)

………………………………………………………………………………………….………………………………………………………………………………………….………………………………………….…………………………………………………………………………………………………………………………………………..…………………………………………………………………………………….…….…………………………………………………………………………………………..…………………………………………………………………………………………..…………………………………….…………………………………………………….…………………………………………………………………………………………..……………………………………………………………………………….………….…………………………………………………………………………………………..…………………………………………………………………………………………..……………………………….………………………………………………………….…………………………………………………………………………………………...………………………………………………………………………….………………..…………………………………………………………………………………………...…………………………………………………………………………………………...………………………….………………………………………………………………..…………………………………………………………………………………………...…………………………………………………………………………………………...

**MỤC LỤC**

Nội dung Trang

[CHƯƠNG I: TỔNG QUAN VỀ HỆ MẬT MÃ 1](#_Toc479093958)

[1.1 Khái niệm về mã hóa thông tin 1](#_Toc479093959)

[1.1.1 Khái niệm 1](#_Toc479093960)

[1.1.2 Vai trò của mã hóa 1](#_Toc479093961)

[1.1.3 Các thành phần của hệ mã hóa 1](#_Toc479093962)

[1.2 Tiêu chuẩn để đánh giá hệ mã hóa 3](#_Toc479093963)

[1.2.1 Độ an toàn của thuật toán 3](#_Toc479093964)

[1.2.2 Tốc độ mã hóa và giải mã 3](#_Toc479093965)

[1.2.3 Phân phối khóa 3](#_Toc479093966)

[1.3 Khóa 3](#_Toc479093967)

[1.3.1 Khái niệm 3](#_Toc479093968)

[1.3.2 Ví dụ 4](#_Toc479093969)

[1.4 Phân loại các thuật toán mã hóa 4](#_Toc479093970)

[1.4.1 Phân loại theo các phương pháp 4](#_Toc479093971)

[1.4.1.1 Mã hóa cổ điển (Classical Cryptography) 4](#_Toc479093972)

[1.4.1.2 Mã hóa đối xứng (Symetric Cryptography) 5](#_Toc479093973)

[1.4.1.3 Mã hóa bất đối xứng (Asymmetric Cryptography) 5](#_Toc479093974)

[1.4.1.4 Mã hóa hàm băm (Cryptographic Hash Function) 7](#_Toc479093975)

[1.4.2 Phân loại theo số lượng khóa 8](#_Toc479093976)

[1.4.2.1 Mã hóa khóa bí mật 8](#_Toc479093977)

[1.4.2.2 Mã hóa khóa công khai 9](#_Toc479093978)

[CHƯƠNG 2: MỘT SỐ PHƯƠNG PHÁP MÃ HÓA CỔ ĐIỂN 10](#_Toc479093979)

[2.1 Hệ mã hóa thay thế 10](#_Toc479093980)

[2.1.1 Hệ mã hóa CAESAR 11](#_Toc479093981)

[2.1.2 Hệ mã hóa Vigenere 12](#_Toc479093982)

[2.2 Hệ mã hóa hoán vị 14](#_Toc479093983)

[2.2.1 Đảo ngược toàn bộ bản rõ 14](#_Toc479093984)

[2.2.2 Mã hóa theo hình mẫu học 14](#_Toc479093985)

[2.2.3 Đổi chỗ cột 14](#_Toc479093986)

[2.2.4 Hoán vị các ký tự của bản rõ theo chu kì cố định 15](#_Toc479093987)

[CHƯƠNG 3: MỘT SỐ THUẬT TOÁN MÃ HÓA HIỆN ĐẠI 15](#_Toc479093988)

[3.1 Thuật toán mã hóa DES 15](#_Toc479093989)

[3.1.1 Lịch sử ra đời 16](#_Toc479093990)

[3.1.2 Mô tả thuật toán DES 16](#_Toc479093991)

[3.1.2.1 Sơ đồ tổng quát 16](#_Toc479093992)

[3.1.2.2 Quá trình tạo khóa 18](#_Toc479093993)

[3.1.2.3 Hoán vị khởi đầu 22](#_Toc479093994)

[3.1.2.4 Quá trình mã hóa 23](#_Toc479093995)

[3.1.2.5 Hoán vị cuối cùng 28](#_Toc479093996)

[3.1.2.6 Quá trình giãi mã 30](#_Toc479093997)

[3.1.4 Độ an toàn của thuật toán DES 30](#_Toc479093998)

[3.1.5 Mô hình ứng dụng 31](#_Toc479093999)

[3.2 Thuật toán mã hóa RSA 31](#_Toc479094000)

[3.2.1 Khái quát về RSA 31](#_Toc479094001)

[3.2.2 Mô tả thuật toán 31](#_Toc479094002)

[3.2.2.1 Tạo khóa 32](#_Toc479094003)

[3.2.2.2 Mã hóa và giải mã 32](#_Toc479094004)

[3.2.2.3 Sơ đồ thuật toán 33](#_Toc479094005)

[3.2.2.4 Ví dụ minh họa 33](#_Toc479094006)

[3.2.3 Một số phương pháp tấn công 35](#_Toc479094007)

[3.2.3.1 Phương pháp sử dụng (n) 35](#_Toc479094008)

[3.2.3.2 Áp dụng thuật toán phân tích ra thừa số 35](#_Toc479094009)

[3.2.3.3 Bẻ khóa dựa trên tấn công lặp lại 36](#_Toc479094010)

[3.2.4 Đánh giá thuật toán 36](#_Toc479094011)

[3.2.5 Mô hình ứng dụng 37](#_Toc479094012)

[3.2.5.1 Ứng dụng 37](#_Toc479094013)

[3.2.5.2 Thực nghiệm 38](#_Toc479094014)

[3.3 Thuật toán mã hóa MD5 46](#_Toc479094015)

[3.3.1 Hàm băm MD5 46](#_Toc479094016)

[3.3.1.1 Khái niệm 46](#_Toc479094017)

[3.3.1.2 Ứng dụng 46](#_Toc479094018)

[3.3.1.3 Thuật giải 47](#_Toc479094020)

[3.3.2 MD5 48](#_Toc479094021)

[3.3.2.1 Mô tả thuật toán 48](#_Toc479094022)

[3.3.2.2 Thuật toán 48](#_Toc479094023)

[3.3.3 Mô hình ứng dụng 51](#_Toc479094024)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 52](#_Toc479094025)

[4.1 Kết quả đạt được 52](#_Toc479094026)

[4.2 Hướng phát triển 53](#_Toc479094027)

[4.3 Kết luận 53](#_Toc479094028)

**PHỤ LỤC**

**DANH MỤC TÀI LIỆU THAM KHẢO**

[1] Phan Đình Diệu, Lý thuyết mật mã và an toàn thông tin, Đại học quốc gia Hà Nội, 1999.

[2] Dương Anh Đức, Trần Minh Triết, Mã hóa và ứng dụng, Khoa Công nghệ Thông tin, Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia thành phố Hồ Chí Minh, 2005.

[3] Alfred Menezes, Comparing the Security of ECC and RSA, University of Waterloo, 2000.

[4] E. Biham, A. Shamir, Diffrential cryptanalisis of DES-like cryptosystems, Jounal of Cryptology, 1991.

[5] R. Rivest, The MD5 Message-Digest Alorithm, MIT Laboratory for Computer Science and RSA Data Security, Inc, April 1992.

[6] M. Matsui, Linear cryptanalisis method for DES Cipher, Advances in cryptology, proceedings Eurocrypt, LNCS, T. Helleseth, Ed. Springer-Verlag, 1994.

[7] Hồ Thuần, Giáo trình “ Lý thuyết mật mã và an toàn dữ liệu”, Trường Đại học Bách Khoa Hà Nội, 2000.

[8] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu, Conllisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, International Association for Cryptologic Research, 2004.

**DANH MỤC HÌNH VẼ**

*Hình 1.1: Mô hình mã hóa……………………………………………………………………..2*

*Hình 1.2: Mô hình mã hóa đối xứng………………………………………………………….5*

*Hình 1.3: Mô hình mã hóa bất đối xứng……………………………………………………..6*

*Hình1.4 : Mô hình mã hóa hàm băm…………………………………………………………8*

*Hình 1.5: Mô hình mã hóa bí mật…………………………………………………………….8*

*Hình 1.6: Mật mã hóa khóa công khai……………………………………………………….9*

*Hình 1.7: Dùng khóa công khai để mã hóa, nhưng dùng khóa bí mật để giải mã…..10*

*Hình 2.1: Vòng tròn caesar…………………………………………………………………..11*

*Hình 2.2: Hình vuông VIGERENE được sử dụng để mã hóa và giải mã……………...13*

*Hình 3.1: Sơ đồ tổng quát mã hóa DES………………………………………………….…17*

*Hình 3.2: Sơ đồ tạo khóa……………………………………………………………………..18*

*Hình 3.3: Bảng hoán vị PC1…………………………………………………………………19*

*Hình 3.4: Bảng hoán vị PC2…………………………………………………………………22*

*Hình 3.5: Biểu diễn dãy 64 bit x chia thành 2 thành phần L0R0……………………..…23*

*Hình 3.6: Bảng hoán vị khởi đầu IP..……………………………………………………….23*

*Hình 3.7: Bảng E-box……………………………………………………………………..…..25*

*Hình 3.8: Bảng S1-box………………………………………………………………………...27*

*Hình 3.9: Bảng P-box…………………………………………………………………………28*

*Hình 3.10: Bảng IP-1 ………………………………………………………………………….29*

*Hình 3.11: Sơ đồ thuật toán RSA…………………………………………………………….33*

*Hình 3.12: Mô hình tạo chữ ký số từ thuật toán RSA để mã hóa thông tin…………...37*

*Hình 3.13: Mô hình quy trình giải mã của chữ ký số RSA……………………………….38*

*Hình 3.14: Mô hình dùng hàm băm xác nhận tính toàn vẹn của dữ liệu.……………..39*

*Hình 3.15: Mô hình sử dụng hàm băm để mã hóa mật khẩu.…………………………....40*

*Hình 3.16: Sơ đồ vòng lặp chính của MD5………………………………….………….….43*

*Hình 3.17: Sử dụng md5 checker để kiểm tra tệp tin……………………………………..44*

**KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT**

DES : Data Encryption Standard.

FIPS: Federal Information Processing Standard (Tiêu chuẩn xử lý thông tin liên bang).

AES : Advanced Encryption Standard.

NSA : National Security Agency (Cơ quan bảo mật quốc gia Hoa Kỳ).

RSA : Rivest, Shamir, Adleman.

MD : Message Digest.

MIT : [Massachusetts Institute of Technology](http://web.mit.edu/) (Học viện công nghệ Massachusetts).

# CHƯƠNG I: TỔNG QUAN VỀ HỆ MẬT MÃ

## 1.1 Khái niệm về mã hóa thông tin

### 1.1.1 Khái niệm

Mã hóa thông tin là chuyển đổi thông tin từ dạng rõ (dạng đọc được) sang dạng mã (dạng không thể đọc được) và ngược lại. Nhằm mục đích ngăn chặn nguy cơ truy cập thông tin truyền đi trên mạng một cách bất hợp pháp. Thông tin sẽ được truyền đi trên mạng dưới dạng mã và không thể đọc được với bất kỳ ai cố tình muốn lấy thông tin đó.

Khi chúng ta có nhu cầu trao đổi thông tin, thì internet là môi trường không an toàn, đầy rủi ro và nguy hiểm, không có gì đảm bảo rằng thông tin mà chúng ta truyền đi không bị đọc trộm trên đường truyền. Vì vậy mã hóa là biện pháp giúp ta bảo vệ chính mình cũng như thông tin mà ta gửi đi.

Ngoài ra mã hóa còn đảm bảo tính toàn vẹn của dữ liệu.

### 1.1.2 Vai trò của mã hóa

Các hệ mã hóa phải thực hiện được các vai trò sau.

- Các hệ mã hóa phải che giấu được nội dung của các văn bản rõ (Plain Text) để đảm bảo sao cho chỉ người chủ hợp pháp của thông tin mới có quyền truy cập thông tin, hay nói cách khác là chống truy cập không đúng quyền hạn.

- Tạo các yếu tố xác thực thông tin, đảm bảo thông tin lưu hành trên hệ thống đến người nhận hợp pháp xác thực.

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

Ưu điểm lớn nhất của các hệ mã hóa là có thể đánh giá được độ phức tạp của tính toán mà “kẻ địch” phải giải quyết bài toán để lấy được thông tin của dữ liệu. Tuy nhiên mỗi hệ mã hóa đều có một số ưu và nhược điểm khác nhau, nhưng nhờ đánh giá được độ phức tạp tính toán, mức độ an toàn của mỗi hệ mã hóa mà ta có thể ứng dụng cụ thể tùy theo yêu cầu về độ an toàn.

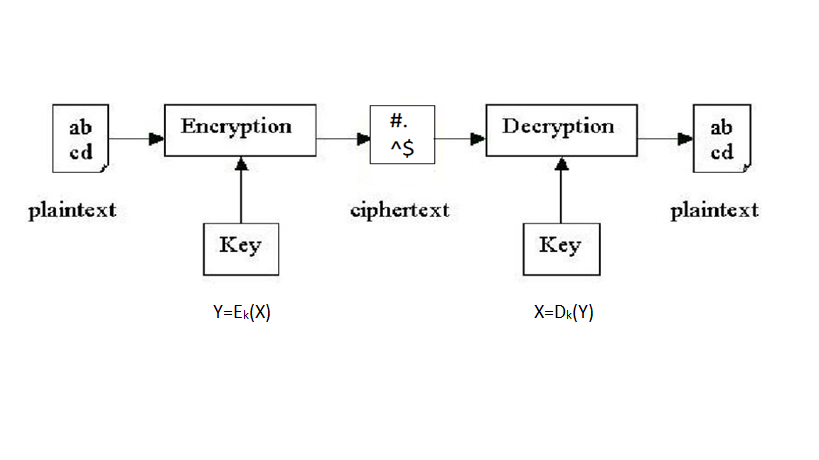
### 1.1.3 Các thành phần của hệ mã hóa

Một hệ mã hóa là một bộ 5 (P, C, D, K, E) thỏa mãn các điều kiện sau.

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

- C là tập hợp hữu hạn các bản mã (CipherText), nó còn dược gọi là không gian bản mã. Mỗi phần tử của C có thể nhận được bằng cách áp dụng phép mã hóa EK lên một phần tử của P.

- K là tập hợp hữu hạn các khóa hay còn gọi là không gian khóa. Đối với mỗi phần tử k của K được gọi là một khóa (Key). Số lượng của không gian khóa phải đủ lớn để “kẻ địch” không đủ thời gian để thử mọi khóa (phương pháp vét cạn).

- E và D lần lượt là tập luật mã hóa và giải mã. Với mỗi k của K có một quy tắc mã hóa EK: P→C và một quy tắc giải mã tương ứng Dk є D. Mỗi Ek: P→C và Dk: C→P là những hàm mà: Dk(Ek(x))=x với mọi bản rõ x є P.

*Hình 1.1: Mô hình mã hóa.*

Bên cạnh việc làm thế nào để che giấu nội dung thông tin thì mã hóa phải đảm bảo các mục tiêu sau:

**○** Tính bí mật (Confidentiality): Đảm bảo dữ liệu truyền đi một cách an toàn và không thể bị lộ thông tin nếu như có ai đó cố tình muốn có được nội dung của dữ liệu gốc ban đầu. Chỉ những người được phép mới có khả năng đọc được nội dung thông tin ban đầu.

*○* Tính xác thực (Authentication): Giúp cho người nhận dữ liệu xác định được chắc chắn dữ liệu họ nhận được là dữ liệu gốc ban đầu. Kẻ giả mạo không thể có khả năng để giả dạng một người khác hay nói cách khác không thể mạo danh để gửi dữ liệu. Người nhận có khả năng kiểm tra nguồn gốc thông tin mà họ nhận được.

**○** Tính toàn vẹn (Integrity): Giúp cho người nhận dữ liệu kiểm tra được rằng dữ liệu không bị thay đổi trong quá trình truyền đi. Kẻ giả mạo không thể có khả năng thay thế dữ liệu ban đầu bằng dữ liệu giả mạo.

**○** Tính không thể chối bỏ (Non-repudation): Người gửi hay người nhận không thể chối bỏ sau khi đã gửi và nhận thông tin.

## 1.2 Tiêu chuẩn để đánh giá hệ mã hóa

### 1.2.1 Độ an toàn của thuật toán

Nguyên tắc đầu tiên trong mã hóa là “Thuật toán nào cũng có thể bị phá vỡ”. Các thuật toán khác nhau cung cấp mức độ an toàn khác nhau, phụ thuộc vào độ phức tạp để phá vỡ chúng. Tại một thời điểm, độ an toàn của một thuật toán phụ thuộc:

○ Nếu chi phí hay phí tổn cần thiết để phá vỡ một thuật toán lớp hơn giá trị của thông tin đã mã hóa thuật toán thì thuật toán đó tạm thời được coi là an toàn.

○ Nếu thời gian cần thiết để phá vỡ một thuật toán là quá lâu thì thuật toán đó tạm thời được coi là an toàn.

○ Nếu lượng dữ liệu cần thiết để phá vỡ một thuật toán quá lớn so với lượng dữ liệu đã được mã hóa thì thuật toán đó tạm thời được coi là an toàn.

### 1.2.2 Tốc độ mã hóa và giải mã

Khi đánh giá hệ mã hóa phải chú ý đến tốc độ mã hóa và giải mã. Hệ mã hóa tốt thì thời gian mã hóa và giải mã nhanh.

### 1.2.3 Phân phối khóa

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

## 1.3 Khóa

### 1.3.1 Khái niệm

Một khóa mã hóa là một phần thông tin đặc biệt được kết hợp với một thuật toán để thi hành mã hóa và giải mã. Mỗi khóa khác nhau có thể tạo ra các văn bản mã hóa khác nhau, nếu không chọn đúng khóa thì không thể mở được tài liệu đã mã hóa trên, cho dù biết được thuật toán trên dung thuật toán mã hóa gì, sử dụng khóa càng phức tạp thì độ an toàn của dữ liệu càng lớn.

### 1.3.2 Ví dụ

Mã hóa nội dung của một bức thư với khóa là “Thay thế mỗi kí tự xuất hiện trong bức thư bằng ký tự đứng thứ 3 sau nó”. Cùng thuật toán trên nhưng sử dụng khóa là “Thay thế mỗi ký tự xuất hiện trong bức thư bằng ký tự đứng thứ 4 sau nó”. Như vậy kết quả của một bức thư có nội dung như nhau sau khi sử dụng 2 khóa khác nhau sẽ có 2 bản mã khác nhau.

## 1.4 Phân loại các thuật toán mã hóa

### 1.4.1 Phân loại theo các phương pháp

#### 1.4.1.1 Mã hóa cổ điển (Classical Cryptography)

Xuất hiện trong lịch sử, thuật toán không dùng khóa, dễ hiểu. Từ chính các phương pháp mã hóa này đã giúp chúng ta tiếp cận với các thuật toán mã hóa đối xứng được sử dụng ngày nay.

Trong mã hóa cổ điển có 2 phương pháp nổi bật đó là:

- Mã hóa thay thế (Substitution Cipher): Là phương pháp mà từng ký tự (hay từng nhóm ký tự) trong bản rõ được thay thế bằng một ký tự (hay nhóm ký tự) khác nhau tạo nên bản mã. Bên nhận chỉ cần đảo ngược lại trình tự thay thế trên thì sẽ nhận được bản rõ ban đầu.

- Mã hóa hoán vị (Transposition Cipher) hay còn gọi là mã hóa dịch chuyển: Trong phương pháp mã hóa hoán vị, các ký tự bản rõ vẫn được giữ nguyên, chúng chỉ được sắp xếp lại vị trí để tạo ra bản mã. Tức là các ký tự trong bản rõ hoàn toàn không bị thay đổi bằng các ký tự khác.

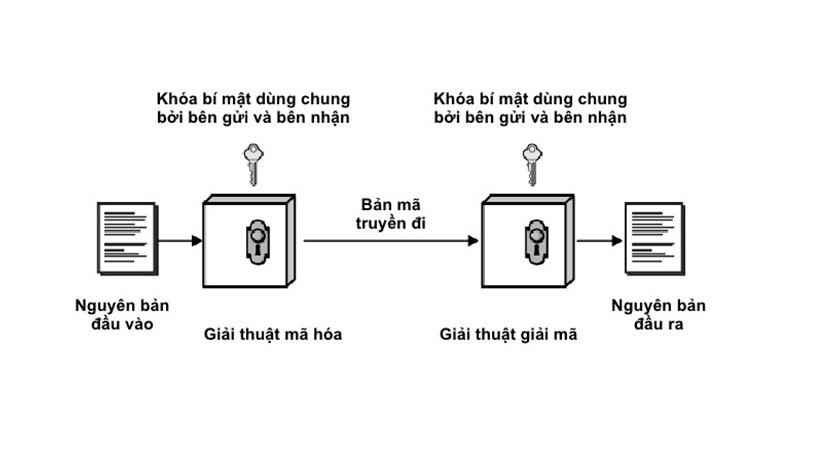
Ví Dụ: Nếu chúng ta dùng thuật toán đổi kí tự trong câu văn thành kí tự liền kề trong bảng chữ cái thì chữ “Tinh tế” sẽ biến thành “Ujoi uf”. Người nhận khi nhận được chữ “Ujoi uf” thì chỉ việc dịch ngược lại là xong.

Tuy nhiên, giải pháp mã hóa này được xem là không an toàn, vì nếu một người thứ ba biết được thuật toán thì xem như thông tin không còn bảo mật nữa. Việc giữ bí mật thuật toán trở nên rất quan trọng, và không phải ai cũng có thể giữ bí mật đó một cách trọn vẹn. Có khả năng người đó sẽ rò rỉ ra, hoặc có ai đó ngồi giải ra thuật toán thì xem như chúng ta thua cuộc.

Ngoài ra còn một số phương pháp: Hệ mã hóa Ceasar, Vigenere, Hill,…

#### 1.4.1.2 Mã hóa đối xứng (Symetric Cryptography)

Mã hóa đối xứng hay mã hóa chia sẻ khóa là mô hình mã hóa 2 chiều, có nghĩa là tiến trình mã hóa và giải mã đều dùng chung một khóa. Khóa này được chuyển giao bí mật giữa hai đối tượng tham gia giao tiếp. Mã hóa đối xứng thực hiện nhanh nhưng có thể gặp rủi ro nếu khóa bị đánh cắp.

*Hình 1.2: Mô hình mã hóa đối xứng.*

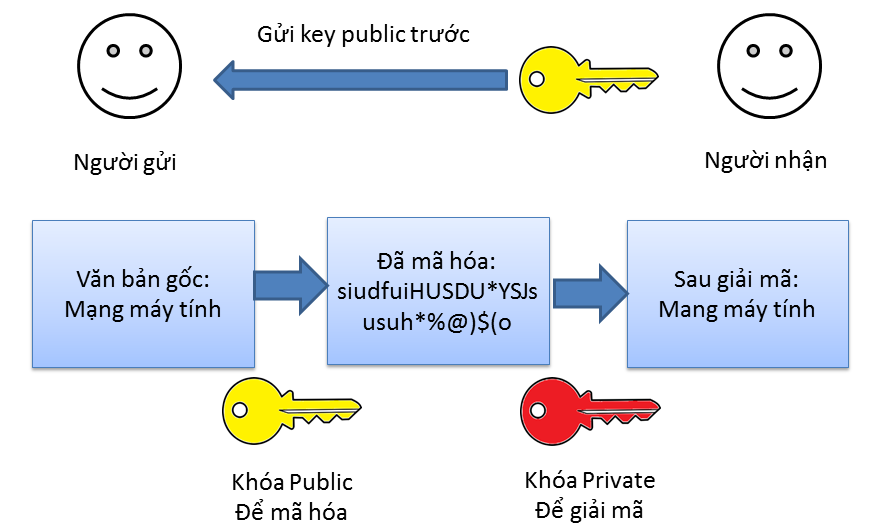
Ví dụ: Jack muốn gửi cho Jonh một tập tin trên một kênh công cộng nào đó và không muốn người khác có thể đọc thông tin, thì Jack sẽ cài đặt password cho tập tin của mình và bí mật chuyển password qua cho Jonh. Những người khác không thể xem tập tin nếu họ không biết password của Jack, cũng giống như mô hình mã hóa đối xứng cả Jack và Jonh đều sử dụng chung một khóa là password của Jack. Nhược điểm lớn nhất của mô hình này là làm sao có thể chuyển được khóa an toàn. Nếu khóa này bị lộ ra thì bất kì ai cũng có thể sử dụng để giải mã tập tin, như vậy thì tính bảo mật sẽ không còn nữa.

Một số thuật toán mã hóa đối xứng nổi tiếng như: DES, AES, RC2, RC4, RC5, RC6,…

Ngoài ra còn một số thuật toán như: Skipjact, Blowfish, CATS-128.

#### 1.4.1.3 Mã hóa bất đối xứng (Asymmetric Cryptography)

Mã hóa bất đối xứng là mô hình mã hóa hai chiều sử dụng một cặp khóa gọi là khóa chung (Public Key) và khóa riêng (Private Key). Trong đó khóa chung Public Key có thể được công bố rộng rãi. Thông thường thông tin được người sử dụng khóa Public Key để mã hóa và gửi đi. Người nhận thông tin sẽ dung khóa Private Key để giải mã. Khóa Private Key chỉ do một người giữ do đó các phương pháp mã hóa bất đối xứng đảm bảo tính bí mật hơn. Một điều quan trọng của phương pháp này là cặp khóa Public Key và Private Key phải tương đồng nhau. Có nghĩa là chỉ có Private Key trong cùng một cặp khóa mới có thể giải mã được dữ liệu đã mã hóa bởi khóa Public Key tương ứng.



*Hình 1.3: Mô hình mã hóa bất đối xứng*

Ví dụ: Bob muốn Kate gửi cho anh ấy một gói hàng bí mật và không thể cho bất kì ai biết được đó là gì nếu biết được có thể làm hại đến anh ấy, thay vì phương pháp mã hóa đối xứng mang nhiều rủi ro, Bob thông minh hơn, anh ấy gửi cho Kate một ổ khóa và yêu cầu Kate sử dụng nó để khóa gói hàng bí mật của mình (đương nhiên chìa khóa do Bob giữ) và gửi đến cho anh ấy. Vì vậy gói hàng sẽ an toàn chuyển đến cho Bob vì không ai có thể mở gói hàng mà không có chìa khóa. Bob chỉ cần nhận gói hàng mà mở ra bằng khóa của anh ấy.

Tương tự ví dụ trên, mã hóa bất đối xứng bao gồm ổ khóa là Public Key và chìa khóa là Private Key với độ bảo mật rất cao, nhưng một nhược điểm của mã hóa bất đối xứng đó là tốc độ giải mã chậm hơn so với mã hóa đối xứng, tức là chúng ta phải tốn nhiều năng lực xử lý của CPU hơn, phải chờ lâu hơn, dẫn đến “chi phí” cao hơn. Khoảng thời gian lâu hơn là bao nhiêu thì còn tùy vào thuật toán mã hóa, cách thức mã hóa và key.

Chính vì thế mà hiện tại ít ai mã hóa cả một file bằng phương pháp bất đối xứng. Thay vào đó, họ xài phương pháp bất đối xứng để mã hóa chính cái key dùng trong mã hóa đối xứng (hoặc tạo ra key đó bằng cách tổng hợp public và private key của bên gửi và nhận).

Mã hóa đối xứng có nhược điểm là key bị lộ là coi như xong đời, vậy thì giờ chúng ta mã hóa luôn cái key đó cho an toàn và có thể gửi key thoải mái hơn. Một khi đã giải mã bất đối xứng để ra key gốc rồi thì tiến hành giải mã thêm lần nữa bằng phương pháp đối xứng để ra file ban đầu.

Thuật toán mã hóa bất đối xứng nổi tiếng và được sử dụng nhiều nhất hiện nay là RSA.

Ngoài ra còn một số thuật toán khác như: Hellman, Elgamal,…

#### 1.4.1.4 Mã hóa hàm băm (Cryptographic Hash Function)

Là cách thức mã hóa một chiều tiến hành biến đổi bản rõ thành bản mã mà không bao giờ giải mã được. Người ta ví loại mã hóa này như một củ hành được băm nhuyễn thì sẽ không bao giờ tái tạo lại được củ hành ban đầu.

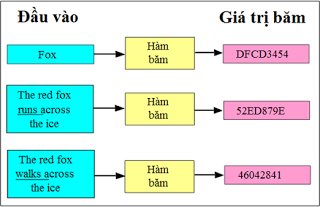
Trong xử lý hàm băm, dữ liệu đầu vào có thể khác nhau về độ dài, nhưng độ dài của xử lý băm luôn xác định. Hàm băm được xử lý trong mô hình xác thực Password.

Ví dụ: Khi bạn đăng nhập vào gmail, mật khẩu mà bạn nhập sẽ được chuyển thành một chuỗi dài các kí tự bằng một thứ gọi là hàm băm (Hash Function).

Chuỗi này sẽ được lưu vào cơ sở dữ liệu, chứ không lưu mật khẩu thô của bạn nhằm tăng tính bảo mật. Lỡ hacker có trộm dữ liệu thì cũng chỉ thấy những thứ như FIiyXYB547bhvyuuUIbZ chứ không biết password thật của bạn là gì.

Mỗi lần bạn đăng nhập, hash function sẽ “băm” password thật của bạn thành chuỗi kí tự rồi so sánh nó với cái trong cơ sở dữ liệu, nếu khớp thì đăng nhập tiếp, không thì báo lỗi. Chúng ta không có nhu cầu dịch ngược chuỗi nói trên ra lại thành password thật để làm gì cả.

Ngoài ra, hàm băm còn dùng để kiểm tra tính toàn vẹn của dữ liệu.



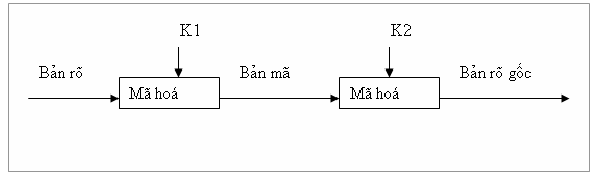
*Hình1.4 : Mô hình mã hóa hàm băm.*

Một số thuật toán mã hóa hàm băm thường dùng như: MD4, MD5, SHA,…

### 1.4.2 Phân loại theo số lượng khóa

#### 1.4.2.1 Mã hóa khóa bí mật

Mã hóa khóa bí mật là thuật toán mà tại đó khóa giải mã có thể được tính toán từ khóa mã hóa. Trong rất nhiều trường hợp khóa mã hóa và khóa giải mã là giống nhau. Thuật toán này yêu cầu người gửi và người nhận thỏa thuận một khóa trước khi thông tin được gửi đi và khóa này phải đảm bảo tính bí mật. Độ an toàn của thuật toán này phụ thuộc nhiều vào độ bí mật của khóa, nếu để tiết lộ khóa thì bất kì người nào cũng có thể mã hóa và giải mã thông tin một cách dễ dàng.



*Hình 1.5: Mô hình mã hóa bí mật.*

Trong đó:

K1 có thể trùng K2

K1 có thể được tính từ K2

K2 có thể được tính từ K1

Mã hóa khóa bí mật thường được sử dụng cho các trường hợp dễ chuyển khóa. Tức là người nhận và người gửi có thể trao đổi khóa cho nhau an toàn, khó bị kẻ khác tấn công. Thường dùng để trao đổi trong văn phòng.

Một số vấn đề liên quan

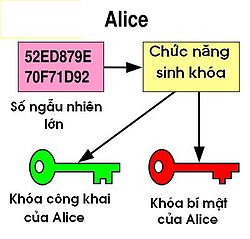
- Các phương pháp mã hóa khóa bí mật đòi hỏi người mã hóa và người giải mã phải cùng chung một khóa hoặc là có thể biết khóa của nhau, khí đó khóa phải được giữ bí mật tuyệt đối. Do đó kẻ địch dễ dàng xác định được một khóa nếu biết khóa kia.

- Phương pháp này không đảm bảo được sự an toàn nếu có sác xuất cao khóa người gửi bị lộ. Khóa phải được gửi đi trên kênh an toàn nếu kẻ địch tấn công trên kênh này có thể phát hiện ra khóa.

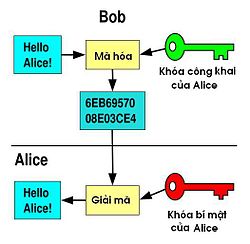
- Vấn đề quản lí và phân phối khóa là khó khăn và phức tạp, người gửi và người nhận phải thống nhất với nhau về khóa việc thay đổi khóa là khó khăn và dễ bị lộ.

#### 1.4.2.2 Mã hóa khóa công khai

Mã hóa khóa công khai là các thuật toán sử dụng khóa mã hóa và khóa giải mã là hoàn toàn khác nhau. Hơn nữa khóa giải mã không thể tính toán được từ khóa mã hóa.

Khác với mã hóa bí mật, khóa mã hóa của một thuật toán mã hóa công khai được công bố rộng rãi. Một người bất kì có thể sử dụng khóa công khai để mã hóa thông tin, nhưng chỉ có người nhận thông tin có khóa giải mã phù hợp với khóa mã hóa để giải mã thông tin đó.

*Hình 1.6: Mật mã hóa khóa công khai.*

**

*Hình 1.7: Dùng khóa công khai để mã hóa, nhưng dùng khóa bí mật để giải mã.*

Trong đó:

Khóa mã hóa không thể giống khóa giải mã.

Khóa giải mã không thể được tính từ khóa mã hóa.

Một điều đặc biệt của loại mã hóa này là cả khóa công khai và bản mã có thể được gửi đi trên kênh không an toàn mà thông tin vẫn không hoặc khó bị đọc trộm dù biết được khóa mã hóa.

Chính vì mức độ an toàn cao nên mã hóa khóa công khai được sử dụng trên mạng công khai Internet. Mã hóa khóa công khai có nhiều ứng dụng quan trọng trong các hệ thống lớn.

# CHƯƠNG 2: MỘT SỐ PHƯƠNG PHÁP MÃ HÓA CỔ ĐIỂN

## 2.1 Hệ mã hóa thay thế

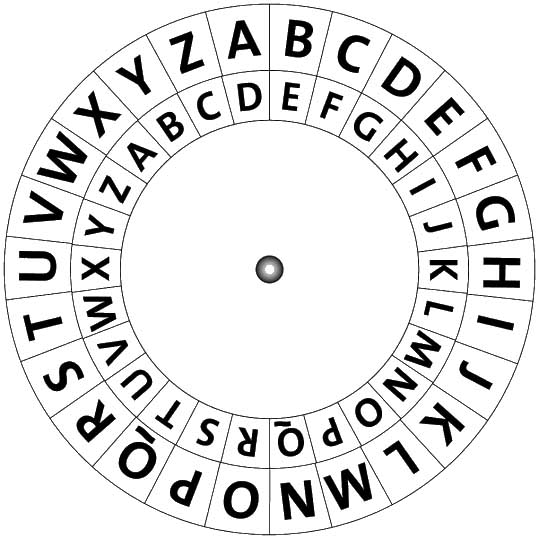
Trong mật mã học, mã thay thế một là phương thức mã hóa mà mỗi "đơn vị" của văn bản gốc được thay thế bằng một "đơn vị" của văn bản mã, đối với các hệ thống thông thường, "đơn vị" ở đây là một chữ cái, một cặp chữ cái, ba chữ cái,...

Mã thay thế thường được so sánh với [mã hoán vị](https://sites.google.com/site/cryptography1100/tai-nguyen/ma-hoan-vi-transposition-cipher). Trong mã hoán vị, mỗi "đơn vị" của văn bản gốc được đổi chỗ, nhưng không có sự thay đổi nào.

Có rất nhiều dạng của mã thay thế. Nếu "đơn vị" là một chữ cái, đó là mã thay thế đơn giản (Simple Substitution Cipher). Nếu "đơn vị" là tập hợp của nhiều chữ cái, đó là *polygraphic*. Một hệ mã thay thế một bảng chữ cái (Monoalphabetic Cipher) là một hệ mã mà mỗi "đơn vị" của văn bản gốc được thay thế với một giá trị cố định. Trong khi đó, trong một hệ mã nhiều bảng chữ cái (Polyalphabetic Cipher), mỗi "đơn vị" của văn bản gốc được thay thế với một giá trị trong một tập hợp, phụ thuộc vào vị trí của "đơn vị" đó trong văn bản gốc. Có nghĩa là, với mỗi "đơn vị" của văn bản gốc, có nhiều giá trị tương ứng trong văn bản mã và ngược lại.

### 2.1.1 Hệ mã hóa CAESAR

Phương pháp này được Caesar xây dựng trong cuộc chiến Gallic Wars, thay thế tứng ký tự trong bản rõ bằng ký tự thứ ba đứng sau nó. Điều này tương ứng với phép dịch 3 vị trí.



*Hình 2.1: Vòng tròn caesar.*

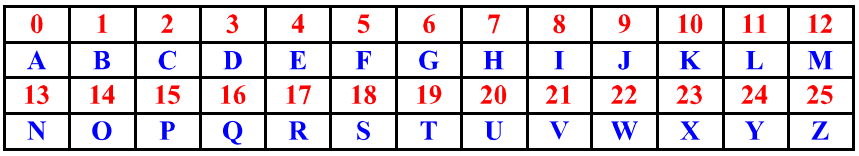
Ví dụ:

○ Meet me after the toga party.

○ PHHW PH DIWHU WKH WRJD SDUWB.

Ở đây thay chữ m bằng chữ đứng thứ 3 sau m là p (m, n, o, p); thay chữ e bằng chữ đứng thứ 3 sau e là h (e, f, g, h).

Về toán học, nếu ta gán số thứ tự cho mỗi chữ trong bảng chữ cái. Các chữ ở dòng trên có số thứ tự tương ứng là số ở dòng dưới:



Mã Ceasar được định nghĩa qua phép tịnh tiến các chữ như sau:

C = E(P) = (P + K) mod (26).

P = D(C) = (C – K) mod (26).

Trong đó:

P, C: số thứ tự của bản rõ và bản mã trong bảng trên.

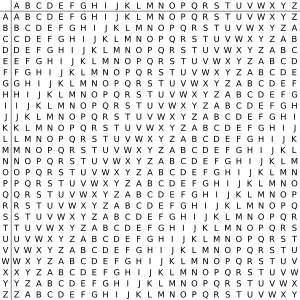
K là phép dịch.

Ví dụ: Bản rõ là: EASY TO BREAK. Sử dụng phép dịch K = 24 để mã hóa. Vậy áp dụng công thức, ta được: P = EASY TO BREAK(4, 0, 18, 24, 19, 14, 1, 17, 4, 0, 10) trở thành C = CYQW RM ZPCYI (2, 24, 16, 22, 17, 12, 25, 15, 2, 24, 8).

Thám mã Ceasar là việc làm đơn giản, do số khoá có thể có là rất ít. Chỉ có 26 khoá có thể, vì A chỉ có thể ánh xạ vào một trong số 26 chữ cái của bảng chữ cái tiếng Anh: A, B, C, …Các chữ khác sẽ được xác định bằng số bước tịnh tiến tương ứng của A. Kẻ thám mã có thể thử lần lượt từng khoá một, tức là sử dụng phương pháp tìm duyệt tổng thể. Vì số khoá ít nên việc tìm duyệt là khả thi. Cho trước bản mã, thử 26 cách dịch chuyển khác nhau (phương pháp vét cạn), ta sẽ tìm được thông tin chính xác thông qua nội dung các bản rõ nhận được.

### 2.1.2 Hệ mã hóa Vigenere

Hệ mã hóa này được đặt theo tên một nhà mật mã người Phap Blaise De Vigenere (1523-1596). VIGERENE cũng giống như CAESAR, nhưng ở đây khóa được thay đổi theo từng bước.



*Hình 2.2: Hình vuông VIGERENE được sử dụng để mã hóa và giải mã.*

Mỗi cột hình vuông VIGERENE có thể xem như một hệ Caesar, với các khóa: 0, 1, 2,…, 25. Để mã hóa thì bản rõ được đọc từ các hàng và khóa được đọc từ các cột.

Ví dụ: Mã hóa bản rõ “MANGMAYTINH” với từ khóa “MAAVIGENERE”. Đầu tiên ta tìm điểm giao của hàng M cột M ta dược Y, tiếp tục ta tìm điểm giao của hàng A cột A ta được A. Cứ như vậy ta được bản mã là “YANBUGCGMEL”. Ta sẽ thu được bản mã tương tự nếu ta đọc bản rõ tương ứng với cột và khóa đọc tương ứng với hàng. Muốn giải mã thông tin vừa mã hóa trên ta thực hiện bằng cách, ta nhìn vào hàng nào chứa Y trong cột M, ta tìm được chữ M, tương tự nhìn vào hàng nào chứa A trong cột A, ta tìm được chữ A. Cứ như vậy ta tìm được bản rõ là “MANGMAYTINH”.

Trong ví dụ trên thì độ dài bản rõ bằng độ dài khóa. Nhưng trong thực tế độ dài bản rõ thường dài hơn rất nhiều so với khóa. Như vậy để mã hóa hay giải mã thì ta phải áp dụng từ khóa một cách tuần hoàn. Nghĩa là từ khóa được lặp đi lặp lại nhiều lần sao cho các ký hiệu trong bản rõ phải được đọc hết.

Ta thấy rằng trong hệ mã hóa VIGERENE, với khóa có độ dài d thì sẽ có 26d khóa hợp lệ. Vì vậy chỉ cần với giá trị d nhỏ thì phương pháp thám mã vét cạn cũng đòi hỏi khá nhiều thời gian.

## 2.2 Hệ mã hóa hoán vị

Hệ mã hóa hoán vị hay còn goị là hệ mã hóa đổi chỗ. Là hệ mã hóa mà các kí tự của bản rõ vẫn được giữ nguyên, nhưng thứ tự của chúng được đổi chỗ vòng quanh.

Phương pháp này có các kĩ thuật mã hóa sau.

### 2.2.1 Đảo ngược toàn bộ bản rõ

Nghĩa là bản gốc được viết theo thứ tự ngược lại từ sau ra trước, để tạo bản mã. Đây là phương pháp mã hóa đơn giản nhất vì vậy không đảm bảo an toàn.

Ví dụ: Bản rõ: MANGMAYTINH

Bản mã: HNITYAMGNAM

### 2.2.2 Mã hóa theo hình mẫu học

Bản gốc được sắp xếp theo một mẫu hình học nào đó, thường là một mảng hoặc ma trận hai chiều.

Ví dụ: Bản rõ “KHOA CÔNG NGHỆ THÔNG TIN” được viết theo ma trận 4x5 như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cột 1 | Cột 2 | Cột 3 | Cột 4 | Cột 5 |
| K | H | O | A | C |
| O | N | G | N | G |
| H | E | T | H | O |
| N | G | T | I | N |

Nếu lấy các ký tự ra theo số thứ tự cột 3, 1, 4, 2, 5 thì ta thu được bản mã “OGTTKOHNANHIHNEGCGON”.

### 2.2.3 Đổi chỗ cột

Đầu tiên biễu diễn các ký tự trong ban rõ thành dạng hình chữ nhật theo cột, sau đó các cột được sắp xếp lại và các chữ cái được lấy ra theo hàng ngang

Ví dụ : Bản rõ là “KHOA CONG NGHE THONG TIN” được viết theo cột như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cột 1 | Cột 2 | Cột 3 | Cột 4 | Cột 5 |
| K | C | N | T | G |
| H | O | G | H | T |
| O | N | H | O | I |
| A | G | E | N | N |

Vì có 5 cột nên ta có thể sắp xếp theo 5!=120 cách khác nhau. Để tăng độ an toàn có thể chọn một trong các cách sắp xếp lại đó.

Nếu ta chuyển vị trí các cột theo các cột thứ tự 3, 5, 2, 4, 1. Rồi lấy các ký tự ra theo hàng ngang ta sẽ được bản mã “NGCTKGTOHHHINOOENGNA”.

Lưu ý: Các ký tự cách trống được bỏ đi.

### 2.2.4 Hoán vị các ký tự của bản rõ theo chu kì cố định

Giả sử văn bản gốc M = m1m2…Mn được chia thành các khối, mỗi khối gồm d ký tự. Cho f là một hàm hoán vị trên d ký tự, khi đó khóa mã hóa được biểu diễn bởi K(d,f). Văn bản gốc sẽ được viết thành: M = m1m2…md md+1md+2…m2d ….

Sẽ được mã hóa thành: EK(M) = mf(1)mf(2)…mf(d) mf(d+1)mf(d+2)…mf(2d) …. Trong đó mf(1)mf(2)…mf(d) là một hoán vị của m1m2…md.

Ví dụ: Ta sẽ mã hóa xâu “ AN TOAN BAO MAT THONG TIN TREN MANG”, với chu kỳ d=6, và hàm f hoán vị dãy i=123456 thành f(i)=352164:

Ta viết xâu dưới dạng: “ANTOAN BAOMAT THONGT INTREN MANG”.

Xâu mã hóa trở thành: “TANABO OAABTM OGHTTN TENINR NAMG”.

# CHƯƠNG 3: MỘT SỐ THUẬT TOÁN MÃ HÓA HIỆN ĐẠI

## 3.1 Thuật toán mã hóa DES

DES (viết tắt của Data Encryption Standard, hay Tiêu chuẩn Mã hóa Dữ liệu) là một phương pháp [mật mã hóa](https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a) được [FIPS](https://vi.wikipedia.org/w/index.php?title=FIPS&action=edit&redlink=1) (Tiêu chuẩn Xử lý Thông tin Liên bang Hoa Kỳ) chọn làm chuẩn chính thức vào năm [1976](https://vi.wikipedia.org/wiki/1976). Sau đó chuẩn này được sử dụng rộng rãi trên phạm vi thế giới. Ngay từ đầu, thuật toán của nó đã gây ra rất nhiều tranh cãi, do nó bao gồm các thành phần thiết kế mật, [độ dài khóa](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%99_l%E1%BB%9Bn_kh%C3%B3a) tương đối ngắn, và các nghi ngờ về [cửa sau](https://vi.wikipedia.org/w/index.php?title=C%E1%BB%ADa_sau&action=edit&redlink=1) để [Cơ quan An ninh quốc gia Hoa Kỳ](https://vi.wikipedia.org/w/index.php?title=C%C6%A1_quan_An_ninh_qu%E1%BB%91c_gia_Hoa_K%E1%BB%B3&action=edit&redlink=1) (NSA) có thể bẻ khóa. Do đó, DES đã được giới nghiên cứu xem xét rất kỹ lưỡng, việc này đã thúc đẩy hiểu biết hiện đại về [mật mã khối](https://vi.wikipedia.org/w/index.php?title=M%E1%BA%ADt_m%C3%A3_kh%E1%BB%91i&action=edit&redlink=1) (block cipher) và các phương pháp [thám mã](https://vi.wikipedia.org/w/index.php?title=Th%C3%A1m_m%C3%A3&action=edit&redlink=1) tương ứng.

Hiện nay DES được xem là không đủ an toàn cho nhiều ứng dụng. Nguyên nhân chủ yếu là độ dài 56 [bit](https://vi.wikipedia.org/wiki/Bit) của [khóa](https://vi.wikipedia.org/wiki/Kh%C3%B3a_(m%E1%BA%ADt_m%C3%A3)) là quá nhỏ. Khóa DES đã từng bị phá trong vòng chưa đầy 24 giờ. Đã có rất nhiều kết quả phân tích cho thấy những điểm yếu về mặt lý thuyết của mã hóa có thể dẫn đến phá khóa, tuy chúng không khả thi trong thực tiễn. Thuật toán được tin tưởng là an toàn trong thực tiễn có dạng [Triple DES](https://vi.wikipedia.org/w/index.php?title=Triple_DES&action=edit&redlink=1) (thực hiện DES ba lần), mặc dù trên lý thuyết phương pháp này vẫn có thể bị phá. Gần đây DES đã được thay thế bằng [AES](https://vi.wikipedia.org/wiki/AES_(m%C3%A3_h%C3%B3a)) (Advanced Encryption Standard, hay Tiêu chuẩn Mã hóa Tiên tiến).

**3.1.1 Lịch sử ra đời**

Khoảng những năm 1970, tiến sĩ Horst Feistel đã đặt nền móng đầu tiên cho chuẩn mã hóa DES với phương pháp mã hóa Feistel Cipher. Vào năm 1976 Cơ quan Bảo mật quốc gia Hoa Kỳ (NSA) đã công nhận DES dựa trên phương pháp Feistel là chuẩn mã hóa dữ liệu. Kích thước khóa ban đầu của DES là 128 bit nhưng tại bản công bố FIPS kích thước đước rút xuống 56 bit để tăng tốc độ xử lý và đưa ra các tiêu chuẩn thiết kế một chuẩn mã hóa dữ liệu.

Nội dung phương pháp mã hóa DES: DES thực hiện mã hóa dữ liệu qua 16 vòng lặp mã hóa, mỗi vòng sử dụng một khóa chu kỳ 48 bit dược tạo ra từ khóa ban đầu có độ dài 56 bit. DES có 8 bảng hằng số S-box để thao tác.

**3.1.2 Mô tả thuật toán DES**

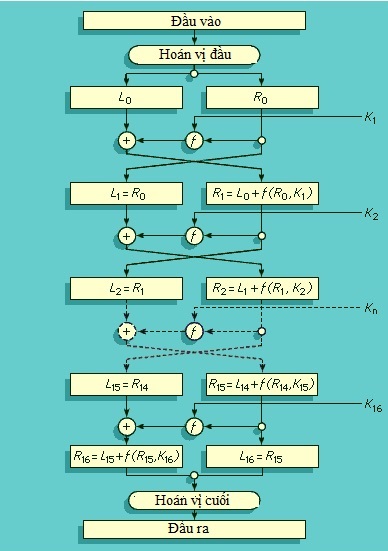
**3.1.2.1 Sơ đồ tổng quát**

Phương pháp DES mã hóa khối thông tin x có độ dài 64 bit với khóa k có độ dài 56 bit thành khối y có độ dài 64 bit.

Nền tảng xây dựng khối của DES là sự kết hợp đơn giản của các kỹ thuật thay thế và hoán vị bản rõ dựa trên khóa, đó là vòng lặp. DES sử dụng 16 vòng lặp áp dụng cùng một kiểu kết hợp các kỹ thuật trên khối bản rõ.

Thuật toán chỉ sử dụng các phép toán số học và logic thông thường trên các số 16 bit, vì vậy nó dễ dàng thực hiện vào nhưng năm 1970 trong điều kiện về công nghệ phần cứng lúc bấy giờ.

Sơ đồ tổng quát:



*Hình 3.1: Sơ đồ tổng quát mã hóa DES*

Quá trình có 16 vòng thực hiện giống nhau trong quá trình xử lý. Ngoài ra có hai lần hoán vị đầu (IP) và cuối (IP-1), hai hoán vị này được sử dụng với mục đích đưa thông tin vào và lấy thông tin ra.

Muốn vào vòng mã hóa thì khối thông tin x ban đầu 64 bit được chia làm hai khối, mỗi khối 32 bit. Hàm f làm biến đổi một nữa của khối đang xử lý với một khóa con tương ứng với vòng mã hóa.

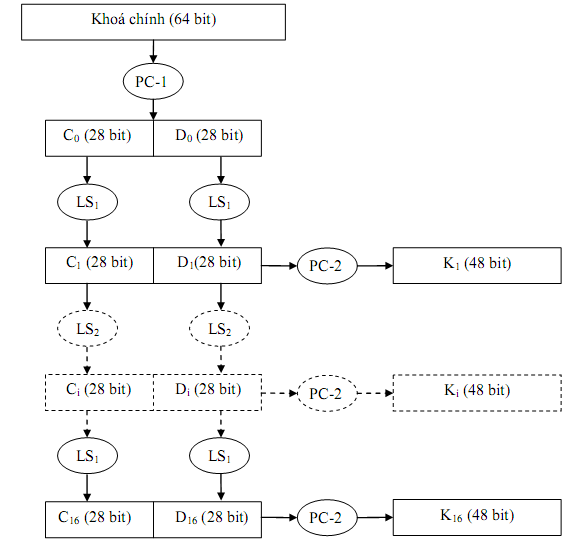
Trong đầu ra của hàm có hàm f được kết hợp với nửa khối còn lại bằng phép toán XOR, và hai phần được trao đổi để xử lý trong chu trình kế tiếp.

Cứ thực hiện các vòng như vậy cho tới vòng cuối cùng thì hai phần không bị tráo đổi nữa, chính vì điều này mà quá trình mã hóa và giải mã là giống nhau.

**3.1.2.2 Quá trình tạo khóa**

Quá trình mã hóa được thực hiện 16 vòng, mỗi vòng cần một khóa. Như vậy từ khóa ban đầu tạo ra 16 khóa con cho 16 vòng lặp tương ứng.

Sơ đồ quá trình tạo khóa:



*Hình 3.2: Sơ đồ tạo khóa*

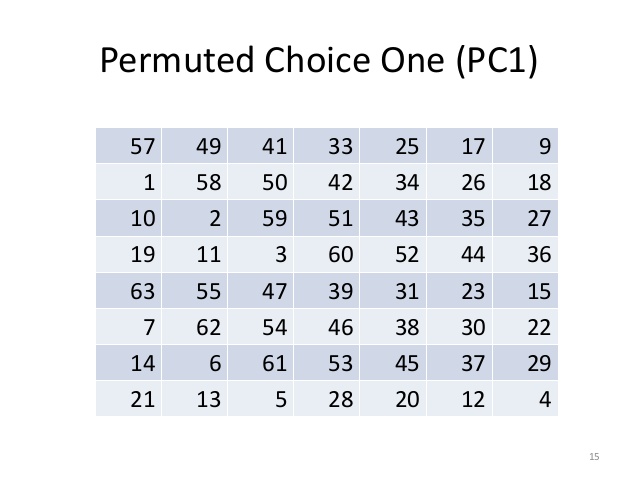
Ví dụ: ta mã hoá một bản tin M “0123456789ABCDEF” với chìa khoá K là “133457799BBDFF1”

DES hoạt động trên các khối 64 bit sử dụng 1 khóa có kích thước 56 bit. Tuy nhiên khóa thực sự được lưu trữ với chiều dài 64 bit và có 8 bit (chẵn lẻ) không được sử dụng (8, 16, 24, 32, 40, 48, 56, 64). Nhưng ta sẽ phải xem xét 8 bit này khi tạo ra các khóa con (Subkeys).

Ta chuyển khóa K:” “133457799BBDFF1” thành dạng nhị phân.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** | **0** | **0** |
| **0** | **1** | **0** | **1** | **0** | **1** | **1** | **1** |
| **0** | **1** | **1** | **1** | **1** | **0** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **1** | **1** | **1** | **1** | **0** | **0** |
| **1** | **1** | **0** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** | **1** |

Khóa 64 bit sẽ được hoán đổi vị trí theo bảng PC1.

****

*Hình 3.3: Bảng hoán vị PC1*

Ta có thể thấy phần tử đầu tiên của bàng là 57 diều đó nghĩa là bit thứ 57 của khóa K sẽ trở thành bit đầu tiên để trở thành khóa K+. Bit thứ 49 trong khóa ban đầu sẽ trở thành bit thứ 2. Sau cùng bit thứ 4 trong khóa K sẽ trở thành bit cuối cùng sau khi hoán vị. Ở đây chỉ có 56 bit trong khóa K thực hiện việc hoán vị (8 bit không thực hiện).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **1** | **1** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** | **0** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **1** | **1** |

K+

Tiếp tục ta chia 56 bit thu được thành hai phần, mỗi phần 28 bit, các phần được xử lý độc lập nhau.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **1** | **1** | **1** | **1** |

C0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** | **0** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **1** | **1** |

D0

Các phần này được dịch 1 hay 2 bit là phụ thuộc vào vòng đó. Số bit dịch được cho trong bảng sau.

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

Ta tiến hành dịch bit, ở đây ta dịch sang trái, áp dụng vào C0 và D0 ta được C1 và D1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **0** | **0** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** | **0** |

C1

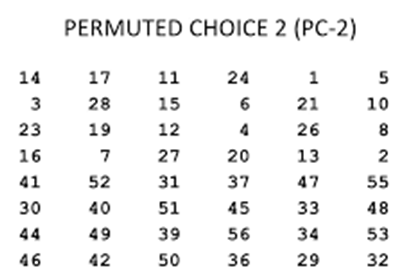
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **0** | **1** | **0** | **1** | **0** | **0** |
| **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **1** | **0** |

D1

Sau đó cộng C1 và D1 ta được.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **0** | **0** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** | **1** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **0** |
| **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **1** | **1** | **1** | **1** | **0** |

Sau khi dịch bit, 56 bit này được chọn ra 48 bit. Bởi vì sự thực hiện đổi chỗ thứ tự các bit như là sự lựa chọn một tập con các bit, nó còn được gọi là hoán vị nén hoặc hoán vị lựa chọn. Sự thực hiện này cung cấp một tập hợp các bit cùng cỡ với đầu ra của hoán vị mở rộng.



*Hình 3.4: Bảng hoán vị PC2*

Ta tiến hành dò với bản hoán vị PC2 để tìm K1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** | **0** |

K1

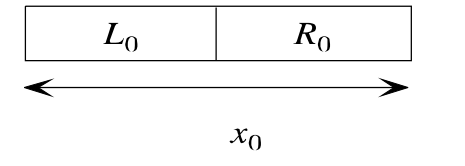
Như vậy, sau khi qua PC2 ta được khóa K1 48 bit để sử dụng trong vòng mã hóa. Tương tự, ta tiếp tục chia hai phần, mỗi phần 28 bit sau khi được dịch bit ở lần thứ nhất, tiếp tục dịch bit lần thứ 2 và băng qua PC2 để hoán vị nén 48 bit và làm K2.

Quá trình cứ tiếp tục như vậy ta thu được 16 khóa Ki (i=1, 2,…, 16).

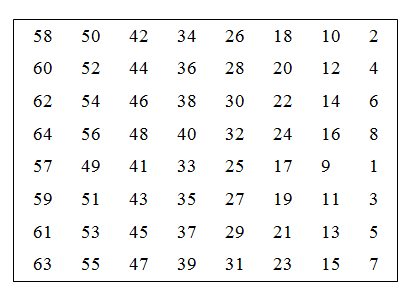
#### 3.1.2.3 Hoán vị khởi đầu

Mục đích của hoán vị khởi đầu là đổi chỗ các bit của khối dữ liệu vào thông qua bảng IP. Nó không ảnh hưởng đến sự an toàn của DES.

Với khối dữ liệu vào x 64 bit cho trước, một xâu bit x0 sẽ được xây dựng bằng cách hoán vị các bit của x theo phép hoán vị cố định ban đầu IP. Ta viết x0=IP(x)=L0R0 trong đó L0 gồm 32 bit đầu và R0 gồm 32 bit cuối.



*Hình 3.5: Biểu diễn dãy 64 bit x chia thành 2 thành phần L0R0.*



*Hình 3.6: Bảng hoán vị khởi đầu IP.*

#### 3.1.2.4 Quá trình mã hóa

Tiếp tục ví dụ phần 3.1.2.2 trên, ta chuyển M “0123456789ABCDEF” thành dạng nhị phân sẽ thu được một khối 64 bit.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** |
| **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** | **0** | **1** |
| **0** | **1** | **1** | **0** | **0** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **1** | **0** | **0** | **1** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **0** | **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** |

Ta tìm hoán vị IP (Initial Permutation) cho chuổi M 64 bit cần mã hóa bằng bảng hoán vị IP (Hình 3.5) . Bit thứ 58 của M trở thành bit đầu tiên của IP, bit 50 trở thành bit thứ 2 của IP và bit cuối cùng của IP là bit thứ 7 của M.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |

Hoán vị IP của M

Kế đến, khối thông tin IP được tách làm 2 phần 32 bit, 32 bit trái L0 và 32 bit R0 phải.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

L0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |

R0

Chu trình sau được lặp lại 16 lần , mỗi khối 32 bit sẽ được xử lý tuần tự thông qua hàm F. Ta có công thức sau:

Ln=R(n-1).

Rn=L(n-1) F(Rn-1,Kn).

L16R16 là khối cuối cùng (n=16).

Với n=1 ta có:

L1=R0­.

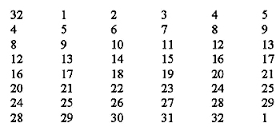
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** |

R1=L0 F(R0,K1)

Hàm F được tính như sau:

Đầu tiên mở rộng khối Rn-1 từ 32 bit thành 48 bit bằng cách sử dụng bảng chọn E để lặp lại mốt số bit trong Rn-1. Đầu vào của E(Rn-1) là 1 khối 32 bit và đầu ra là khối 48 bit.

Đầu ra của E là 48 bit, được viết thành 8 khối, mỗi khối 6 bit bằng cách chọn một số bit từ đầu vào theo bảng bên dưới:

****

*Hình 3.7: Bảng E-box*

Từ đó ta tính được E(R0) là:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0** | **1** | **1** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** | **0** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** | **0** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** |

Tiếp theo ta thực hiện phép XOR E(R(n-1)) với khóa Kn. Ta sẽ sử dụng khóa K1 đã tìm ở trên:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **1** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** |
| **1** | **1** | **0** | **0** | **1** | **0** |

K1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0** | **1** | **1** | **0** | **0** | **0** |
| **0** | **1** | **0** | **0** | **0** | **1** |
| **0** | **1** | **1** | **1** | **1** | **0** |
| **1** | **1** | **1** | **0** | **1** | **0** |
| **1** | **0** | **0** | **0** | **0** | **1** |
| **1** | **0** | **0** | **1** | **1** | **0** |
| **0** | **1** | **0** | **1** | **0** | **0** |
| **1** | **0** | **0** | **1** | **1** | **1** |

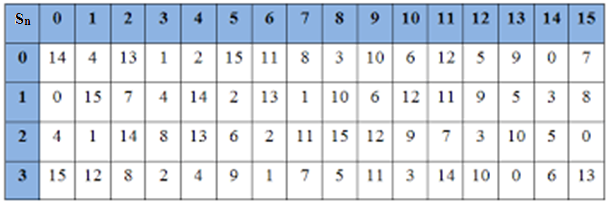
K1  E(R0)

48 bit sau khi thực hiện xog phép toán sẽ được chia làm 8 khối con theo thứ tự B1,B2,……,B8 và được xử lý qua hôp S-box. Mỗi khối gồm 6 bit.

Kn E(R(n-1))=S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8).

Dựa vào hàm S1,S2,…,S8 ta có thể thay thế khối 6 bit thành khối 4 bit.

Sau đây là bảng S1:

****

S1

*Hình 3.8: Bảng S1-box.*

Cách tìm Sn(Bn) với n từ 1 đến 8:

- Xác định hàng i: Lấy bit đầu tiên và bit cuối cùng của B ghép lại được mốt số trong hệ nhị phân có giá trị từ 00 đến 11, đổi số này sang hệ thập phân ta xác định được hàng i trong bảng S1.

- Xác định cột j: Lấy 4 bit ở giữa của B đổi sang hệ thập phân ta xác định được cột j trong bảng Sn.

- Từ i và j vừa tìm được tìm trọng bảng Sn được giá trị hệ thập phân, đổi giá trị này sang hệ nhị phân ta dược Sn(Bn).

○ Tính S1(B1).

B1=011000.

Bit đầu tiên là 0, bit cuối cùng là 0, ghép lại thành 00 đổi sang thập phân là 0. Nên ta được hàng trong bảng Sn là 0.

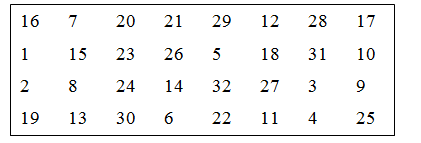
4 bit ở giữa là 1100, đổi sang thập phân là 12, ta được cột trong bảng S1. Dò trong bảng Sn hàng 0 cột 12 được giá trị là 5, đổi sang nhị phân là.

Vậy S1(B1)=0101.

Tương tự ta dò bảng S2, S3,…, S8 ta sẽ dc kết quả S-box S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) là:

|  |  |  |  |
| --- | --- | --- | --- |
| **0** | **1** | **0** | **1** |
| **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** |
| **0** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** |
| **0** | **1** | **1** | **1** |

Tiếp theo 32 bit thu được sau S-box sẽ được sắp xếp lại theo một thứ tự cho trước (còn gọi là P-box):



*Hình 3.9: Bảng P-box*.

Dựa theo P-box ta được F là:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **1** | **0** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **0** | **1** | **0** | **0** | **1** |
| **1** | **0** | **1** | **1** | **1** | **0** | **1** | **1** |

F(R0,K1)

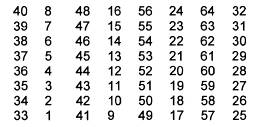
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **1** | **1** | **0** | **1** | **1** | **1** | **1** |
| **0** | **1** | **0** | **0** | **1** | **0** | **1** | **0** |
| **0** | **1** | **1** | **0** | **0** | **1** | **0** | **1** |
| **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |

R1=L0F(R0,K1)

Tương tự ta thực hiện hết 16 vòng lặp ta thu được L16 và R16.

#### 3.1.2.5 Hoán vị cuối cùng

Khi có được L16 và R16 ta hợp 2 khối thành khối 64 bit R16L16 và tính theo bảng IP-1 bên dưới:

****

*Hình 3.10: Bảng IP-1*

Sau 16 vòng lặp ta tính được L16 vs R16 như sau:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** |
| **0** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **0** | **0** |

R16

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** |
| **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** |
| **1** | **0** | **0** | **1** | **0** | **1** | **0** | **1** |

L16

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **0** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** |
| **0** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
| **0** | **0** | **1** | **1** | **0** | **1** | **0** | **0** |
| **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** |
| **0** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |
| **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** |
| **1** | **0** | **0** | **1** | **0** | **1** | **0** | **1** |

R16L16

Sau đó ta dò với bảng IP-1 ta được:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **0** | **0** | **0** | **0** | **1** | **0** | **1** |
| **1** | **1** | **1** | **0** | **1** | **0** | **0** | **0** |
| **0** | **0** | **0** | **1** | **0** | **0** | **1** | **1** |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** | **0** |
| **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** |
| **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** |
| **1** | **0** | **1** | **1** | **0** | **1** | **0** | **0** |
| **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** |

Sau đó ta đổi IP-1 từ nhị phân sang thập lục phân: 85E813540F0AB405. Vậy kết quả cuối cùng ta được là:

Bản rõ M=0123456789ABDEF.

Bản mã DES=85E813549F9AB405.

#### 3.1.2.6 Quá trình giãi mã

Việc giải mã dùng cùng một thuật toán như việc mã hoá. Để giải mã dữ liệu đã được mã hoá, quá trình như giống như mã hoá được lăp lại nhưng các chìa khoá phụ được dùng theo thứ tự ngược lại từ K16 đến K1, nghĩa là trong bước 2 của quá trình mã hoá dữ liệu đầu vào ở trên R(r-1) sẽ được XOR với K(17-r) chứ không phải với Kr.

### 3.1.4 Độ an toàn của thuật toán DES

Ta hãy xem xét tính an toàn của DES trước một vài phương pháp tấn công phá mã.

Tấn công vét cạn khóa (Brute Force Attack): Vì khóa của mã DES có chiều dài là 56 bít nên để tiến hành brute-force attack, cần kiểm tra 256 khóa khác nhau. Hiện nay với những thiết bị phổ dụng, thời gian gian để thử khóa là rất lớn nên việc phá mã là không khả thi. Tuy nhiên vào năm 1998, tổ chức Electronic Frontier Foundation (EFF) thông báo đã xây dựng được một thiết bị phá mã DES gồm nhiều máy tính chạy song song, trị giá khoảng 250.000$. Thời gian thử khóa là 3 ngày. Hiện nay mã DES vẫn còn được sử dụng trong thương mại, tuy nhiên người ta đã bắt đầu áp dụng những phương pháp mã hóa khác có chiều dài khóa lớn hơn (128 bít hay 256 bít) như TripleDES hoặc AES.

Phá mã DES theo phương pháp vi sai (Differential Cryptanalysis): Năm 1990 Biham và Shamir đã giới thiệu phương pháp phá mã vi sai. Phương pháp vi sai tìm khóa ít tốn thời gian hơn brute-force. Tuy nhiên phương pháp phá mã này lại đòi hỏi phải có 247 cặp bản rõ - bản mã được lựa chọn (Chosen-Plaintext). Vì vậy phương pháp này là bất khả thi dù rằng số lần thử có thể ít hơn phương pháp brute-force.

Phá mã DES theo phương pháp thử tuyến tính (Linear Cryptanalysis): Năm 1997 Matsui đưa ra phương pháp phá mã tuyến tính. Trong phương pháp này, cần phải biết trước 243 cặp bản rõ-bản mã (Known-Plaintext). Tuy nhiên 243 cũng là một con số lớn nên phá mã tuyến tính cũng không phải là một phương pháp khả thi.

### 3.1.5 Mô hình ứng dụng

DES thường được dùng để mã hoá bảo mật các thông tin trong quá trình truyền tin cũng như lưu trữ thông tin. Một ứng dụng quan trọng khác của DES là kiểm tra tính xác thực của mật khẩu truy nhập vào một hệ thống (hệ thống quản lý bán hàng, quản lý thiết bị viễn thông…), hay tạo và kiểm tính hợp lệ của một mã số bí mật (thẻ internet, thẻ điện thoại di động trả trước), hoặc của một thẻ thông minh (thẻ tín dụng, thẻ payphone…).

## 3.2 Thuật toán mã hóa RSA

### 3.2.1 Khái quát về RSA

Thuật toán được Ron Rivest, Adi Shamir và Len Adleman mô tả lần đầu tiên vào năm 1977 tại Học viện Công nghệ Massachusetts (MIT). Tên của thuật toán lấy từ 3 chữ cái đầu của tên 3 tác giả.

Trước đó, vào năm 1973 Clifford Cocks, một nhà toán học người Anh, đã mô tả một thuật toán tương tự. Với khả năng tính toán thời điểm đó thì thuật toán này không khả thi và chưa bao giờ được thực hiện. Tuy nhiên, phát minh này chỉ được công bố vào năm 1997 vì được xếp vào loại tuyệt mật.

Thuật toán RSA được MIT đăng kí bằng sáng chế tại Hoa Kỳ vào năm 1983. Bằng sáng chế này hết hạn vào ngày 21 tháng 9 năm 2000. Tuy nhiên, do thuật toán đã được công bố trước khi có đăng ký bảo hộ nên sự bảo hộ hầu như không có giá trị bên ngoài Hoa Kỳ. Ngoài ra, nếu công trình của Clifford Cocks đã được công bố trước đó thì bằng sáng chế RSA đã không thể được đăng ký.

### 3.2.2 Mô tả thuật toán

Thuật toán RSA có hai khóa: Khóa công khai (Public Key) và khóa bí mật (Private Key). Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.

Cũng giống như mã hóa bất đối xứng. Mã hóa RSA có hiểu đơn giản là bạn sử dụng một cặp bao gồm chìa khóa và chiếc rương có khóa. Khi bạn muốn lấy một thông tin nào đó từ bạn bè hay bất cứ ai bạn chỉ việc gửi cho họ chiếc rương mà chỉ bạn mới có chìa khóa để mở ra, sau khi bỏ tài liệu vào và khóa lại không ai có thể xem hay chỉnh sửa lại tài liệu đó kể cả người gửi, trong mô hình RSA thì chìa khóa tương ứng với khóa bí mật còn chiếc rương có khóa tương ứng với khóa công khai.

#### 3.2.2.1 Tạo khóa

Đối với thuật toán RSA có 5 bước để tạo khóa công khai và khóa bí mật:

Bước 1: Chọn 2 số nguyên tố lớn p và q với p≠q, lựa chọn ngẫu nhiên và độc lập.

Bước 2: Tính: n= p\*q.

Bước 3: Tính giá trị hàm số Ơle (n) = (p-1)\*(q-1)

Bước 4: Chọn một số tự nhiên e sao cho 1<e<n.

Bước 5: Tính d sao cho 1 ≡ d\*e mod (n) (tìm d sao cho d\*e chia cho (n) dư 1).

Khóa công khai bao gồm:

○ n: module.

○ e: số mũ công khai (cũng gọi là số mũ hóa).

Khóa bí mật bao gồm:

○ n: module (xuất hiện trong cả khóa công khai và khóa bí mật).

○ d: số mũ bí mật (cũng gọi là số mũ giải mã).

#### 3.2.2.2 Mã hóa và giải mã

Giả sử Kate muốn gửi đoạn thông tin M cho Ben. Đầu tiên Kate sẽ chuyển M thành một số m<n theo một hàm có thể đảo ngược (từ m có thể xác định lại M) được thỏa thuận trước. Quá trình này được mô tả như sau:

Lúc này Kate đã có m và biết n cũng như e do Ben gửi. Kate sẽ tính c là bản mã hóa của m theo công thức:

c ≡ me mod n

Hàm trên có thể tính dễ dàng bằng cách sử dụng phương pháp tính hàm mũ (theo module) bằng thuật toán bình phương và nhân. Sau đó Kate gửi c cho Ben.

Ben nhận c từ Kate và biết khóa bí mật d. Ben có thể tìm dược m từ c theo công thức sau:

m ≡ cd mod n

Biết m, Ben tìm lại M theo phương pháp đã thỏa thuận trước.

Như vậy để mã hóa và giải mã RSA ta phải thực hiện qua 5 bước sau:

Bước 1: A nhận khoá công khai của B.

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

Bước 3: Tính

Bước 4: Gửi kết quả C cho B

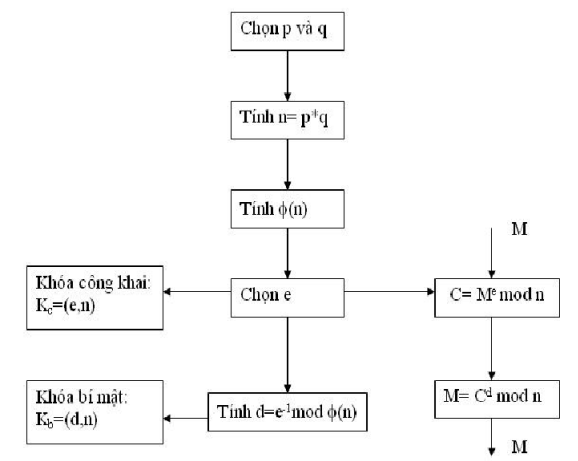
Bước 5: Giải mã

Tính

=> m là thông tin nhận được dưới dạng số

Biểu diễn m thành thông tin ban đầu.

#### 3.2.2.3 Sơ đồ thuật toán



*Hình 3.11: Sơ đồ thuật toán RSA*

#### 3.2.2.4 Ví dụ minh họa

Sau đây là một số ví dụ với những số cụ thể. Ở đây sử dụng những số nhỏ để tiện tính toán còn trong thực tế phải dùng các số có giá trị đủ lớn.

Ta chọn:

|  |  |
| --- | --- |
| p = 5 | Số nguyên tố thứ nhất (giữ bí mật hoặc hủy sau khi tạo khóa) |
| q = 7 | Số nguyên tố thứ hai (giữ bí mật hoặc hủy sau khi tạo khóa) |
| N = pq = 35 | Module (công bố công khai) |
| (n)= (p-1)\*(q-1) =24 | Giá trị hàm số Ơle |
| e = 5 | Số mũ công khai (chon e thỏa điều kiện 1<e<n) |
| d = 5 | Số mũ bí mật (tìm d sao cho 1≡e\*d mod (n)) |

Như vậy ta có cặp khóa:

Public Key = (e,n) = (5,35).

Private Key = (d,n) = (5,35).

Áp dụng để mã hóa chuỗi: SECURE

Trong bảng chữ cái, có tất cả 26 ký tự, các ký tự ứng với một con số. Do đó, ta có bảng sau:

Mã hóa chuỗi SECURE

Áp dụng c ≡ me mod n

|  |  |  |  |
| --- | --- | --- | --- |
| Nội dung | Vị trí (m) | me | Nội dung bị mã hóa (c) |
| S | 19 | 2476099 | 24 |
| E | 5 | 3125 | 10 |
| C | 3 | 243 | 33 |
| U | 21 | 4084101 | 21 |
| R | 18 | 1889568 | 23 |
| E | 5 | 3125 | 10 |

Nếu tại đây, dữ liệu trên đường chuyển tới người nhận bị một người kahc1 bắt được, người đó sẽ không biết nội dung muốn nói điều gì, mà chỉ nhận được đó là những con số, không nói lên được điều gì. Nếu muốn đọc được nội dung, người đó phải có Private Key, mà ứng với Public Key dùng để mã hóa dữ liệu trên thì phải có Private thích hợp. Do đó, dữ liệu sẽ an toàn.

Khi dữ liệu đến tay người nhận, muốn khôi phục lại dữ liệu gốc ban đầu, ta giải mã lại với n=35, d=5 và áp dụng sẽ thu được bản rõ ban đầu.

|  |  |  |  |
| --- | --- | --- | --- |
| Nội dung bị mã hóa (c) | Cd | Vị trí (m) | Nội dung |
| 24 | 245 | 19 | S |
| 10 | 105 | 5 | E |
| 33 | 335 | 3 | C |
| 21 | 215 | 21 | U |
| 23 | 235 | 18 | R |
| 10 | 105 | 5 | E |

### 3.2.3 Một số phương pháp tấn công

Tính chất an toàn của RSA dựa trên cơ sở, chi phí cho việc giải mã bất hợp lệ thông tin đã được mã hóa sẽ quá lớn nên xem như không thể thực hiện được.

Vì khóa là công cộng nên việc tấn công bẻ khóa phương pháp RSA thường dưa vào khóa công cộng để xác định khóa riêng tương ứng. Điều quan trọng là dựa vào n để tính p, q, từ đó tính được d.

#### 3.2.3.1 Phương pháp sử dụng (n)

Giả sử người tấn công biết được giá trị (n). Khi đó việc xác định giá trị p, q được đưa về giải hai phương trình sau:

n = p\*q.

(n) = (p-1)\*(q-1).

Thay q = ta được phương trình bậc hai:

P2 + p\*( (n)-n-1) + n = 0.

p, q chính là 2 nghiệm của phương trình bậc hai này. Hoặc có thể tính trực tiếp số mũ bí mật d từ giá trị (n) và số mũ công khai e.

Tuy nhiên vấn đề phát hiện được (n) còn khó hơn việc xác định 2 thừa số nguyên của n.

#### 3.2.3.2 Áp dụng thuật toán phân tích ra thừa số

Có nhiều thuật toán để phân tích ra thừa số. Tuy nhiên thuật toán Pollard p-1, là một trong những thuật toán đơn giản, hiệu quả, dùng để phân tích ra thừa số nguyên của các số nguyên lớn, ở đây ta phân tích số nguyên n dựa vào phân tích p-1 với p là một ước nguyên tố của n.

Các bước của thuật toán:

Bước 1: Xác định đầu vào của hai số n và b (n là số nguyên lẻ cần phân tích, b là một giá trị giới hạn).

Bước 2: Gán a = 2.

Bước 3: for j = 2 to b do

a = aj mod n

Bước 4: Tìm d = gcd(a-1,n) (d là ước chung lớn nhất của a-1 và n).

Bước 5: Nếu 1<d<n then

d là thừa số nguyên tố của n (thành công).

Ngược lại

Không xác định dược thừa số nguyên của n.

Giải thuật này chỉ hiệu quả khi tấn công phương pháp RSA trong trường hợp n có thừa số nguyên tố p mà p-1 chỉ có các ước số nguyên tố rất nhỏ. Do đó chúng ta có thể dễ dàng xây dựng một hệ mã hóa công cộng RSA an toàn đối với giải thuật tấn công p-1. Cách đơn giản là tìm một số nguyên tố pi lớn mà p = 2pi+1 cũng là số nguyên tố, tương tự, tìm qi là số nguyên tố lớn mà q = 2 qi+1 là nguyên tố.

#### 3.2.3.3 Bẻ khóa dựa trên tấn công lặp lại

Siimons và Norris đã chỉ ra hệ thống RSA có thể bị tổn thương khi sử dụng phương pháp tấn công lặp lại liên tiếp. Đó là khi đối thủ biết cặp khóa công cộng (e, n) và từ khóa C thì họ có thể tính chuỗi các từ khóa sau:

C1 = Ce mod n.

C2 = C1e mod n.

……………....

Ci = Ci-1e mod n.

Nếu có một phần tử Cj trong chuỗi C1, C2,…,Ci. Sao cho Cj = C thì khi đó kẻ tấn công sẽ tìm được M = Cj-1 bởi vì:

Cj = Cj-1e mod n.

C = Me mod n.

Ví dụ: Giả sử anh ta biết{e, n, C} = {17, 35, 3}. Người ta sẽ tính.

C1 = Ce mod n = 317 mod 35 = 33.

C2 = C1e mod n = 3317 mod 35 = 3.

Vì C2 = C nên M = C1 = 33.

### 3.2.4 Đánh giá thuật toán

-RSA đơn giản, dễ hiểu, dễ cài đặt.

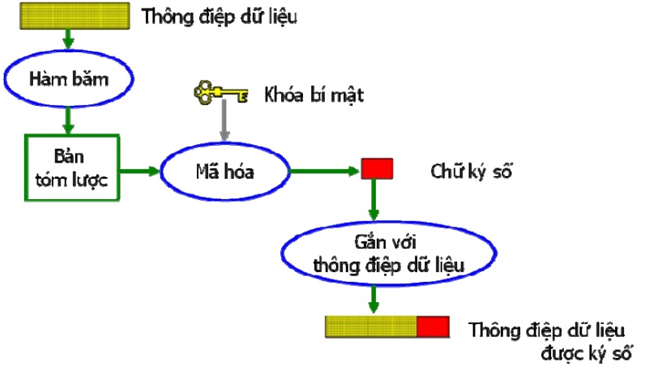
-Hiệu suất hoạt động: RSA chạy chậm do việc phát sinh khóa công khai, khóa bí mật hay quá trình mã hóa, giải mã tốn nhiều thời gian vì phải tính toán trên các số nguyên dương cực lớn, có chiều dài vượt quá khả năng chứa của thanh ghi nên phải thực hiện lại nhiều lần và sử dụng nhiều đến bộ xử lý. Do đó, RSA không được sử dụng vào mục đích mã hóa khối dữ liệu lớn mà chỉ ứng dụng trong chữ ký điện tử để mã hóa thông điệp ngắn qua hàm băm (hash), giải thuật trao đổi khóa bí mật (khóa dùng cho các hệ thống mã hóa đối xứng, hay khóa riêng) hay là chỉ mã hóa dữ liệu với số lượng nhỏ.

-Tính bảo mật: Độ an toàn của RSA dựa trên bài toán phân tích ra thừa số nguyên tố. Do đó, chiều dài của số càng lớn thì càng khó phân tích ra thừa số nguyên tố.

### 3.2.5 Mô hình ứng dụng

#### 3.2.5.1 Ứng dụng

Thuật toán RSA được dùng trọng lĩnh vục tạo chữ ký số

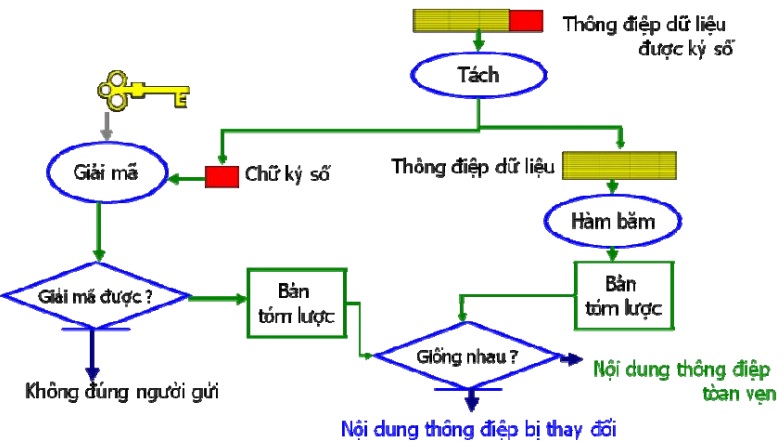


*Hình 3.12: Mô hình tạo chữ ký số từ thuật toán RSA để mã hóa thông tin.*

– Xác thực được người gửi: Các hệ thống mật mã hóa khóa công khai cho phép mật mã hóa văn bản với khóa bí mật mà chỉ có người chủ của khóa biết. Để sử dụng chữ ký số thì văn bản không cần phải được mã hóa mà chỉ cần mã hóa hàm băm của văn bản đó (thường có độ dài cố định và ngắn hơn văn bản). Khi cần kiểm tra, bên nhận giải mã (với khóa công khai) để lấy lại hàm băm và kiểm tra với hàm băm của văn bản nhận được. Nếu 2 giá trị này khớp nhau thì bên nhận có thể tin tưởng rằng văn bản xuất phát từ người sở hữu khóa bí mật. Tất nhiên là chúng ta không thể đảm bảo 100% là văn bản không bị giả mạo vì hệ thống vẫn có thể bị phá vỡ. Vấn đề nhận thực đặc biệt quan trọng đối với các giao dịch tài chính. Chẳng hạn một chi nhánh ngân hàng gửi một gói tin về trung tâm dưới dạng (a,b), trong đó a là số tài khoản và b là số tiền chuyển vào tài khoản đó. Một kẻ lừa đảo có thể gửi một số tiền nào đó để lấy nội dung gói tin và truyền lại gói tin thu được nhiều lần để thu lợi (tấn công truyền lại gói tin).

– Chống chối bỏ: Trong giao dịch, một bên có thể từ chối nhận một văn bản nào đó là do mình gửi. Để ngăn ngừa khả năng này, bên nhận có thể yêu cầu bên gửi phải gửi kèm chữ ký số với văn bản. Khi có tranh chấp, bên nhận sẽ dùng chữ ký này như một chứng cứ để bên thứ ba giải quyết. Tuy nhiên, khóa bí mật vẫn có thể bị lộ và tính không thể phủ nhận cũng không thể đạt được hoàn toàn.

– Xác thực sự toàn vẹn của thông tin: Cả hai bên tham gia vào quá trình thông tin đều có thể tin tưởng là văn bản không bị sửa đổi trong khi truyền vì nếu văn bản bị thay đổi thì hàm băm cũng sẽ thay đổi và lập tức bị phát hiện. Quá trình mã hóa sẽ ẩn nội dung của gói tin đối với bên thứ 3 nhưng không ngăn cản được việc thay đổi nội dung của nó. Một ví dụ cho trường hợp này là tấn công đồng hình (Homomorphism Attack): Tiếp tục ví dụ như ở trên, một kẻ lừa đảo gửi 1.000.000 đồng vào tài khoản của a, chặn gói tin (a,b) mà chi nhánh gửi về trung tâm rồi gửi gói tin (a,b3) thay thế để lập tức trở thành triệu phú.

[](http://tuongluaweb.com/wp-content/uploads/2015/11/thuattoan22.jpg)

*Hình 3.13: Mô hình quy trình giải mã của chữ ký số RSA.*

#### 3.2.5.2 Thực nghiệm

○ Tìm cặp khóa:

#include <iostream>

#include <math.h>

#include <conio.h>

using namespace std;

int main ()

{

int q,p,n,Ole;

cout<<"Nhap so nguyen thu nhat: p = ";cin>>p;

cout<<"Nhap so nguyen thu hai: q = ";cin>>q;

n=q\*p;

Ole=(q-1)\*(p-1);

cout<<"Module: n = "<<n<<endl;

cout<<"Cac cap khoa cong khai va bi mat:"<<endl;

for (int i=2;i<n;i++)

{

for (int j=1;j<100;j++)

{

if ((i\*j)%Ole==1)

{

cout<<"e = "<<i<<"\t"<<"d = "<<j<<endl;

}

}

}

system ("pause");

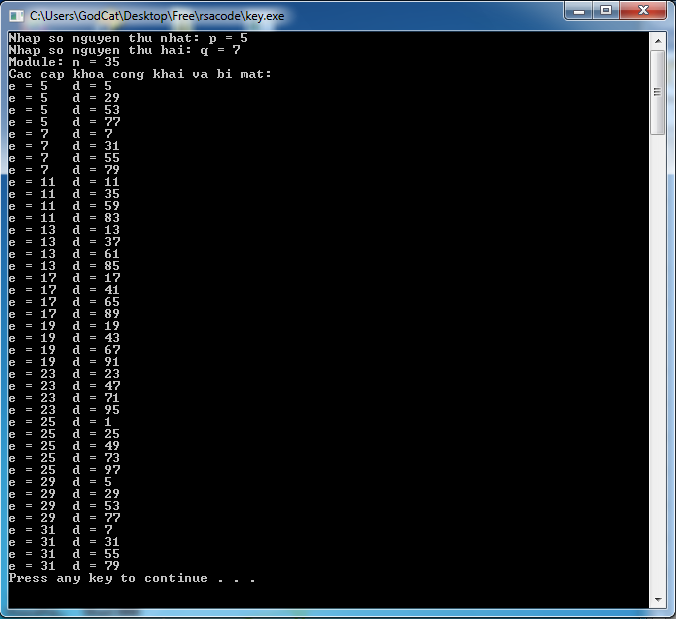
return 0;

}

Để sử dụng mã hóa RSA ta cần phải có PublicKey và PrivateKey trước nên ta phải xây dựng một chương trình tìm cặp khóa cho RSA.

Đầu tiên ta cần người dùng nhập q và p là 2 số nguyên tố ngẫu nhiên và khác nhau để tính module n=p\*q và hàm Ơle (n) = (q-1)\*(p-1).

Dòng for (int i=2;i<n;i++) dùng để tìm các giá trị của số mũ công khai. Bởi vì số mũ công khai e phải thỏa điều kiện 1<e<n nên vòng lặp for phải bắt đầu từ 2 và kết thúc khi i=n. Dòng for (int j=1;j<100;j++) là vòng lặp con của i và là vòng lặp tìm số mũ bí mật d theo công thức 1 d\*e mod (n) (nghĩa là d\*e chia cho (n) dư 1). Ở đây ta chỉ giới hạn d<100 để tiện tính toán bởi vì số mũ càng lớn thì chương trình chạy càng lâu. Vì ngôn ngữ C++ hỗ trợ phép toán chia lấy dư nên ta có thể tìm cặp khóa RSA rất đơn giản. Cứ như vậy ta sẽ được các cặp khóa d và e. Việc tiếp theo là chọn 1 cặp khóa rồi tiến hành mã hóa RSA.



Chọn p=5 và q=7 ta thu được module n=35 và các cặp khóa, ở đây ta không thấy các giá trị e=2 hay e=3,.. bởi vì những giá trị đó không tồn tại d thỏa điều kiện 1 d\*e mod (n) và d<100.

○ Mã hóa RSA:

Vì mã hóa RSA hoạt động dựa trên các hàm toán học nên ta phải chuyển các ký tự sang số để có thể mã hóa. Bảng chữ cái sẽ được đánh số từ 1 đến 26 và sử dụng vào RSA hoặc có thể đánh số theo bước dịch Caesar để tăng độ an toàn. Nếu có ai biết được khóa bí mật và mã hóa thì cũng sẽ nhận được một dãy bản mã Caesar, phải giải mã thêm Caesar nữa mới có được thông tin gốc.

#include <iostream>

#include <math.h>

#include <conio.h>

using namespace std;

int main ()

{

Unsigned long int n,b,e,m;

double t;

char k[100];

cout<<"Nhap so ky tu trong chuoi can ma hoa: ";cin>>m;

for (int i=0;i<m;i++)

{

cout<<"Nhap ky tu thu "<<(i+1)<<": ";

cin>>k[i];

}

for (int i=0;i<m;i++)

{

switch (k[i])

{

case 'a': k[i]=1;break;

case 'b': k[i]=2;break;

case 'c': k[i]=3;break;

case 'd': k[i]=4;break;

case 'e': k[i]=5;break;

case 'f': k[i]=6;break;

case 'g': k[i]=7;break;

case 'h': k[i]=8;break;

case 'i': k[i]=9;break;

case 'j': k[i]=10;break;

case 'k': k[i]=11;break;

case 'l': k[i]=12;break;

case 'm': k[i]=13;break;

case 'n': k[i]=14;break;

case 'o': k[i]=15;break;

case 'p': k[i]=16;break;

case 'q': k[i]=17;break;

case 'r': k[i]=18;break;

case 's': k[i]=19;break;

case 't': k[i]=20;break;

case 'u': k[i]=21;break;

case 'v': k[i]=22;break;

case 'w': k[i]=23;break;

case 'x': k[i]=24;break;

case 'y': k[i]=25;break;

case 'z': k[i]=26;break;

}

}

cout<<"Nhap so mu cong khai va module: e = ";cin>>e;

cout<<"n = ";cin>>n;

cout<<"Day da duoc ma hoa la:";

for (int i=0;i<m;i++)

{

t=pow(k[i],e);

b=(unsigned long int)t%n;

cout<<" "<<b;

}

cout<<endl;

system ("pause");

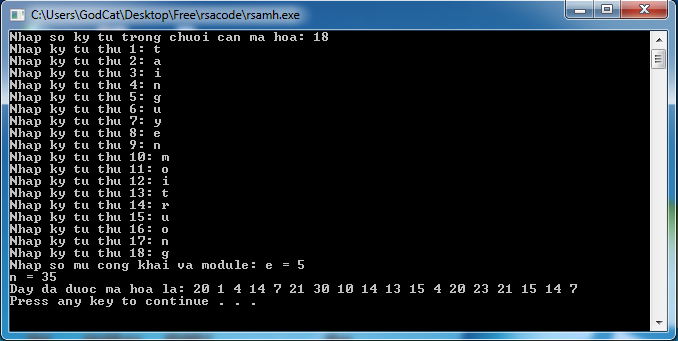
return 0;

}

Đầu tiên ta tạo một mảng k[i] để lưu đoạn thông tin người dùng muốn mã hóa và sử dụng cấu trúc switch để chuyển các ký tự đầu vào thành số tương ứng do người dùng quy định rồi lưu đè vào mảng. Tiếp theo sử dụng khóa công khai và module đã chọn ở bước tìm key và mã hóa theo công thức c ≡ me mod n (Tức là me chia cho n dư c). c chính là bản mã.

Chạy for (int i=0;i<m;i++) để mã hóa hết các ký tự trong mảng k[i]. Dùng hàm pow để tính me và sử dụng phép chia lấy dư để tìm c.

Ta chọn cặp khóa công khai là e=5 và d=5, module n=35 để thực hiên mã hóa dãy “TAINGUYENMOITRUONG”.



Với khóa công khai e=5 và module n=35 thì ta được bản mã “20 1 4 14 7 21 30 10 14 13 15 4 20 23 21 15 14 7”.

○ Giải mã RSA

Quá trình giải mã RSA cũng như quá trình mã hóa nhưng sử dụng khóa bí mật d và sau khi giải mã ta sẽ trả các giá trị về ký tự ban đầu để thu được bản rõ.

#include <iostream>

#include <conio.h>

#include <math.h>

using namespace std;

int main ()

{

int n,a[100],b,m,d;

double t;

cout<<"Nhap so ky tu trong chuoi can giai ma: ";cin>>m;

for (int i=0;i<m;i++)

{

cout<<"Nhap ban ma cua ky tu thu "<<(i+1)<<": ";cin>>a[i];

}

cout<<"Nhap so mu bi mat va module: d = ";cin>>d;

cout<<"n = ";cin>>n;

cout<<"Day da duoc giai ma la: ";

for (int i=0;i<m;i++)

{

t=pow(a[i],d);

b=(int)t%n;

switch (b-0)

{

case 1: cout<<"A";break;

case 2: cout<<"B";break;

case 3: cout<<"C";break;

case 4: cout<<"D";break;

case 5: cout<<"E";break;

case 6: cout<<"F";break;

case 7: cout<<"G";break;

case 8: cout<<"H";break;

case 9: cout<<"I";break;

case 10: cout<<"J";break;

case 11: cout<<"K";break;

case 12: cout<<"L";break;

case 13: cout<<"M";break;

case 14: cout<<"N";break;

case 15: cout<<"O";break;

case 16: cout<<"P";break;

case 17: cout<<"Q";break;

case 18: cout<<"R";break;

case 19: cout<<"S";break;

case 20: cout<<"T";break;

case 21: cout<<"U";break;

case 22: cout<<"V";break;

case 23: cout<<"W";break;

case 24: cout<<"X";break;

case 25: cout<<"Y";break;

case 26: cout<<"Z";break;

}

cout<<" ";

}

cout<<endl;

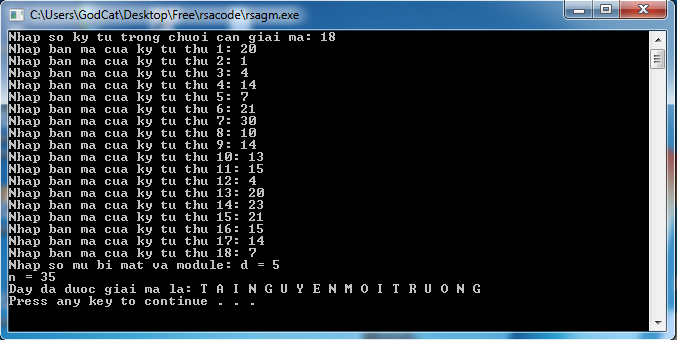
system ("pause");

return 0;

}

Ta cũng sử dụng một mảng để lưu các giá trị bản mã rồi giải mã theo công thức m ≡ cd mod n (Tức là cd chia cho n lấy dư). Sau đó dùng switch chuyển số thành ký tự thì sẽ thu được bản rõ ban đầu.

Với khóa bí mật d=5, module n=35 và bản mã “20 1 4 14 7 21 30 10 14 13 15 4 20 23 21 15 14 7”. Ta được:



Lưu ý: Vì ta sử dụng phép chia lấy dư và mã hóa 26 ký tự nên cần module n ≥ 26, nếu n=20 thì chương trình sẽ không mã hóa (hay giải mã) được các ký tự từ 21 đến 26 mà nó sẽ quay lại mã hóa (hay giải mã) ký tự từ 1 đến 6.

Ở đây ta sử dụng kiểu dữ liệu unsigned long int nên giới hạn của nó là từ 0 đến 4.294.967.285 nếu me hay cd vượt quá giới hạn của kiểu dữ liệu thì chương trình sẽ in ra khoảng trống.

○ Sử dụng

Với chương trình RSA cơ bản này thì ta chỉ cần gửi chương trình mã hóa và PublicKey (e, n) cho đối tượng cần trao đổi thông tin bí mật. Đối tượng sẽ sử dụng chương trình để mã hóa thành dãy số và gửi cho ta trên kênh công cộng, khi nhận được thông tin thì chỉ cần sử dụng chương trình giải mã với PrivateKey (d, n) tương ứng. Thông tin sẽ được an toàn. Dù người khác có thấy được thì đó cũng chỉ là những con số.

## 3.3 Thuật toán mã hóa MD5

### 3.3.1 Hàm băm MD5

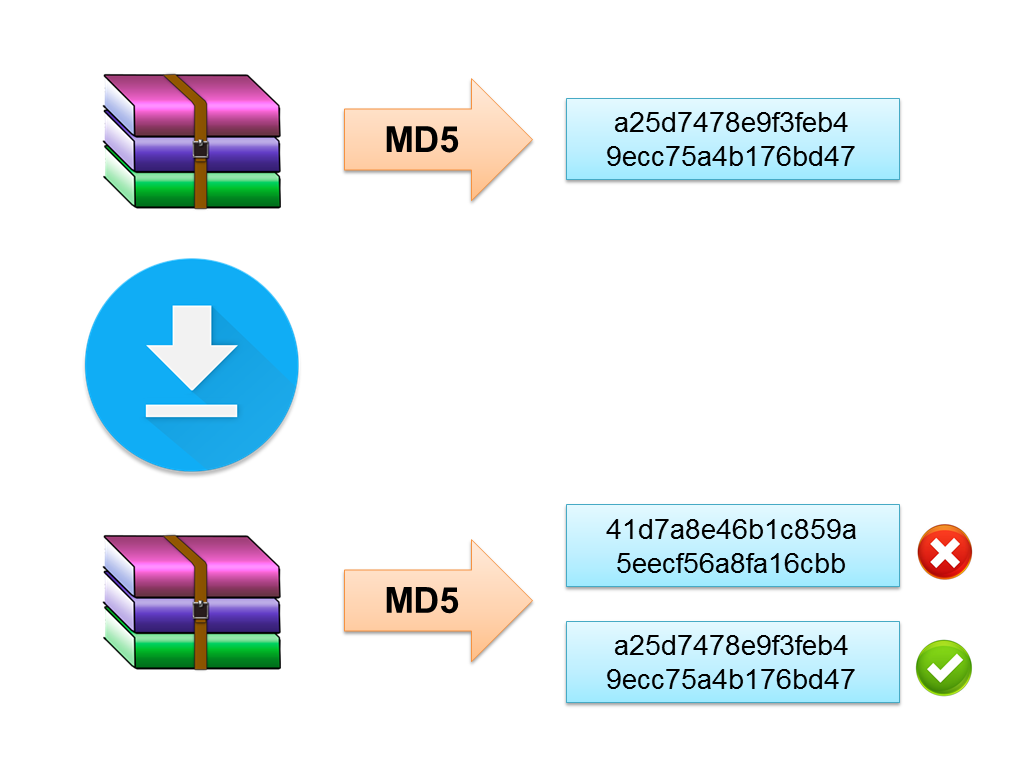
#### 3.3.1.1 Khái niệm

MD5 (Message-Digest algorithm 5) là một hàm băm để mã hóa với giá trị băm là 128 bit. Từng được xem là một chuẩn trên Internet, MD5 đã được sử dụng rộng rãi trong các chương trình an ninh mạng, và cũng thường được dùng để kiểm tra tính nguyên vẹn của tập tin.

MD5 được thiết kế bởi Ronald Rivest vào năm 1991 để thay thế cho hàm băm trước MD4 (cũng do ông thiết kế, trước đó nữa là MD2).

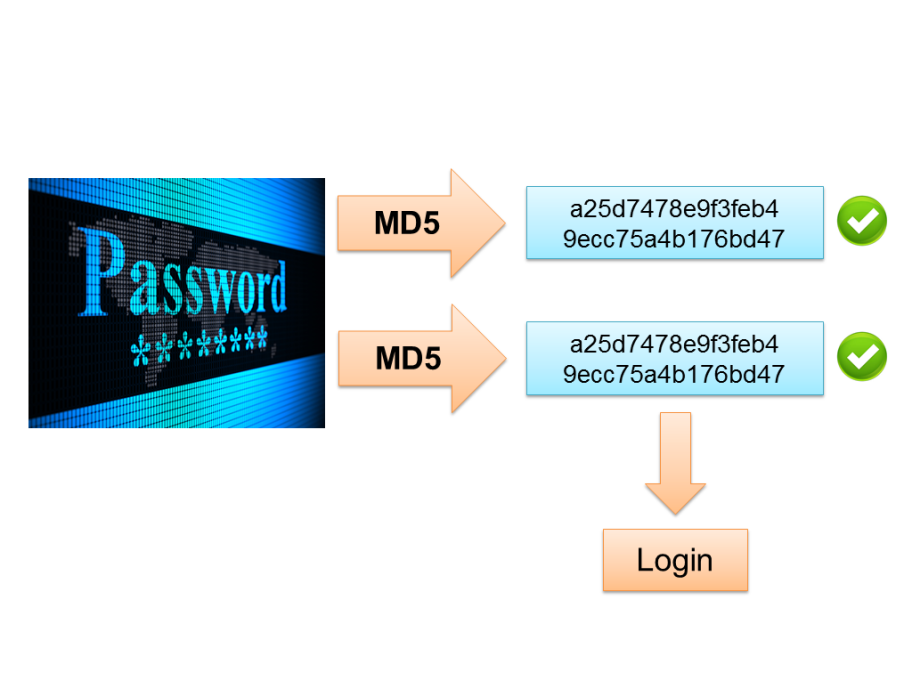
#### 3.3.1.2 Ứng dụng

 MD5 được dùng rộng rãi trong các phần mềm trên toàn thế giới để đảm bảo việc truyền tập tin được nguyên vẹn. Ví dụ, máy chủ tập tin thường cung cấp một [checksum](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_t%E1%BB%95ng_ki%E1%BB%83m) MD5 được tính toán trước cho tập tin, để người dùng có thể so sánh với checksum của tập tin đã tải về. Những hệ điều hành dựa trên nền tảng [Unix](https://vi.wikipedia.org/wiki/Unix) luôn kèm theo tính năng MD5 sum trong các gói phân phối của họ, trong khi người dùng Windows sử dụng ứng dụng của hãng thứ ba.



*Hình 3.14: Mô hình dùng hàm băm xác nhận tính toàn vẹn của dữ liệu.*

MD5 được dùng để mã hóa mật khẩu. Mục đích của việc mã hóa này là biến đổi một chuổi mật khấu thành một đoạn mã khác, sao cho từ đoạn mã đó không thể nào lần trở lại mật khẩu. Có nghĩ là việc giải mã là không thể hoặc phải mất một khoảng thời gian vô tận (đủ để làm nản lòng các hacker).



*Hình 3.15: Mô hình sử dụng hàm băm để mã hóa mật khẩu.*

#### 3.3.1.3 Thuật giải

MD5 chuyển một đoạn thông tin chiều dài thay đổi thành một kết quả chiều dài không đổi 128 bit. Mẩu tin đầu vào được chia thành từng đoạn 512 bit; mẩu tin sau đó được [độn](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BB%99n_(m%E1%BA%ADt_m%C3%A3_h%C3%B3a)&action=edit&redlink=1) sao cho chiều dài của nó chia chẵn cho 512. Công việc độn vào như sau: đầu tiên một bit đơn, 1, được gắn vào cuối mẩu tin. Tiếp theo là một dãy các số 0 sao cho chiều dài của mẩu tin lên tới 64 bit ít hơn so với bội số của 512. Những bit còn lại được lấp đầy bằng một số nguyên 64-bit đại diện cho chiều dài của mẩu tin gốc.

Giải thuật MD5 chính hoạt động trên trạng thái 128-bit, được chia thành 4 từ 32-bit, với ký hiệu A, B, C và D. Chúng được khởi tạo với những hằng số cố định. Giải thuật chính sau đó sẽ xử lý các khối tin 512-bit, mỗi khối xác định một trạng thái. Quá trình xử lý khối tin bao gồm bốn giai đoạn giống nhau, gọi là vòng; mỗi vòng gồm có 16 tác vụ giống nhau dựa trên hàm phi tuyến F, [cộng module](https://vi.wikipedia.org/w/index.php?title=C%E1%BB%99ng_m%C3%B4_%C4%91un&action=edit&redlink=1), và dịch trái.

Hằm băm MD5 ( còn được gọi là hàm tóm tắt thông điệp-message degests) sẽ trả về một chuỗi số hex (thập lục phân) gồm 32 số liên tiếp.

Dưới đây là các ví dụ mô tả các kết quả thu được sau khi băm.

MD5 (“cộng hòa xã hội chủ nghĩa việt nam”).

= 7b8e76fac176d53c53cb24843e31e759.

Thậm chí chỉ cần một thay đổi nhỏ cũng làm thay đổi hoàn toán kết quả trả về:

MD5 (“Cộng Hòa Xã Hội Chũ Nghĩa Việt Nam”).

= 4B290F07CAC9A678697B6D23CCD1062B.

Ngay cả một chuỗi rỗng cũng cho ra một kết quả phức tạp:

MD5(“”) = d41d8cđ8f00b204e9800998ecf8427e.

### 3.3.2 MD5

#### 3.3.2.1 Mô tả thuật toán

Thuật toán có đầu vào là một thông điệp có độ dài tuỳ ý và có đầu ra là một chuỗi có độ dài cố định là 128 bit. Thuật toán được thiết kế để chạy trên các máy tính 32 bit.

#### 3.3.2.2 Thuật toán

Thông điệp đầu vào có độ dài b bit bất kỳ. Biểu diễn các bit dưới dạng như sau: m[0] m[1] m[2] ... m[b-1].

Bước 1: Các bit gắn thêm: Thông điệp được mở rộng, thêm bit vào phía sau sao cho độ dài của nó (bit) đồng dư với 448 theo module 512. Nghĩa là thông điệp được mở rộng sao cho nó còn thiếu 64 bit nữa thì sẽ có một độ dài chia hết cho 512. Việc thêm bit này được thực hiện như sau: một bit ‘1’ được thêm vào sau thông điệp, sau đó các bit ‘0’ được thêm vào để có một độ dài đồng dư với 448 module 512.

Bước 2: Gắn thêm độ dài: Biểu diễn độ dài b của chuỗi ban đầu thành dạng nhị phân 64 bit được thêm vào phía sau kết quả của bước 1. Ta sẽ thu được một chuỗi 512 bit.

Bước 3: Khởi tạo bộ đệm MD: Một bộ đệm 4 từ (A,B,C,D) được dùng để tính mã số thông điệp. Ở đây mỗi A,B,C,D là một thanh ghi 32 bit. Những thanh ghi này được khởi tạo theo những giá trị hex sau :

A=01234567

B=89abcdef

C=fedcba98

D=76543210

Bước 4 :Xử lý thông điệp theo từng khối 16 từ. Định nghĩa các hàm phụ, các hàm này nhận giá trị đầu vào là 3 từ 32 bit và tạo tạo ra một word 32 bit.

F1(A, B, C) = (A ˄ B) ˅ ((¬A) ˄ C).

F2(A, B, C) = (A ˄ C) ˅ ((¬C) ˄ B).

F3(A, B, C) = A  B  C.

F4(A, B, C) = B  (A ˅ (¬C)).

Quy ước:

A ˄ B Phép toán AND.

A ˅ B Phép toán OR.

A  B Phép toán XOR.

¬A Phép toán NOT.

A <<<s Các bit của X dược dịch chuyển xoay vòng sang trái s vị trí, với (0 ≤ s ≤ 32).

Bảng giá trị bit dịch chuyển mỗi vòng:

s[ 0..15] := { 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22 }

s[16..31] := { 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20 }

s[32..47] := { 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23 }

s[48..63] := { 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21 }

Bước này sử dụng một bảng 64 giá trị K[1 .. 64] được tạo ra từ hàm sin. Gọi K là phần tử thứ i của bảng, thì K là phần nguyên của 4294967296\*|sin(i)| , i được tính theo radian.

Bảng 64 giá trị K theo hàm sin: (4294967296\*|sin(i)|)

K[ 0.. 3] := { d76aa478, e8c7b756, 242070db, c1bdceee }

K[ 4.. 7] := { f57c0faf, 4787c62a, a8304613, fd469501 }

K[ 8..11] := { 698098d8, 8b44f7af, ffff5bb1, 895cd7be }

K[12..15] := { 6b901122, fd987193, a679438e, 49b40821 }

K[16..19] := { f61e2562, c040b340, 265e5a51, e9b6c7aa }

K[20..23] := { d62f105d, 02441453, d8a1e681, e7d3fbc8 }

K[24..27] := { 21e1cde6, c33707d6, f4d50d87, 455a14ed }

K[28..31] := { a9e3e905, fcefa3f8, 676f02d9, 8d2a4c8a }

K[32..35] := { fffa3942, 8771f681, 6d9d6122, fde5380c }

K[36..39] := { a4beea44, 4bdecfa9, f6bb4b60, bebfbc70 }

K[40..43] := { 289b7ec6, eaa127fa, d4ef3085, 04881d05 }

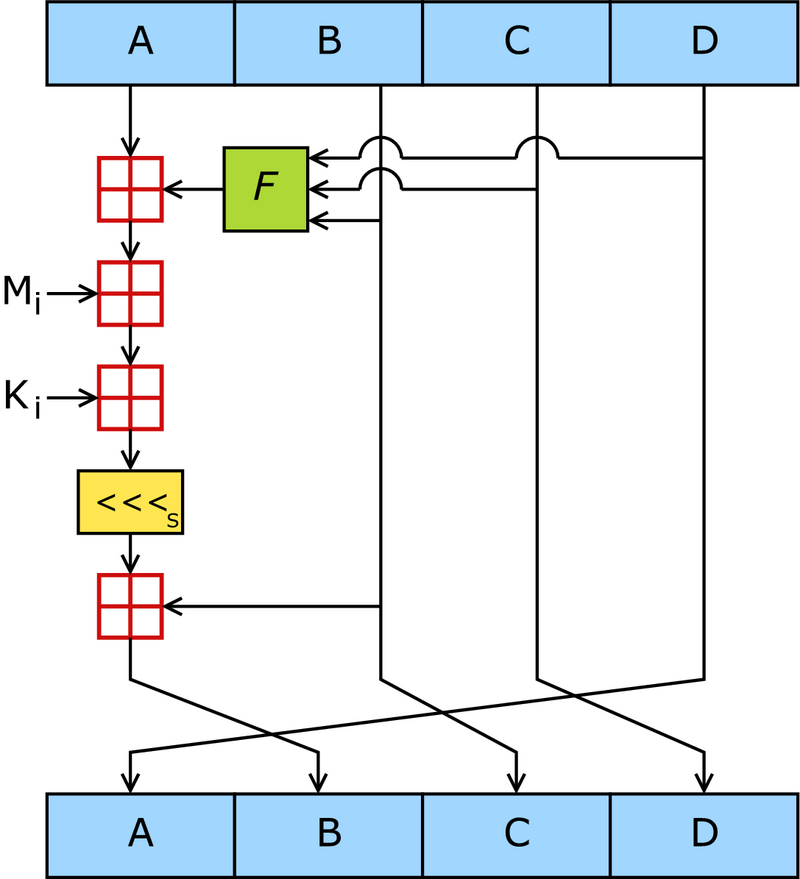
K[44..47] := { d9d4d039, e6db99e5, 1fa27cf8, c4ac5665 }

K[48..51] := { f4292244, 432aff97, ab9423a7, fc93a039 }

K[52..55] := { 655b59c3, 8f0ccc92, ffeff47d, 85845dd1 }

K[56..59] := { 6fa87e4f, fe2ce6e0, a3014314, 4e0811a1 }

K[60..63] := { f7537e82, bd3af235, 2ad7d2bb, eb86d391 }



*Hình 3.16: Sơ đồ vòng lặp chính của MD5*

Đầu tiên, Khối 512 bit sẽ được chia thành 16 khối con (mỗi khối 32 bit). Sau đó, từ 4 thanh ghi ban đầu A0, B0, C0, D0 thuật toán MD5 sẽ xử lý theo sơ đồ như sau:

Vòng 1: Sử dụng hàm F1(A, B, C) = (A ˄ B) ˅ ((¬A) ˄ C) và băm hết 16 khối con của đoạn 512 bit. Cho các thanh ghi B, C, D ban đầu đi qua hàm logic F1 rồi cộng với A (phép cộng module 32). Sau đó cộng tiếp với M0 là khối 32 bit đầu tiên trong dãy 512 bit, rồi cộng tiếp với K0= d76aa478 trong bảng 64 giá trị K tính theo hàm sin. Tiếp theo, ta dịch sang trái s0=7 bit theo bảng dịch bit s (dịch bit theo nhị phân) và tiếp tục cộng với B. Cuối cùng hoán vị các thanh ghi theo sơ đồ (ABCD thành DABC). Ta thu được 4 thanh ghi mới A1, B1, C1, D1 và lại tiếp tục sử dụng để băm tiếp các chuỗi con 32 bit trong đoạn 512 bit.

Vòng 2: Sau khi xử lý hết 16 chuỗi con trong vòng 1. Thuật toán MD5 sẽ tiếp tực đến vòng 2. Cũng tương tự như vòng 1 nhưng vòng 2 sẽ sử dụng hàm F2(A, B, C) = (A ˄ C) ˅ ((¬C) ˄ B) và bắt đầu với chuỗi con M1 với bước nhảy h=5 (M1, M6, M11, M0,…). Vòng 2 sẽ tiếp tục băm với 4 thanh ghi của vòng 1 chứ không băm lại các thanh ghi ban đầu. Cứ như vậy băm hết 16 chuỗi con sẽ thu được các giá trị thanh ghi mới và chuyển sang vòng 3.

Vòng 3: Xử lý tương tự với hàm F3(A, B, C) = A  B  C, bắt đầu từ M5 với bước nhảy h=3, sử dụng các giá trị thanh ghi cuối của vòng 2 để tiếp tục băm. Sau khi băm xong 16 chuỗi cũng sẽ thu được 4 giá trị thanh ghi mới được chuyển sang sử dụng trong vòng 4.

Vòng 4: Cũng là vòng cuối cùng của thuật toán MD5. Tương tự như các vòng trước, sử dụng 4 thanh ghi A, B, C, D kết quả của vòng 3, dùng hàm F4(A, B, C) = B  (A ˅ (¬C)) và bắt đầu bằng M0 với bước nhảy h=7. Sau khi băm hết 16 chuỗi trong vòng 4 ta sẽ nhận được 4 giá trị thanh ghi A, B, C, D cuối cùng. Sau đó, ta cộng với 4 giá trị thanh ghi khởi tạo ban đầu, để thu được 4 chuỗi A, B, C, D được băm hoàn chỉnh. Sử dụng phép toán nối ( || ) ghép A || B || C || D lại ta được một đoạn mã hex 128 bit, đó chính là bản mã hàm băm MD5.

### 3.3.3 Mô hình ứng dụng

Các ứng dụng của MD5:

- Chữ ký điện tử.

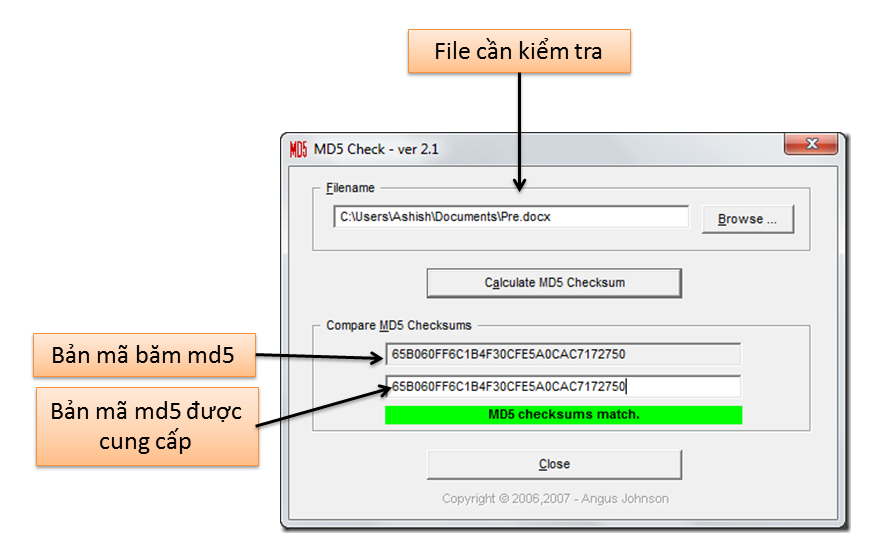
- Dùng trong các ứng dụng bảo mật.

- Kiểm tra tính toàn vẹn của tập tin được truyền đi.

- Lưu trữ mật khẩu.

- Ứng dụng trong các phần mềm để đảm bảo rằng tập tin tải về không bị lỗi.

Hiện tại MD5 được sử dụng rộng rãi để kiểm tra tính toàn vẹn thông tin. Nếu người chủ tập tin có cung cấp đoạn mã MD5, bạn có thể sử dụng phần mềm MD5 Checker để kiểm tra các tệp tin mình tải về có bị lỗi khi tải hay bị can thiệp bởi hacker hay không?



*Hình 3.17: Sử dụng md5 checker để kiểm tra tệp tin.*

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 4.1 Kết quả đạt được

Sau một thời gian tập trung nghiên cứu. Chúng em đã tìm hiểu được một số vấn đề về mã hóa thông tin. Qua đó biết được rằng mã hóa có vai trò không thể thiếu trong lĩnh vực an toàn thông tin mạng. Mã hóa có ứng dụng to lớn trong đời sống xã hội, các lĩnh vực kinh tế, chính trị, quân sự của một quốc gia.

Trong đề tài này chúng em đã tìm hiểu được một số vấn đề liên quan đến mã hoá như sau:

- Tìm hiểu một số khái niệm về mã hóa thông tin

- Các tiêu chuẩn để đánh giá một hệ mã hóa

- Một số phương pháp mã hóa cổ điển như: Caesar, Vigerene, Hoán vị.

- Một số phương pháp mã hóa hiện đại nổi tiếng như: DES, RSA, MD5.

- Xây dựng chương trình mô phỏng sử dụng thuật toán RSA.

- Ứng dụng thuật toán MD5 và RSA trong chữ ký số và toàn vẹn dữ liệu.

## 4.2 Hướng phát triển

Qua quá trình nghiên cứu chúng em biết được các thuật toán mã hóa này đã tạo nên nhiều ứng dụng to lớn như:

- Chữ ký thư điện tử trong doanh nghiệp, ngân hàng,…

- Mã hóa password.

- Chứng thực người dùng.

- Kiểm tra tính toàn vẹn của phần mềm/ dữ liệu khi download,…

Tuy nhiên đây là lĩnh vực khó, đòi hỏi nhiều thời gian và kiến thức sâu. Trong tương lai em sẽ tìm hiểu sâu, rộng hơn về các thuật toán mã hóa để có thể xây dựng thành công một chương trình ứng dụng để mã hóa các tệp văn bản, dữ liệu có kích thước lớn, sáng tạo ra phương thức mã hóa mới an toàn hơn, nhanh chóng hơn để có thể phục vụ cho các hoạt động thi cử hay thậm chí là trong chính trị, quân sự nước nhà. Tìm hiểu thêm một số ngôn ngữ, các phần mềm ứng dụng khác để nâng cao và mở rộng phạm vi ứng dụng, tính bảo mật của các phương pháp mã hóa hiện tại.

## 4.3 Kết luận

Trong quá trình nghiên cứu đề tài “Nghiên cứu và ứng dụng mã hóa”. Em đã được hiểu hơn về các vấn đề mã hóa dữ liệu. Tuy nhiên do thời gian và kiến thức còn hạn chế nên đồ án môn học của chúng em chắc chắn sẽ không tránh khỏi những thiếu sót, chúng em rất mong được sự đóng góp ý kiến của thầy và các bạn để đồ án môn học vủa chúng em được hoàn thiện hơn.

Chúng em xin chân thanh cám ơn.