**I. Giới thiệu và ứng dụng của PGP**

**1. Giới thiệu**

PGP là ký hiệu viết tắt của Pretty Good Privacy (Bảo mật rất mạnh). Đây là một phần mềm mã hóa được thiết kế để đảm bảo tính riêng tư, bảo mật và xác thực cho các hệ thống truyền thông trực tuyến. Phil Zimmerman là cái tên đứng sau chương trình PGP đầu tiên, và theo ông, chương trình này được cung cấp miễn phí do nhu cầu ngày càng cao của xã hội về quyền riêng tư.

Kể từ khi ra đời vào năm 1991, đã có nhiều phiên bản phần mềm PGP được tạo ra. Năm 1997, Phil Zimmerman đã đề xuất với Lực lượng đặc nhiệm kỹ thuật Internet (IETF) để tạo ra một tiêu chuẩn PGP nguồn mở. Đề xuất đã được chấp nhận và dẫn đến việc tạo ra giao thức Open PGP, đây là giao thức xác định các định dạng tiêu chuẩn cho các khóa và thông điệp mã hóa.

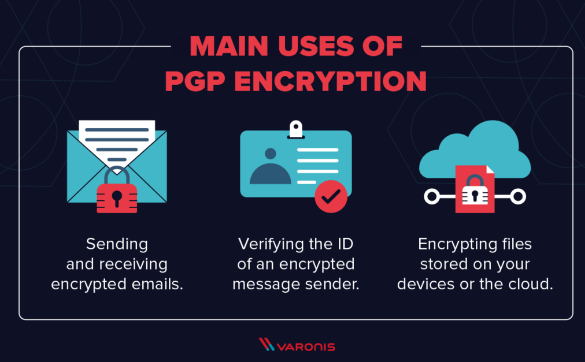
Mặc dù ban đầu chỉ được sử dụng để bảo mật các email và tệp đính kèm, PGP hiện được sử dụng cho nhiều ứng dụng khác nhau, bao gồm chữ ký số, mã hóa toàn bộ đĩa và bảo vệ mạng.

Ban đầu công ty PGP Inc. là chủ sở hữu của PGP, sau đó Network Associates Inc. đã mua lại nó. Vào năm 2010, Symantec Corp đã mua lại PGP với giá 300 triệu đô la, và hiện sử dụng nhãn hiệu này cho các sản phẩm tuân thủ giao thức OpenPGP của họ.

**2. Ứng dụng**

Về cơ bản, có ba cách sử dụng chính của PGP:

* Gửi và nhận email được mã hóa.
* Xác minh danh tính của người đã gửi cho bạn tin nhắn này.
* Mã hóa các tệp được lưu trữ trên thiết bị của bạn hoặc trên đám mây.



Trong ba cách sử dụng này, cách sử dụng đầu tiên - gửi email an toàn - cho đến nay là ứng dụng thống trị của PGP.

**Mã hóa email**

Hầu hết mọi người sử dụng PGP để gửi email được mã hóa. Trong những năm đầu của PGP, nó chủ yếu được sử dụng bởi các nhà hoạt động, nhà báo và những người khác xử lý thông tin nhạy cảm. Trên thực tế, hệ thống PGP ban đầu được thiết kế bởi một nhà hoạt động chính trị và hòa bình tên là Phil Zimmermann, người gần đây đã gia nhập Startpage, một trong những công cụ tìm kiếm tư nhân phổ biến nhất.

Ngày nay, sự phổ biến của PGP đã phát triển đáng kể. Khi ngày càng có nhiều người dùng nhận ra rằng các tập đoàn thông tin và chính phủ của họ đang thu thập bao nhiêu thông tin về họ, thì một số lượng lớn người dân hiện sử dụng tiêu chuẩn này để giữ bí mật thông tin cá nhân của họ.

**Xác minh chữ ký điện tử**

Một cách sử dụng liên quan của PGP là nó có thể được sử dụng để xác minh email. Ví dụ, nếu nhà báo không chắc chắn về danh tính của người gửi tin nhắn cho họ, họ có thể sử dụng Chữ ký số cùng với PGP để xác minh điều này.

Chữ ký điện tử hoạt động bằng cách sử dụng một thuật toán để kết hợp khóa của người gửi với dữ liệu mà họ đang gửi. Điều này tạo ra một “hàm băm”, một thuật toán khác có thể chuyển đổi một thông báo thành một khối dữ liệu có kích thước không đổi. Sau đó, điều này được mã hóa bằng khóa riêng của người gửi.

Người nhận thư sau đó có thể giải mã dữ liệu này bằng khóa công khai của người gửi. Nếu ngay cả một ký tự của tin nhắn đã bị thay đổi khi chuyển tiếp, người nhận sẽ biết. Điều này có thể chỉ ra rằng người gửi không phải là người mà họ nói, rằng họ đã cố gắng giả mạo Chữ ký điện tử hoặc thư đã bị giả mạo.

**Mã hóa tệp**

Công dụng thứ ba của PGP là mã hóa tệp. Bởi vì thuật toán được PGP sử dụng - thường là thuật toán RSA - về cơ bản là không thể phá vỡ, PGP cung cấp một cách mã hóa tệp an toàn cao, đặc biệt là khi được sử dụng cùng với Giải pháp phát hiện và phản hồi mối đe dọa. Trên thực tế, thuật toán này an toàn đến mức nó thậm chí còn được sử dụng trong các phần mềm độc hại cao cấp như phần mềm độc hại CryptoLocker.

Trở lại năm 2010, Symantec mua lại PGP Corp., công ty nắm giữ quyền đối với hệ thống PGP. Kể từ đó, Symantec đã trở thành nhà cung cấp phần mềm mã hóa tệp PGP thống trị thông qua các sản phẩm như Symantec Encryption Desktop và Symantec Encryption Desktop Storage. Phần mềm này cung cấp mã hóa PGP cho tất cả các tệp của bạn, đồng thời che giấu sự phức tạp của các quy trình mã hóa và giải mã.

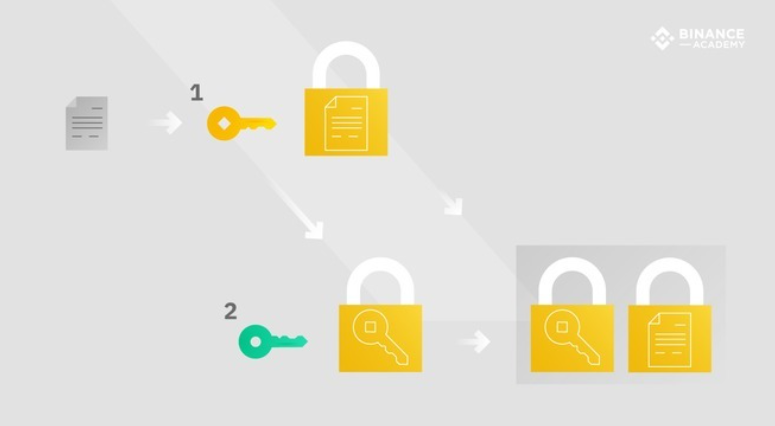
**II. Mô hình hoạt động của PGP**

PGP là một trong những phần mềm được cung cấp rộng rãi đầu tiên để triển khai mật mã khóa công khai. Nó là một hệ thống mật mã kết hợp, sử dụng cả mã hóa đối xứng và mã hóa bất đối xứng để bảo đảm mức độ bảo mật cao.

Trong một quy trình mã hóa văn bản cơ bản, văn bản thuần túy (dữ liệu có thể đọc được) được chuyển đổi thành văn bản mã hóa (dữ liệu không thể đọc được). Nhưng trước khi quá trình mã hóa diễn ra, hầu hết các hệ thống PGP đều thực hiện nén dữ liệu. Bằng cách nén các tệp văn bản thuần túy trước khi gửi chúng, PGP tiết kiệm cả không gian đĩa và thời gian gửi - đồng thời cải thiện tính bảo mật.

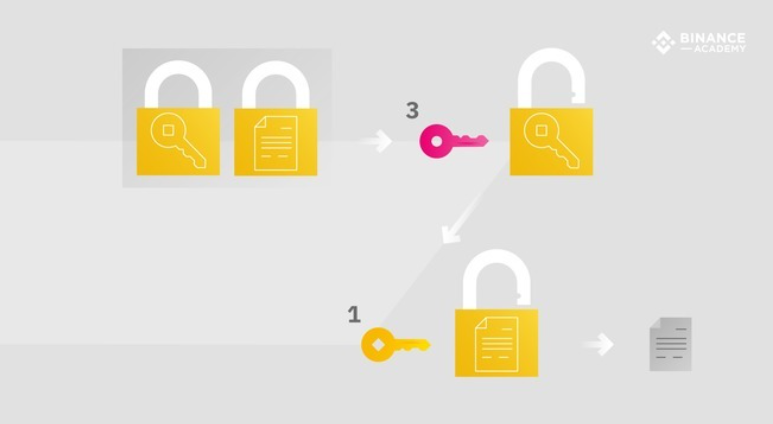
Quá trình mã hóa thực sự bắt đầu sau khi tệp được nén. Ở giai đoạn này, các tệp văn bản thuần túy đã nén được mã hóa bằng khóa sử dụng một lần, được gọi là khóa phiên. Khóa này được tạo ngẫu nhiên thông qua việc sử dụng mật mã đối xứng và mỗi phiên giao tiếp PGP có một khóa phiên duy nhất.

Tiếp theo, chính khóa phiên (1) được mã hóa bằng mã hóa bất đối xứng: người nhận cung cấp khóa công khai (2) của mình cho người gửi tin nhắn để có thể mã hóa khóa phiên. Bước này cho phép người gửi chia sẻ khóa phiên với người nhận một cách an toàn qua Internet, bất kể tình trạng bảo mật.



Việc mã hóa bất đối xứng của khóa phiên thường được thực hiện thông qua việc sử dụng thuật toán RSA. Nhiều hệ thống mã hóa sử dụng RSA, bao gồm giao thức TLS (Transport Layer Security) - một phương thức bảo mật được sử dụng nhiều trên Internet.

Sau khi văn bản mã hóa của tin nhắn và khóa phiên được mã hóa được gửi đi, người nhận có thể sử dụng khóa riêng (3) của mình để giải mã khóa phiên, sau đó khóa phiên này được sử dụng để giải mã văn bản mã hóa trở lại văn bản gốc.

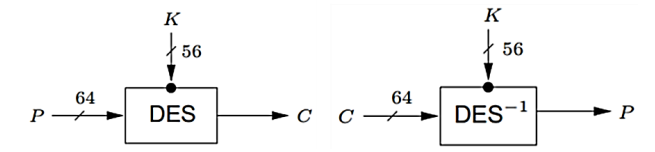


**III. Nền tảng mã hóa khóa bí mật cho PGP**

**1. DES**

1.1. Giới thiệu

DES được phát triển tại công ty IBM vào đầu những năm 1970 và được chấp nhận là chuẩn mã hóa ở Hoa Kỳ vào năm 1977. DES được sử dụng rộng rãi trong những năm 1970 và 1980. DES là dạng mã hóa khối với kích thước khối 64 bit và khóa 64 bit. Mặc dù DES sử dụng khóa 64 bit, chỉ có 56 bit được thực sự sử dụng, 8 bit còn lại dùng cho kiểm tra chẵn lẻ. Vì vậy, 56 bit cũng được gọi là kích thước hiệu dụng của khóa DES.

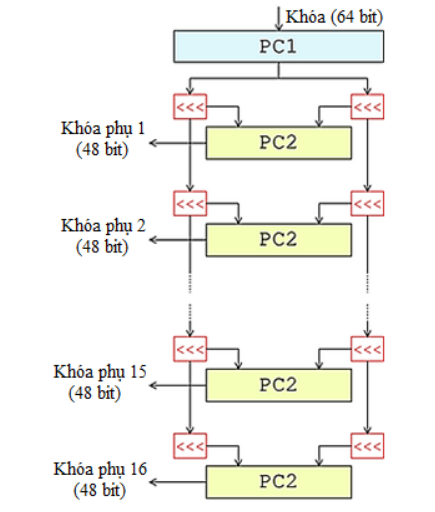


Mô hình DES: các khâu mã hóa và giải mã (hình 4.14)

Một ưu điểm của DES là sử dụng chung một giải thuật cho cả khâu mã hóa và khâu giải mã, như minh họa trên Hình 4.14, trong đó là khối bản rõ 64 bit, là khóa với kích thước hiệu dụng 56 bit, là khối bản mã 64 bit, biểu diễn khâu mã hóa và biểu diễn khâu giải mã. Hiện nay, DES được coi là không an toàn do có không gian khóa nhỏ, dễ bị vét cạn và một lý do khác là tốc độ tính toán của các hệ thống máy tính ngày càng nhanh trong những năm gần đây.

1.2. Thủ tục sinh khóa phụ

DES sử dụng một thủ tục sinh 16 khóa phụ (Subkey) từ khóa chính để sử dụng trong 16 vòng lặp hàm Feistel - là hàm xử lý khối dữ liệu của DES. Hình 4.15 mô tả thủ tục sinh 16 khóa phụ từ khóa chính của DES.



Thủ tục sinh các khóa phụ từ khóa chính của DES (hình 4.15)

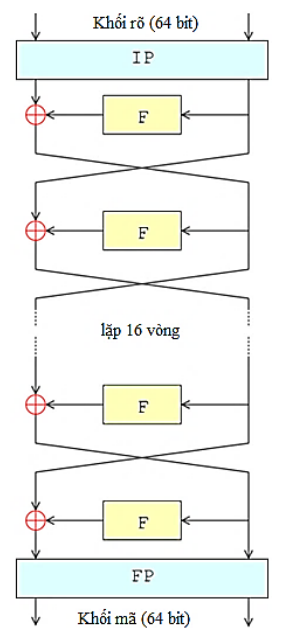
Thủ tục sinh khóa phụ DES gồm các bước xử lý chính như sau:

* Bước 1: 56 bit khóa được chọn từ khóa gốc 64 bit bởi khâu PC1 (Permuted Choice 1 - Hoán vị lựa chọn 1). 8 bit còn lại được hủy hoặc dùng để kiểm tra chẵn lẻ.
* Bước 2: 56 bit được chia thành 2 phần 28 bit, mỗi phần được xử lý riêng.
* Bước 3: Mỗi phần được quay trái 1 hoặc 2 bit.
* Bước 4: Hai phần 28 bit được ghép lại và 48 bit được chọn làm Khóa phụ 1 bởi khâu PC2.
* Bước 5: Lặp lại các bước 3 và 4 để tạo 15 khóa phụ còn lại.

1.3. Mã hóa khối bản rõ

Với mỗi khối bản rõ 64 bit, DES thực hiện 3 bước xử lý như minh họa trên Hình 4.16 để chuyển khối bản rõ thành khối bản mã 64 bit tương ứng. Các bước cụ thể gồm:

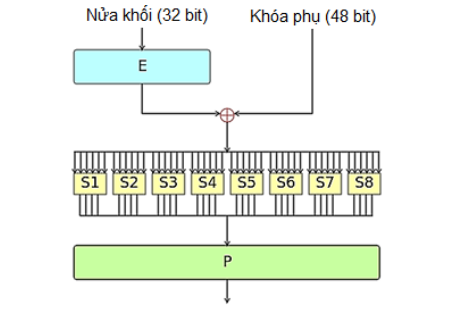
* Bước 1: Hóa vị khởi tạo (IP - Initial Permutation).
* Bước 2: 16 vòng lặp chính thực hiện xáo trộn dữ liệu sử dụng hàm Feistel (F). Trong mỗi vòng lặp, một khóa phụ (được tạo theo thủ tục ở trên) được sử dụng. Sau mỗi vòng lặp, các kết quả trung gian được kết hợp lại sử dụng phép (XOR).
* Bước 3: Hoán vị kết thúc (FP - Final Permutation) để tạo khối bản mã đầu ra.



Các bước xử lý chuyển khối rõ 64 bit thành khối mã 64 bit của DES (hình 4.16)

Hàm Feistel là hạt nhân trong các vòng lặp xử lý dữ liệu của DES. Trước hết, mỗi khối dữ liệu 64 bit được chia thành 2 khối con 32 bit và được xử lý lần lượt. Hàm Feistel được thực hiện trên mỗi khối dữ liệu 32 bit như biểu diễn trên Hình 4.17 gồm 4 bước xử lý như sau:

* Bước E (Expansion - Mở rộng) thực hiện mở rộng 32 bit khối đầu vào thành 48 bit bằng cách nhân đôi một nửa số bit.
* Bước trộn khối 48 bit kết quả ở bước E với khóa phụ 48 bit.
* Bước Si (Substitution - Thay thế) chia khối dữ liệu 48 bit thành 8 khối 6 bit và được chuyển vào các bộ thay thế (S1 - S8). Mỗi bộ thay thế Si sử dụng phép chuyển đổi phi tuyến tính để chuyển 6 bit đầu vào thành 4 bit đầu ra theo bảng tham chiếu. Các bộ thay thế là thành phần nhân an ninh của DES.
* Bước P (Permutation - Hoán vị) sắp xếp khối 32 bit đầu ra từ các bộ thay thế bằng phép hoán vị cố định cho ra đầu ra 32 bit.



Các bước xử lý của hàm Feistel (F) (hình 4.17)

1.4. Giải mã khối bản mã

Như đã đề cập, giải thuật DES có thể được sử dụng cho cả khâu mã hóa và khâu giải mã. Trong khâu giải mã, các bước xử lý tương tự khâu mã hóa. Tuy nhiên, các khóa phụ dùng cho các vòng lặp được sử dụng theo trật tự ngược lại: khóa phụ số 16, 15, …, 2, 1 tương ứng được sử dụng cho các vòng lặp số 1, 2, …, 15, 16.

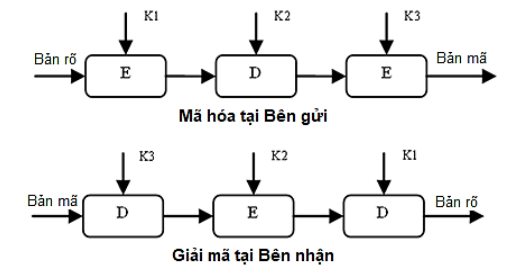
**2. 3-DES**

3-DES hay Triple DES có tên đầy đủ là Triple Data Encryption Algorithm (TDEA). 3-DES ra đời nhằm khắc phục vấn đề không gian khóa nhỏ của DES. Giải thuật 3-DES được phát triển từ giải thuật DES bằng cách áp dụng DES 3 lần cho mỗi khối dữ liệu đầu vào 64 bit. 3-DES sử dụng một bộ gồm 3 khóa DES: K1, K2, K3, trong đó mỗi khóa kích thước hiệu dụng là 56 bit. 3-DES cho phép lựa chọn các bộ khóa như sau:

* Lựa chọn 1: Cả 3 khóa độc lập và tổng kích thước bộ khóa là 168 bit.
* Lựa chọn 2: Các khóa K1 và K2 độc lập, còn K3 = K1 và tổng kích thước bộ khóa là 112 bit.
* Lựa chọn 3: 3 khóa giống nhau, K1 = K2 = K3 và tổng kích thước bộ khóa giữ nguyên là 56 bit.

Hình 4.18 biểu diễn quá trình mã hóa và giải mã với giải thuật 3-DES, trong đó khâu mã hóa được ký hiệu là E và khâu giải mã được ký hiệu là D. Theo đó, ở bên gửi, bản rõ được mã hóa bằng khóa K1, giải mã bằng khóa K2 và mã hóa bằng khóa K3 để cho ra bản mã. Ở bên nhận, quá trình giải mã bắt đầu bằng việc giải mã bằng khóa K3, sau đó mã hóa bằng khóa K2 và cuối cùng giải mã bằng khóa K1 để khôi phục bản rõ.

Ưu điểm của 3-DES là nâng cao được độ an toàn so với DES nhờ tăng kích thước khóa. Ngoài ra, do thành phần chính của 3-DES là DES nên có thể tái sử dụng các mô đun cài đặt DES trong quá trình triển khai ứng dụng 3-DES. Tuy nhiên, nhược điểm chính của 3-DES là tốc độ thực thi chậm do phải thực hiện DES lặp 3 lần cho mỗi khâu mã hóa và giải mã.



Mã hóa và giải mã với giải thuật 3-DES (hình 4.18)

**3. AES**

3.1. Giới thiệu

AES (Advanced Encryption Standard) là một chuẩn mã hóa dữ liệu được Viện Tiêu chuẩn và Công nghệ Hoa Kỳ công nhận năm 2001. AES được xây dựng dựa trên hệ mã hóa Rijndael được phát triển và công bố năm 1998 bởi 2 nhà mật mã học người Bỉ là Joan Daemen và Vincent Rijmen. AES là dạng mã hóa khối, với khối dữ liệu có kích thước 128 bit và khóa bí mật với kích thước có thể là 128, 192, hoặc 256 bit. AES được thiết kế dựa trên mạng hoán vị-thay thế (Substitution-permutation network) và nó có thể cho tốc độ thực thi cao khi cài đặt trên cả phần cứng và phần mềm. Đặc biệt, giải thuật AES đã được tích hợp vào các bộ vi xử lý gần đây của hãng Intel dưới dạng tập lệnh AES-NI, giúp tăng đáng kể tốc độ thực thi các thao tác mã hóa và giải mã dựa trên AES.

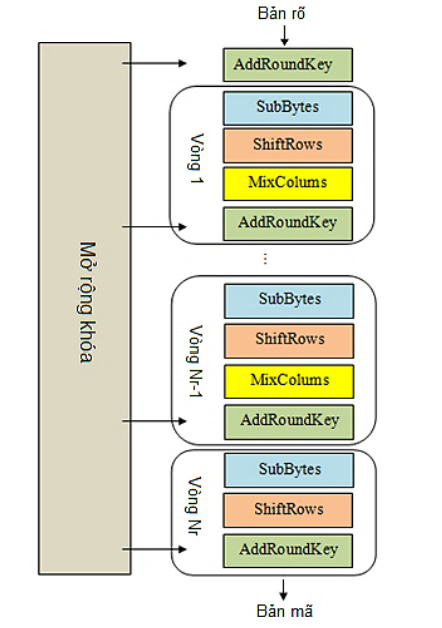
AES vận hành dựa trên một ma trận vuông 4x4, được gọi là *state* (trạng thái). Ma trận này gồm 16 phần tử, mỗi phần tử là 1 byte dữ liệu. State được khởi tạo là khối 128 bit bản rõ và qua quá trình biến đổi sẽ chứa khối 128 bit bản mã ở đầu ra. Như đã đề cập, AES hỗ trợ 3 kích thước khóa và kích thước của khóa quyết định số vòng lặp cần thực hiện để chuyển đổi bản rõ thành bản mã. Số vòng lặp AES cần thực hiện theo kích thước khóa như sau:

* 10 vòng lặp với khóa 128 bit.
* 12 vòng lặp với khóa 192 bit.
* 14 vòng lặp với khóa 256 bit.

3.2. Quá trình mã hóa

Giải thuật AES thực hiện mã hóa khối dữ liệu bản rõ, như minh họa trên Hình 4.19 gồm các bước xử lý chính như sau:

* Bước 1: Mở rộng khóa thực hiện việc sinh các khóa vòng (Round key) dùng trong các vòng lặp từ khóa chính AES sử dụng thủ tục sinh khóa Rijndael.
* Bước 2: Vòng khởi tạo thực hiện hàm AddRoundKey, trong đó mỗi byte trong *state* được kết hợp với khóa vòng sử dụng phép XOR.
* Bước 3: Các vòng lặp chính, trong đó mỗi vòng thực hiện 4 hàm biến đổi dữ liệu như sau:
* SubBytes là hàm thay thế phi tuyến tính, trong đó mỗi byte trong *state* được thay thế bằng một byte khác sử dụng bảng tham chiếu S-box.
* ShiftRows là hàm dịch dòng, trong đó mỗi dòng trong *state* được dịch một số bước theo chu kỳ.
* MixColumns là hàm trộn các cột trong *state*, kết hợp 4 bytes trong mỗi cột.
* AddRoundKey là hàm kết hợp *state* với khóa vòng sử dụng phép XOR.
* Bước 4: Vòng lặp cuối tương tự các vòng lặp chính, nhưng chỉ thực hiện 3 hàm biến đổi dữ liệu, bao gồm SubBytes, ShiftRows và AddRoundKey.

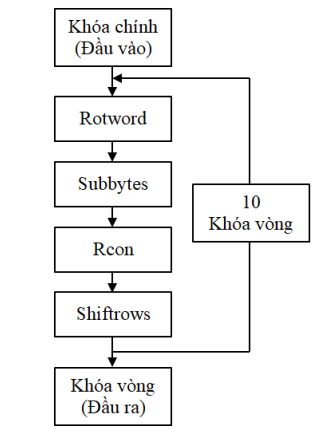


Các bước xử lý mã hóa dữ liệu của AES (hình 4.19)

3.3. Mở rộng khóa

Bước mở rộng khóa AES sinh các khóa vòng cho các vòng lặp xử lý dữ liệu sử dụng thủ tục sinh khóa Rijndael như biểu diễn trên Hình 4.20. Thủ tục sinh khóa Rijndael nhận đầu vào là khóa chính AES và xuất ra một khóa vòng sau mỗi vòng lặp. Một vòng lặp của thủ tục sinh khóa Rijndael gồm các khâu sau:

* Rotword thực hiện quay trái 8 bit từng từ 32 bit lấy từ khóa chính.
* SubBytes là hàm thay thế phi tuyến tính các byte trong khóa tương tự hàm SubBytes của quá trình mã hóa AES.
* Rcon thực hiện việc tính toán giá trị:
* ShiftRows là hàm dịch vòng tương tự hàm ShiftRows của quá trình mã hóa AES.

1. 

Thủ tục sinh khóa Rijndael (hình 4.20)

3.4. Các hàm xử lý dữ liệu

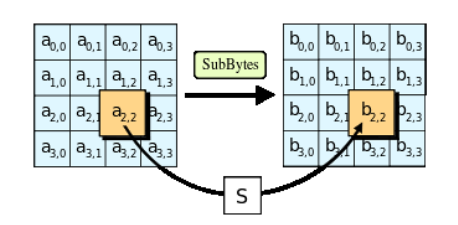
Như đã đề cập, AES sử dụng 4 hàm xử lý để biến đổi dữ liệu, bao gồm SubBytes, ShiftRows, MixColumns và AddRoundKey. Mục này trình bày chi tiết về các hàm này.

Hàm *SubBytes* thay thế mỗi byte trong ma trận *state* bởi 1 byte trong Rijndael S-box hay như minh họa trên Hình 4.21. S-box là một bảng tham chiếu phi tuyến tính, được tạo ra bằng phép nhân nghịch đảo một số cho trước trong trường .

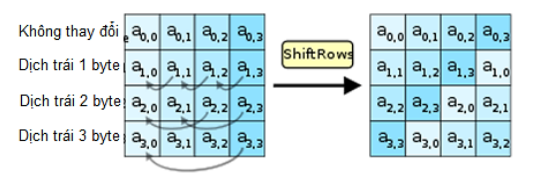
Hàm *ShiftRows* dịch các dòng của ma trận *state* theo chu kỳ sang trái theo nguyên tắc: hàng số 0 giữ nguyên, hàng số 1 dịch 1 byte sang trái, hàng số 2 dịch 2 byte sang trái và hàng số 3 dịch 3 byte sang trái, như minh họa trên Hình 4.22.

Hàm *MixColumns* nhân mỗi cột của ma trận *state* với một đa thức , như minh họa trên Hình 4.23. Đa thức này được tính theo công thức: .

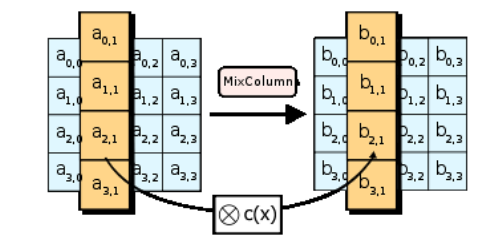
Hàm *AddRoundKey* kết hợp mỗi byte của ma trận *state* với một byte tương ứng của khóa vòng sử dụng phép (XOR), như minh họa trên Hình 4.24.



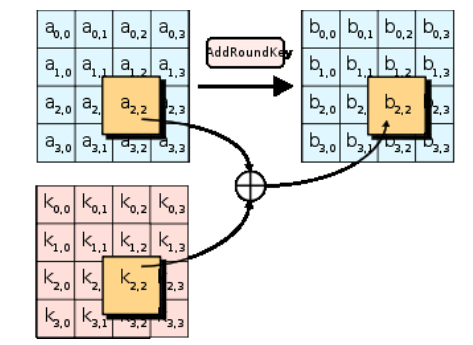
Hàm SubBytes sử dụng Rijndael S-box (hình 4.21)



Hàm ShiftRows (hình 4.22)



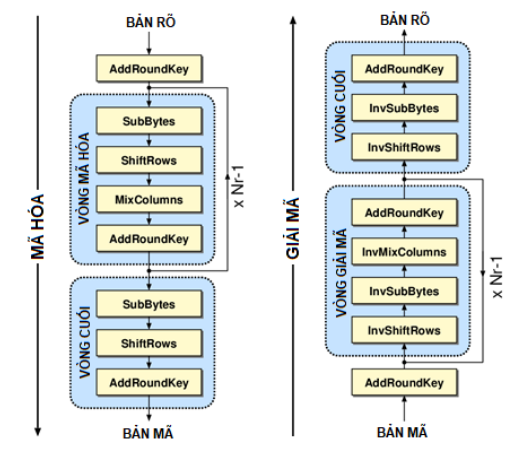
Hàm MixColumns (hình 4.23)



Hàm AddRoundKey (hình 4.24)

3.5. Quá trình giải mã

Quá trình giải mã trong AES cũng gồm các bước xử lý tương tự như quá trình mã hóa. Hình 4.25 biểu diễn quá trình giải mã đối sánh với quá trình mã hóa trong AES. Theo đó, ngoài bước Mở rộng khóa, quá trình giải mã gồm Vòng khởi tạo thực thi hàm AddRoundKey, Các vòng giải mã và Vòng cuối để chuyển đổi khối mã thành khối rõ.



Quá trình mã hóa và giải mã trong AES (hình 4.25)

Điểm khác biệt chính của quá trình giải mã so với quá trình mã hóa là các hàm đảo được sử dụng, bao gồm InvSubBytes, InvShiftRows và InvMixColumns tương ứng thay cho các hàm SubBytes, ShiftRows và MixColumns. Hàm InvSubBytes thay thế phi tuyến tính các byte của ma trận *state* sử dụng bảng *S-box đảo*, hàm InvShiftRows dịch các byte của từng dòng trong ma trận *state* sang phải theo cùng nguyên tắc với ShiftRows và hàm InvMixColumns nhân các cột của ma trận *state* với đa thức  *đảo*.

**IV. Nền tảng mã hóa khóa công khai cho PGP**

**1. Diffie-Hellman**

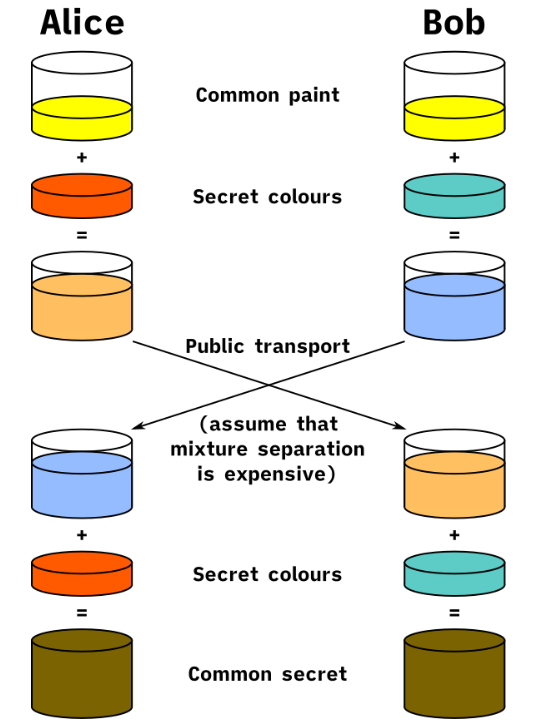
Trao đổi khóa Diffie-Hellman (D-H) là một phương thức trao đổi khóa được phát minh sớm nhất trong mật mã học. Phương pháp trao đổi khóa Diffie-Hellman cho phép hai bên (người, thực thể giao tiếp) thiết lập một khóa bí mật chung để mã hóa dữ liệu sử dụng trên kênh truyền thông không an toàn mà không cần sự thỏa thuận trước về khóa bí mật giữa hai bên. Khóa bí mật tạo ra sẽ được sử dụng để mã hóa dữ liệu với phương thức mã hóa khóa đối xứng. Sơ đồ sau đây minh họa ý tưởng cơ bản của việc trao đổi khóa thông qua ví dụ về màu sơn.

1.1. Ý tưởng cơ bản

Điểm chủ chốt của ý tưởng này là Alice và Bob trao đổi màu sơn bí mật thông qua hỗn hợp sơn.

* Đầu tiên Alice và Bob trộn màu đã biết chung (màu vàng) với màu bí mật riêng của mỗi người.
* Sau đó, mỗi người chuyển hỗn hợp của mình tới người kia thông qua một kênh vận chuyển công cộng.
* Khi nhận được hỗn hợp của người kia, mỗi người sẽ trộn thêm màu bí mật của riêng mình và nhận được hỗn hợp cuối cùng.

Hỗn hợp sơn cuối cùng là hoàn toàn giống nhau cho cả hai người và chỉ có riêng hai người biết. Mấu chốt ở đây là đối với một người ngoài sẽ rất khó (về mặt tính toán) cho họ để tìm ra được bí mật chung của hai người (nghĩa là hỗn hợp cuối cùng). Alice và Bob sẽ sử dụng bí mật chung này để mã hóa và giải mã dữ liệu truyền trên kênh công cộng. Lưu ý, màu sơn đầu tiên (màu vàng) có thể tùy ý lựa chọn, nhưng được thỏa thuận trước giữa Alice và Bob. Màu sơn này cũng có thể được giả sử là không bí mật đối với người thứ ba mà không làm lộ bí mật chung cuối cùng của Alice và Bob.



1.2. Mô tả thuật toán

Để đơn giản và thực tế thực hiện thuật toán, chúng ta sẽ chỉ xem xét 4 biến, một số nguyên tố và (một gốc nguyên tố của ) và hai giá trị riêng và . và đều là những con số có sẵn công khai. Người dùng (giả sử Alice và Bob) chọn các giá trị riêng tư và , sau đó họ tạo khóa và trao đổi công khai. Người đối diện nhận được khóa và điều đó tạo ra khóa bí mật, sau đó họ có cùng khóa bí mật để mã hóa.

**Giải thích từng bước:**

|  |  |
| --- | --- |
| Alice | Bob |
| Các thông tin công cộng khả dụng: , . | Các thông tin công cộng khả dụng: , . |
| Khóa cá nhân được chọn: . | Khóa cá nhân được chọn: |
| Khóa được tạo: . | Khóa được tạo: . |
| *Trao đổi các khóa đã tạo diễn ra.* | |
| Khóa nhận được: . | Khóa nhận được: . |
| Khóa bí mật đã tạo: . | Khóa bí mật đã tạo: . |
| *Về mặt đại số, có thể chỉ ra rằng: .* | |
| *Người dùng hiện có một khóa bí mật đối xứng để mã hóa.* | |

**2. RSA**

2.1. Giới thiệu

Giải thuật mã hóa RSA được 3 nhà khoa học người Mỹ là R. Rivest, A. Shamir và L. Adleman phát minh năm 1977 và tên giải thuật RSA được đặt theo chữ cái đầu của tên 3 đồng tác giả. Độ an toàn của RSA dựa trên tính khó của việc phân tích số nguyên rất lớn, với độ lớn cỡ hàng trăm chữ số thập phân. RSA sử dụng một cặp khóa, trong đó khóa công khai dùng để mã hóa và khóa riêng dùng để giải mã. Chỉ khóa riêng RSA cần giữ bí mật, còn khóa công khai có thể công bố rộng rãi.

Hiện nay, các khóa RSA có kích thước nhỏ hơn 1024 bit được coi là không an toàn do tốc độ các hệ thống máy tính tăng nhanh. Để đảm bảo an toàn, khuyến nghị sử dụng khóa 2048 bit trong giai đoạn 2010 - 2020. Trong tương lai, cần sử dụng RSA có kích thước lớn hơn, chẳng hạn 3072 bit.

RSA hiện là một trong các giải thuật mã hóa khóa bất đối xứng được sử dụng rộng rãi nhất trên thực tế: RSA có thể được sử dụng để mã hóa thông điệp, và để tạo và kiểm tra chữ ký số.

2.2. Sinh khóa

RSA cung cấp một thủ tục sinh cặp khóa, gồm khóa công khai và khóa riêng tương đối đơn giản. Cụ thể, thủ tục sinh khóa gồm các bước như sau:

* Tạo 2 số nguyên tố và .
* Tính modulo .
* Tính .
* Chọn số nguyên tố sao cho 0 < < và gcd(, ) = 1, trong đó hàm gcd() tính ước số chung lớn nhất của 2 số nguyên. Do gcd(, ) = 1 nên và là 2 số nguyên tố cùng nhau.
* Chọn số nguyên sao cho , hoặc hay là modulo nghịch đảo của .
* Ta có là khóa công khai, là khóa riêng và còn được gọi là modulo.

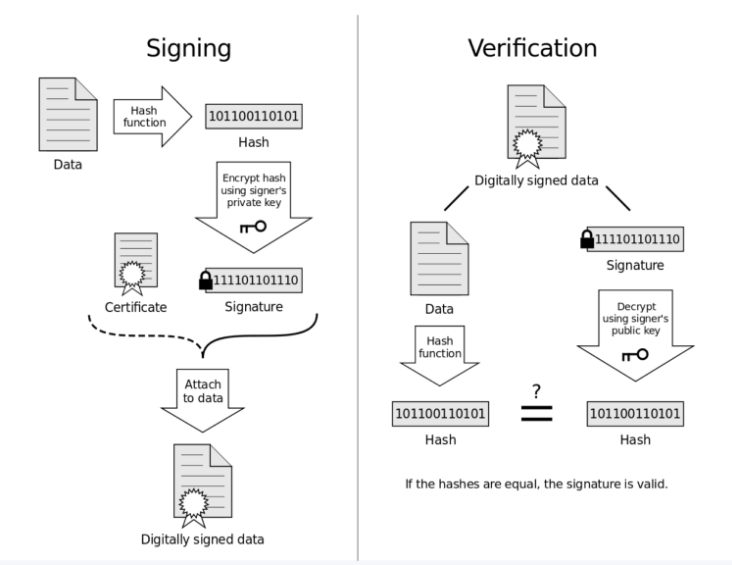
2.3. Mã hóa và giải mã

* Mã hóa:
* Thông điệp bản rõ đã được chuyển thành số, với . Nếu thông điệp bản rõ có kích thước lớn thì được chia thành các khối , với .
* Bản mã .
* Giải mã:
* Bản mã , với .
* Bản rõ .

**V. Chữ ký số cho PGP**

**1. Tổng quan**

Mật mã khóa công khai không chỉ được sử dụng để bảo mật (tức là để bảo vệ thông điệp để người nhận dự định chỉ đọc) mà còn để xác thực (nghĩa là xác minh rằng thông điệp đến từ người gửi chỉ định) và tính toàn vẹn (tức là để đảm bảo rằng thông điệp không bị thay đổi khi chuyển tiếp). Xác thực và tính toàn vẹn được thực thi bằng cách thêm chữ ký số vào thông điệp.

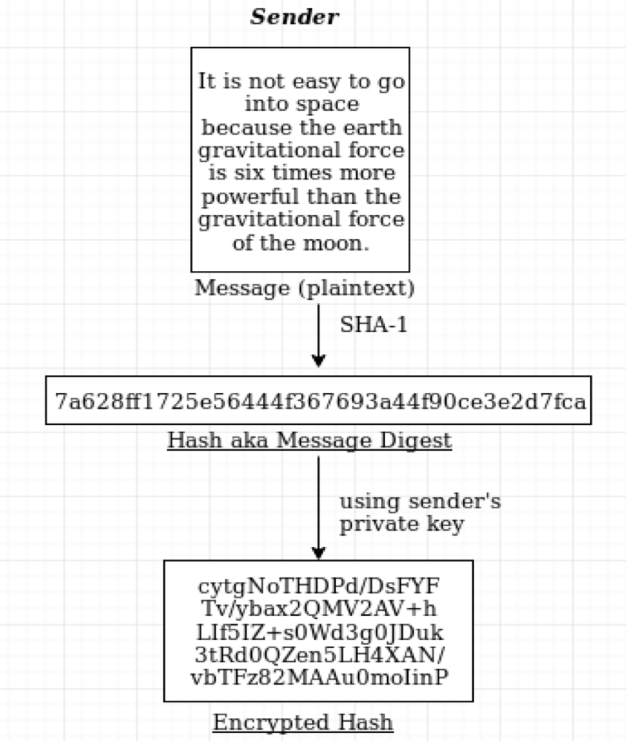


Chữ ký số là một chuỗi các bit được tạo ra bởi một thuật toán sử dụng ***hàm băm*** kết hợp với một khóa. Hàm băm là một hàm nhận đầu vào là một thông điệp có độ dài bất kỳ và xuất ra một chuỗi có độ dài nhỏ cố định được gọi là ***chuỗi đại diện (digest)***, đây là một phần chắt lọc của thông điệp được đưa vào đầu vào. Các tính năng đáng chú ý của hàm băm bao gồm rằng thực tế không thể lấy đầu vào từ đầu ra và chỉ cần thay đổi một bit đầu vào sẽ tạo ra một đầu ra hoàn toàn khác.

Chữ ký số sử dụng mã băm hoặc thuật toán tóm tắt thông điệp, và một thuật toán chữ ký khóa công khai theo trình tự như sau:

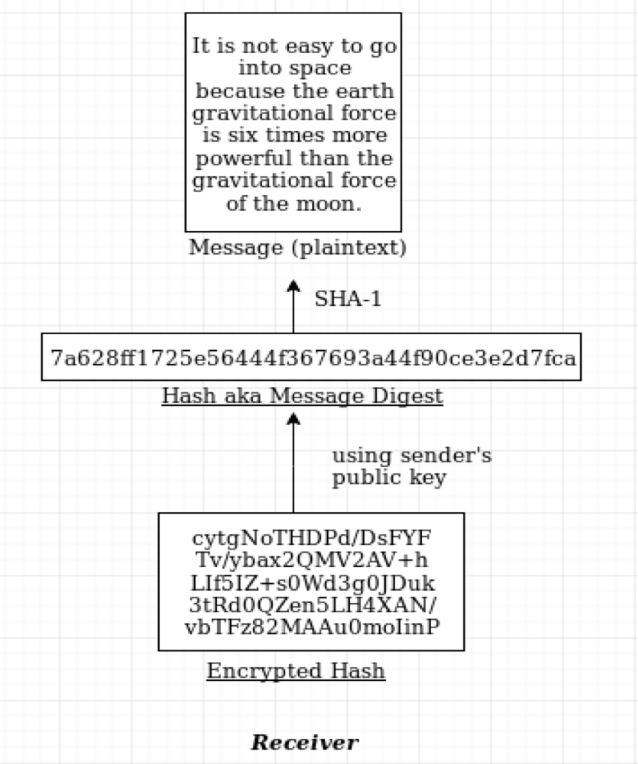
* Người gửi tạo thông điệp.
* Phần mềm gửi tạo mã băm của thông điệp.
* Phần mềm gửi tạo chữ ký từ mã băm bằng khóa bí mật của người gửi.
* Chữ ký dưới dạng nhị phân được đính kèm với thông điệp.
* Phần mềm nhận sẽ giữ một bản sao của chữ ký của thông điệp.
* Phần mềm nhận sẽ tạo một mã băm mới cho thông điệp đã nhận và xác minh nó bằng cách sử dụng chữ ký của thông điệp. Nếu xác minh thành công, thông điệp được chấp nhận là xác thực.

**Cách chữ ký số hoạt động:**



Người gửi tạo **mã băm** của thông điệp văn bản rõ sẽ được gửi. *Hàm băm* là một thuật toán được sử dụng để chuyển đổi chuỗi ký tự thành chuỗi giá trị ngắn hơn (SHA-1 tạo ra 40 ký tự) có độ dài cố định được gọi là **chuỗi đại diện**. *Chuỗi đại diện* này sau đó được mã hóa bằng khóa bí mật của người gửi và được gửi cùng với thông điệp văn bản gốc. Điều này cùng với các thông tin khác như thuật toán băm được sử dụng, tạo nên chữ ký số.

Việc băm thường nhanh hơn mã hóa, đó là lý do tại sao trước tiên chúng ta băm thông điệp và sau đó mã hóa băm.



Sau đó, người nhận giải mã mã băm bằng cách sử dụng khóa công khai của người gửi, khóa này cung cấp cho họ một mã băm được giải mã. Sau đó, người nhận tạo ra mã băm của thông điệp ban đầu nhận được. Nếu mã băm này giống với mã băm đã được giải mã nhận được thì chữ ký số được cho là đã được xác minh.

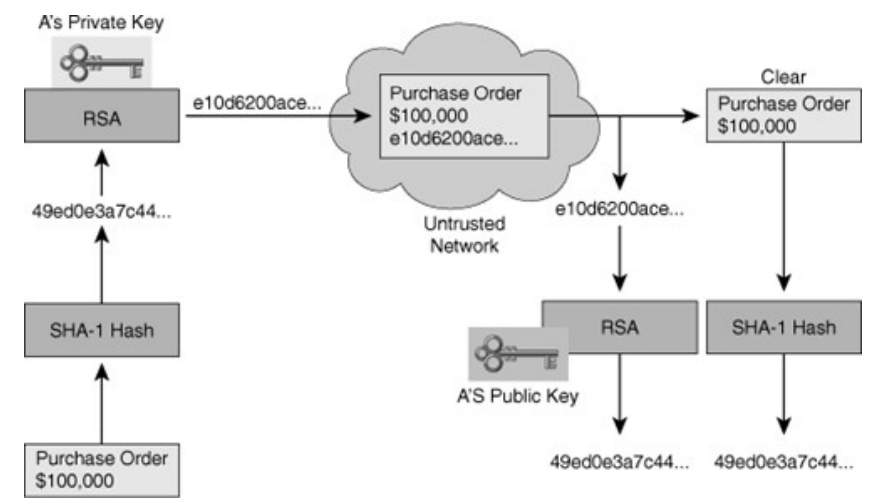
Bây giờ nếu thông điệp bị thay đổi trong quá trình truyền (thậm chí là một ký tự) thì giá trị băm sẽ hoàn toàn khác và không thể xác minh chữ ký số.

Chữ ký số hoạt động bằng cách sử dụng khóa bí mật để mã hóa và khóa công khai để giải mã (ngược lại với mã hóa) vì khóa bí mật chỉ dành cho người gửi ban đầu (người đã tạo khóa công khai) và không ai có thể lấy được điều đó (rõ ràng đó là lý do tại sao nó riêng tư). Điều này xác minh rằng thư ban đầu đã được nhận từ cùng một nguồn.

**2. Các thuật toán chữ ký số**

2.1. Chữ ký số RSA

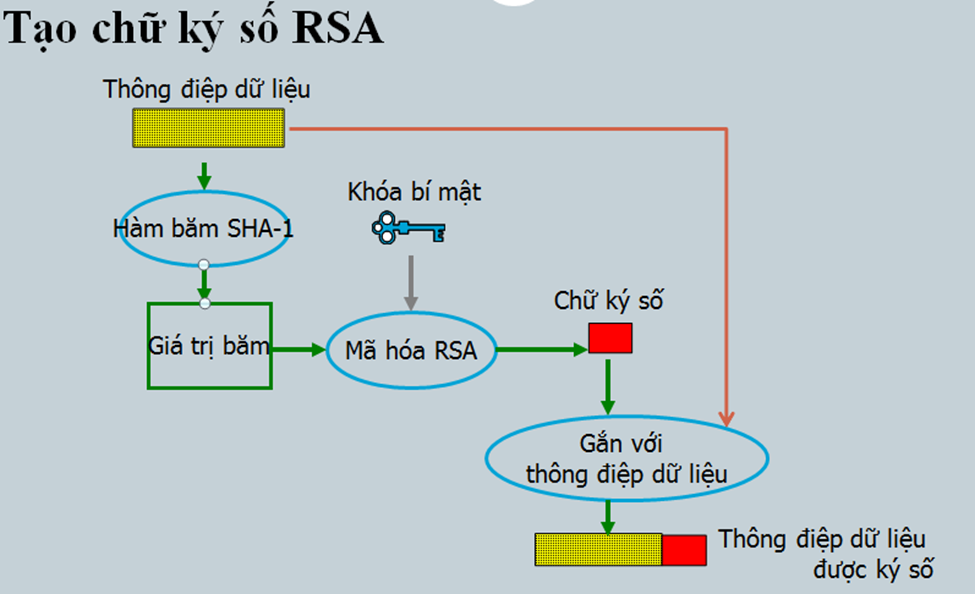
RSA là giải thuật cho phép thực hiện hai tính năng: mã hóa thông điệp và tạo chữ ký số. Việc tạo chữ ký số tương tự như quá trình mã hóa và giải mã ở trên, tuy nhiên vai trò của khóa công khai và bí mật thì có thay đổi đôi chút. Khi mã hóa thông điệp, người gửi mã hóa thông điệp sử dụng khóa công khai của người nhận và người nhận giải mã thông điệp sử dụng khóa bí mật của mình. Còn khi tạo chữ ký số, người gửi tạo chữ ký số sử dụng khóa bí mật của mình và người nhận kiểm tra chữ ký sử dụng khóa công khai của người gửi.

****

Kiến trúc chữ ký số RSA

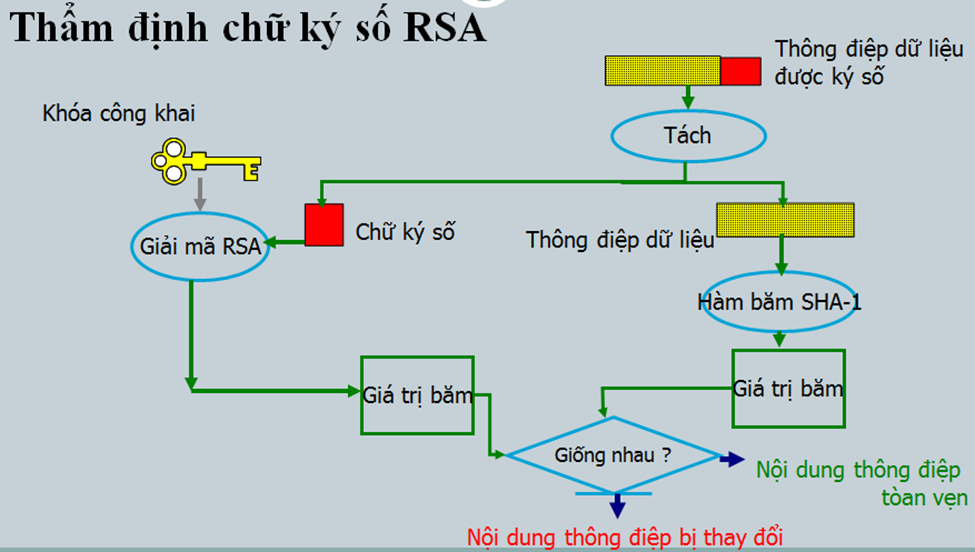
**Quá trình ký (bên gửi)**

* Tính toán chuỗi đại diện của thông điệp sử dụng một giải thuật băm SHA-1.
* Chuỗi đại diện được ký sử dụng khóa bí mật của người gửi và giải thuật tạo chữ ký RSA. Kết quả chữ ký số của thông điệp hay còn gọi là chuỗi đại diện được mã hóa bởi giải thuật RSA.
* Thông điệp ban đầu được ghép với chữ ký số tạo thành thông điệp đã được ký.
* Thông điệp đã được ký được gửi cho người nhận.



**Quá trình kiểm tra chữ ký (bên nhận)**

* Tách chữ ký số RSA và thông điệp gốc khỏi thông điệp đã ký để xử lý riêng.
* Tính toán chuỗi đại diện MD1 của thông điệp gốc sử dụng giải thuật băm.
* Sử dụng khóa công khai của người gửi để giải mã chữ ký RSA → chuỗi đại diện thông điệp MD2.
* So sánh MD1 và MD2:
* Nếu MD1 = MD2 → chữ ký kiểm tra thành công. Thông điệp đảm bảo tính toàn vẹn và thực sự xuất phát từ người gửi (do khóa công khai được chứng thực).
* Nếu MD1 MD2 → chữ ký không hợp lệ. Thông điệp có thể đã bị sửa đổi hoặc không thực sự xuất phát từ người gửi.



2.2. Chữ ký số DSA

**Giải thuật ký số** (*Digital Signature Algorithm*, viết tắt *DSA*) là chuẩn của chính phủ Mỹ hoặc FIPS cho các chữ ký số. Giải thuật này được đề nghị bởi Viện các tiêu chuẩn và công nghệ quốc gia (NIST) vào tháng 8/1991 để sử dụng trong chuẩn chữ ký số (DSS), được chỉ ra trong *FIPS 186*, được chấp nhận năm 1993. Một sửa đổi nhỏ được đưa ra năm 1996 trong FIPS 186-1, chuẩn được mở rộng hơn năm 2000, được xem như FIPS 186-2.

Các thành phần của DSA bao gồm:

* Sinh khóa: sinh cặp khóa. Gồm hai giai đoạn:
* Lựa chọn tham số của giải thuật.
* Sinh cặp khóa cho người dùng.
* Quá trình ký: ký thông điệp.
* Quá trình kiểm tra chữ ký: kiểm tra chữ ký.

2.2.1. Sinh khóa

**Lựa chọn tham số**

* Lựa chọn giải thuật băm chuẩn . Giải thuật băm có thể được lựa chọn là SHA-1 hoặc SHA-2.
* Chọn kích thước cho các khóa và :
* có thể là 1024, 2048, 3072.
* có thể là 160, 224, 256. phải nhỏ hơn hoặc bằng kích thước chuỗi băm đầu ra của hàm đã chọn.
* Chọn số nguyên tố bit.
* Chọn modulo bit sao cho là bội số của .
* Chọn là hệ số nhân sao cho .
* Các tham số (, và ) được chia sẻ giữa các người dùng.

**Sinh khóa cho một người dùng**

* Chọn số ngẫu nhiên sao cho .
* Tính .
* Khóa công khai là (, , , ).
* Khóa bí mật là .

2.2.2. Ký thông điệp

* là hàm băm sử dụng và là thông điệp gốc.
* Tính từ thông điệp gốc.
* Tạo số ngẫu nhiên cho mỗi thông điệp, .
* Tính .
* Nếu , chọn một mới và tính lại .
* Tính .
* Nếu , chọn một mới và tính lại và .
* Chữ ký là cặp .

2.2.3. Kiểm tra chữ ký của thông điệp

* Loại bỏ chữ ký nếu và không thỏa mãn , .
* Tính từ thông điệp nhận được.
* Tính .
* Tính .
* Tính .
* Tính .
* Chữ ký là xác thực nếu .

VI. Cài đặt demo PGP đảm bảo tính bí mật và toàn vẹn trong truyền file, hoặc email