**4 MÃ HÓA ỨNG DỤNG**

**HÀM BĂM MẬT MÃ VÀ ỨNG DỤNG**

**Hàm băm mật mã**

Như ta đã biết, hệ mã có thể là hệ bảo mật thông tin trao đổi, mã đối xứng và mã bất đối xứng là các hệ mã dạng này; hay là hệ chỉ nhằm che giấu và bảo vệ tính toàn vẹn dữ liệu, hàm băm mật mã là dạng hệ mã này.

Một hàm băm mặt mã (*cryptographic hash function*), mà từ nay ta sẽ gọi tắt là hàm băm (*hash function*) là hệ mã được thiết kế để nhận vào một văn bản có chiều dài bit bất kỳ và trả về một giá trị, gọi là trị băm (hash value), là một chuỗi bit có chiều dài cố định sao cho (**i**) từ giá trị băm không thể suy ngược lại được văn bản đầu vào, hơn nữa (**ii**) nó còn có tính kháng đụng độ cao, nghĩa là hai đầu vào khác nhau, dù chỉ rất ít bit, thì hai trị băm tương ứng của chúng cũng khác nhau. Một cách hình thức, hàm băm được định nghĩa như sau:

H: {0, 1}\* → {0, 1}n, có các tính chất

(**1**) Kháng tiền ảnh (*pre-image resistance*). Cho x, tính y = H(x) là dễ, nhưng cho y = H(x), tính x là không thể.

(**2**) Kháng tiền ảnh phụ (*second* *pre-image resistance*). Cho z = H(x), rất khó để tìm ra một y khác x mà có cùng trị băm z = H(y).

(**3**) Kháng đụng độ (*collision resistance*). Rất khó tìm được 2 đầu vào x, y khác nhau mà có cùng trị băm.

**Về các tính chất hàm băm**

Tính kháng tiền ảnh nhằm chống lại những tấn công, ở đó, tin tặc chỉ có trị băm và muốn phục hồi lại bản gốc. Với tính chất này, hàm băm còn được gọi là hàm 1-chiều (1*-way function*) và thường được sử dụng để bảo vệ mật khẩu trong các hệ thống có yêu cầu chứng thực người dùng.

Tính kháng đụng độ đảm bảo không thể tìm được 2 văn bản khác nhau có cùng trị băm. Tính chất này khiến việt thiết kế hàm băm mật mã. Thực vậy, theo định nghĩa, bản chất của hàm băm là một hàm nén thông tin (không thể phục hồi, theo tính kháng tiền ảnh), vì thế, nguyên lý chuồng câu (*nguyên lý Dirichlet*) khẳng định rằng sẽ có các văn bản khác nhau có cùng trị băm, nghĩa là **đụng độ là tồn tại**. Hàm băm có tính chất kháng đụng độ mạnh có thể được dùng để tạo định danh (*identification*) phân biệt các văn bản với nhau. Tính kháng đụng độ làm cho kẻ tấn công hệ thống chứng thực khó tìm ra được 2 mật khẩu có cùng trị băm, chẳng hạn.

Tính kháng tiền ảnh phụ, gọi là ‘*phụ* (*second*)’ vì nó không quan trong bằng tính 1-chiều của hàm băm. Hơn nữa, một hàm băm mạnh, có tính kháng đụng độ mạng thì cũng có tính kháng tiền ảnh phụ. Như vậy, có thể xem tính kháng tiền ảnh phụ để giảm bớt khó khan khi thiết kế hàm băm cho những ứng dụng kiểu như chống lại những tấn công gian lận, ở đó, kẻ tấn công muốn thay thê văn bản gốc bằng một văn bản khác có cùng trị băm.

**Thiết kế hàm băm**

Nguyên lý cơ bản là thiết kế một hàm băm mà trị băm là các giá trị ngẫu nhiên chỉ phụ thuộc vào văn bản đầu vào.

Từ nguyên lý trên, thiết kế một hàm băm sử dụng một hàm trộn (*mixed algorithm*) M mà biến đổi 1 chuỗi n-bit thành 1 chuỗi n-bit, và sử dụng M như ‘*tim*’ hàm hăm H để trộn 2 chuỗi cùng chiều dài bit. Kích thước chuỗi tùy thuộc ứng dụng, thường là 128 hay 256 bit, và đó cùng là chiều dài bit đầu ra của hàm băm. Để băm văn bản D, thuật toán băm H thi hành các bước sau:

**Algorithm H(D)**

(**Bước 1**) Chia văn bản D thành các khối cùng chiều dài n bit. Chuỗi cuối cùng có thể dán (*padding*) thêm một số bit, theo quy ước trước, đề được mọi chuỗi n bit.

. D = B1 || B2 || … || Bk.

Trong đó || ký hiệu phép ghép chuỗi.

(**Bước 2**) Thi hành hàm trộn M liên tục theo cách chuỗi sau sẽ trộn với kết quả biến đổi của chuỗi trước qua M.

. H0 là chuỗi khởi tạo, còn gọi là mầm (*seed*), tính

. Hi = Hi-1 ⊕ M(Hi), i = 1, 2,…, k

(phép trộn ⊕ thường được dùng là phép XoR bit)

(**Bước 3**) Kết quả trả về là chuỗi cuối cùng.

. return Hk.

**Hàm băm có khóa**

Hàm băm có khóa (*keyed hash function*) là cách sử dụng biến thể của hàm băm tổng quát H(D) bằng cách thêm một số thông tin mật k, gọi là khóa (*key*), không thuộc văn bản D, để chỉ người có khóa mới tạo lại được trị băm đúng với trị băm đã tạo với cùng khóa trên D. Chẳng hạn, có thể dùng khóa (padding nếu cần) làm mầm H0. Hay đơn giản là them vào cuối văn bản D.

Hàm băm có khóa có thể được dùng tạo mã chứng thực thông điệp – MAC (Message Authentication Code).

**Cài đặt hàm băm**

***Cài đặt dựa trên hàm mã hóa***

Bản chất của các hệ mã, đối xứng cũng như như bất đối xứng, là các hàm ngẫu nhiên, nên có thể sử dụng một hệ mã E có cùng chiều dài đầu ra của hàm băm như hàm trộn M. Ví dụ, có thể sử dụng hệ mã ECC làm hàm trộn M để tạo hàm băm 256 bit.

Vì tốc độ là yếu tố quan trọng đối với hàm băm nên người ta có xu hướng sử dụng hàm băm được xây dựng bằng cách sử dụng hàm trộn M với các cách trộn khác nhau, thay vì dựa vào các bài toán khó.

***Các hàm băm mật mã thông dụng***

Các hàm băm được sử dụng rộng rãi nhất hiện nay:

*MD5* – **Message Digest 5**

Được giới thiệu bởi Rivest, 1 trong 3 tác giả của RSA, năm 1992. MD5 được sử dụng phổ biến để bảo mật mật khẩu.

*SHA* – **Secure Hash Algorithm**

Là thuật toán băm chuẩn được sử dụng trong nhiều ứng dụng, đặc biệt là trong chuỗi khối. SHA có 3 phiên bản với kích thước trị băm khác nhau:

. SHA-1. Hàm băm tạo ra trị băm theo nguyên tắc tương tự nguyên tắc Ronald L. Rivest sử dụng khi thiết kế thuật toán băm MD2, MD4 và MD5, nhưng tạo ra trị băm chó chiều dai bit lớn hơn (160 bit so với 128 bit).

. SHA-2. Gồm nhiều hàm băm do Cơ quan An ninh Quốc gia Hoa Kỳ (USA National Secure Agency) thiết kế và công bố năm 2001, bằng cách sử dụng cấu trúc Merkle–Damgård, từ một hàm nén một chiều theo cấu trúc Davies–Meyer của một mật mã khối chuyên dụng. Sau đó, được Viện Tiêu Chuẩn Hoa Kỳ (National Institute of Standards and Technology) chọn là hàm băm chuẩn với trị băm dài 256 bit.

SHA-3. Là thành viên mới nhất của nhóm hàm băm chuẩn an toàn, được NIST công bố ngày 5 tháng 8 năm 2015. Mặc dù tên là SHA, nhưng SHA-3 có sự khác biệt về cấu trúc so với SHA-1 và SHA-2 là các SHA theo cấu MD5.

SHA-3 là một hê mã mở rộng của Keccak, được thiết kế bởi Guido Bertoni, Joan Daemen, Michaël Peeters và Gilles Van Assche, dựa trên hàm RadioGatún. Các tác giả của Keccak đã đề xuất các cách sử dụng bổ sung cho hàm này, bao gồm mã dòng (*stream cipher*), mã xác thực (*authentication*), và lược đồ băm "*cây*" để băm nhanh hơn trên một số kiến trúc.

***Cài đặt sử dụng lại các hàm băm đã biết***

Với kết quả sau, ta có thể thiết kế một hàm băm mới dựa trên sự kết hợp của các hàm băm đã có.

**Mệnh đề (hợp các hàm băm)**. Nếu

, và

,

là 2 hàm băm mật mã, thì

, và

,

cũng là các hàm băm mật mã.

**SỐ NGẪU NHIÊN**

Sinh được một số ngẫu nhiên là bài toán quan trọng trong thực tế, đặc biệt trong mật mã, như ta đã thấy trong các hệ mã đã học, ở đó, chẳng hạn như hệ mã ElGamal, nếu hàm sinh số ngẫu nhiên có tính ngẫu nhiên yếu, hệ mã có thể bị phá vỡ.

Lý tưởng nhất là ta có một thiết bị tạo ra danh sách hoàn toàn ngẫu nhiên gồm các số 0 và 1. Thiết bị như thế tồn tại trong thực tế, chẳng hạn bộ đếm Geiger (*Geiger counter*). Tuy nhiên, trên thực tế, việc xây dựng bộ đếm Geiger cho máy tính lại rất tốn kém. Nên thực tế vẫn là là chỉ có thể tạo các số giả ngẫu nhiên (*pseudo-random number*).

**Trình sinh số giả ngẫu nhiên**

Trình sinh số giả ngẫu nhiên – **PRGN** (*Pseudo-Random Number Generator*) là một hàm hai biến

,

F(c, s) nhận vào một số nguyên không âm và một số s (mầm – *seed*), trả về bit thứ c. Quá trình có thể được mô tả như sau:

. Để bắt đầu, chọn giá trị mầm S (thực hoặc ngẫu nhiên nhất có thể).

. Thực tyhi liên tiếp hàm F để tính R0 = F(0, S), R1 = F(1, S), …

. Kết quả trả về là chuỗi bit giả ngẫu nhiên R0||R1||…

**Trình sinh số giả ngẫu nhiên an toàn mật mã**

PRNG an toàn mật mã (*Cryptographically secure PRNG*) nếu:

* Kẻ thám mã Ever biết k bit đầu tiên của chuỗi bit ngẫu nhiên, Ever cũng sẽ không lợi thê trên hơn 50% trong việc dự đoán liệu bit tiếp theo sẽ là 0 hay 1.
* Giả sử rằng Ever có thể tìm ra các giá trị Rt, Rt+1, … thì điều đó cũng sẽ không giúp Ever xác định được giá trị của t bit R0, …, Rt-1 trước đó.

**Cài đặt trình sinh số giả ngẫu nhiên**

Trình sinh số giả ngẫu nhiên **PRGN** có thể được thiết kế với một hàm băm mật mã (*hash function*) H bằng cách chọn giá trị ngẫu nhiên ban đầu S và thực thi: Ri=H (i||S).

Cũng có thể thiết kế **PRGN** từ một hệ mã, đối xứng hay bất đối xứng EK như RSA hay AES chẳng hạn: R = EK(C XoR S), trong đó C = EK(D) với D là thời gian máy tính.

**CHỮ KÝ SỐ**

**Nguyên tắc tạo chữ ký**

Với các hệ mã công khai có tính đối xứng như RSA, bằng cách sử dụng khóa cá nhân d trong quá trình mã hóa (thay vì khóa công khai e), Allice sẽ thực hiện ‘mã hóa’ có thể tạo ra ‘bản mã’ S cho bản rõ M nhập vào. Chỉ người có khóa cá nhân d mới có thể tạo ra được S tương ứng với M, hơn nữa, mọi người đều có thể đọc được S bằng cách sử dụng khóa công khai (mà ai cũng có thể biết) e và n để ‘giải mã’. Trong trường hợp thực hiện ngược như vậy,

. S được gọi là chữ ký (*signature*) của Allice trên thông điệp M, và quá trình mã hóa bây giờ được gọi là quá trình ký (*sign*).

**Sign**: S = RSA(M, d, n) = Md mod n.

. Bob dùng khóa công khai thực hiện giải mã S, nếu “*đọc hiểu*” kết quả thì Bob tin chắc rằng Allice đã ký trên M. Trong thực tiễn, Bob biết M, và so sánh với kết quả giải mã. Tiến trình đó được gọi là xác minh (*verification*).

**Verification**: If RSA(S, e, n) ≡ M: return TRUE else: return FALSE.

Trên nguyên tắc, mọi hệ mã công khai đều có thể được dùng để tạo ra chữ ký bằng cách sử dụng khóa cá nhân trong tiến trình ký và dùng khóa công khai để xác minh. Tuy nhiên, những hệ mã không có tính đối xứng như RSA, chẳng hạn ElGamal, thì quá trình ký và xác minh sẽ phức tạp hơn.

**Thuật giải ký số**

Thuật giải ký số – **DSA/DSS** (*Digital Signature Algorithm/Standard*) được dùng cho mục đích bảo toàn tính toàn vẹn của dữ liệu và phải xác minh được. Tiến trình chung gồm 2 bước chính

. **Ký** – signature. Sử dụng hàm băm để bảo vệ tính toán vẹn và khóa cá nhân để ký nhận trên trị băm.

. **Xác minh** – verification. Sử dụng khóa công khai để xác minh chữ ký có đứng được ký bằng khóa cá nhân tương ứng hay không.

Một hàm băm 1-chiều được sử dụng để tạo tạo ra một phiên bản cô đọng của dữ liệu, gọi là tiêu bản của văn bản (*digist*). Do các tính chất của hàm băm mật mã, tiêu bản là trị băm đảm bảo tính toàn ven của dữ liệu. Chữ ký số được gửi cho người nhận kèm theo dữ liệu được ký. Người nhận thông điệp có thể xác minh chữ ký bằng cách sử dụng khóa công khai của người gửi. Tiến trình xác minh cũng sử dụng cùng hàm băm khi ký.

**Giao thức ký - DSA**

***Các tham số của DSA***

. p là số nguyên tố 512 bit.

. q là số nguyên tố 160 bit.

. n = pq.

. g là phần tử thuộc và có bậc q: .

. H là hàm băm mật mã, băm dữ liệu thành giá trị nhị phân 160 bít: .

. Khoa ký bí mật d, 0 < d < q.

. Khóa kiểm công khai e: e ≡ gd (mod p).

***Tạo chữ ký DSA – signature***

Để ký trên dữ liệu M, người ký thực hiện:

(1) Chọn số ngẫu nhiên sử dụng 1 lần (nonce) .

(2) Tính r ≡ (gk mod p) (mod q) và s ≡ k-1(H(M) + dr) (mod q).

(r, s) chính là chữ ký của dữ liệu M sử dụng khóa ký d.

***Xác minh chữ ký DSA – verification***

Để xác minh xem (r, s) có phải là chữ ký trên M, sử dụng khóa công khai e, người xác minh thực hiện:

(3) Tính t ≡ s-1 (mod q).

(4) Kiểm tra xem r == (gH(M)tert (mod p) (mod q) hay không?

. Nếu đúng, xác nhận (r, s) là chữ ký trên M.

. Nếu không, từ chối xác nhận.

***Tính đúng đắn của DSA***

Ta có

Suy ra,

.

**BẢO MẬT CƠ SỞ DỮ LIỆU**

***Phương pháp số dư Trung Hoa***

CSDL là một thách thức với người thiết kế hệ thống mảo mật thông tin. CSDL bản than mang nghĩa tạo ra để chia sẻ. Tính chia sẻ thường phức tạp và thương được thực hiện theo những chính sách, quy định của tổ chức khi truy cập các thành phần dữ liệu trong CSDL.

Không mất tính tổng quát, ta giả sử CSDL là tập các dữ liệu thành phần:

D = <F1, F2, …, Fn>,

trong đó, D là CSDL và Fi là các dữ liệu thành phần (tập tin, bản ghi, …). Mỗi thành phần của D có thể được xem như một số nguyên .

Để mã hóa D, trước hết, chọn n số nguyên tố phân biệt mi, i = 1, 2, …, n sao cho mi > Fi, ∀i = 1, 2, …, n. Sử dụng định lý số dư Trung Hoa, CRT, giải hệ

.

C chính là bản mã của CSDL D, được tính như sau:

Đặt

M ≡ m1m2…mn

Mi ≡ M/mi, ∀i = 1, 2, …, n

,

và tính

.

Các số nguyên e1, e2, …, em được sử dụng như những khóa ghi.

Để phục hồi dữ liệu thành phần Fi, ∀i = 1, 2, …, n, từ bản mã C của D, thi hành phép tính:

.

Các số nguyên m1, m2, …, mn, được gọi là các khóa đọc. Chỉ người biết khóa đọc mi, 1 ≤ i ≤ n, mới có thể đọc được thành phần dữ liệu Fi, nhưng không thể đọc được các thành phần Fj, j ≠ i, khác.

***Mã hóa CSDL – encryption***

Người quản trị CSDL thi hành các phép toán sau trên CSDL D = {F1, F2, …, Fn}.

(1) Chọn n số nguyên tố khác nhau m1, m2, …, mn thỏa mi > Fi, ∀i = 1, 2, …, n.

(2) Sử dụng định lý số dư Trung Hoa, thiết lập và giải hệ đồng dư bằng cách tính , như mô tả trên.

(3) Phân phối khóa đọc cho các người dùng tương ứng.

***Giải mã CSDL***

Để đọc thành phần dữ liệu F người dùng phải có khóa đọc m ∈ {m1, m2, …, m­n}, và thực hiện phép tính

.

***Ví dụ***

D = {F1, F2, F3} = {2, 2, 5}

. Chọn m1 = 3 > 2, m2 = 5 ≠ 3 > 2, m3 = 7 > 5.

. M = 3\*5\*7 = 105

. M1 = 5\*7 = 35 → (M1)-1 ≡ 35-1 (mod 3) ≡ 2 (mod 3) → e1 = 35\*2 = 70

M2 = 3\*7 = 21 → (M2)-1 ≡ 21-1 (mod 5) ≡ 1 (mod 5) → e2 = 21\*1 = 21

M3 = 3\*5 = 15 → (M3)-1 ≡ 15-1 (mod 7) ≡ 1(mod 7) → e3 = 15\*1 = 15

. C ≡ 70\*2 + 21\*2 + 15\*5 ≡ 47 (mod 105)

Ta có

. 47 % 3 = 2 = F1

47 % 5 = 2 = F2

47 % 7 = 5 = F3.

**HỆ THỐNG NHIỀU THÀNH VIÊN**

**Bài toán n thành viên**

Có n nhà khoa học làm việc trong dự án bí mật. Họ muốn mã hóa tài liệu sao cho chỉ có thể đọc được nếu có ít nhất k thành viên có mặt.

**Mô hình ngưỡng (k, n)**

Bài toán n thành viên có thể được đưa về giải bằng mô hình ngưỡng – **(k, n)-threshold**, phát biểu như sau:

(k, n)-threshold là phương pháp chia se bí mật S giữa n thành viên P1, …, Pn, thỏa các tính chất sau:

. k < n. k được gọi là ngưỡng.

. Mỗi thành viên Pi giữ một bí mật thành phân Ii, i = 1, 2, …, n.

. Cần tối thiểu k thành viên cung cấp tối thiểu k bí mật trong tập {I1, I2, …, In} để đọc được S.

. Ít hơn k thành viên cung cấp bí mật thì không thể đọc được S.

**Phương pháp nội suy Lagrange**

Có nhiều cách để cài đặt mô hình ngưỡng (k, n)-threshold, ow đây chúng ta sử dụng kỹ thuật nội suy Lagrange.

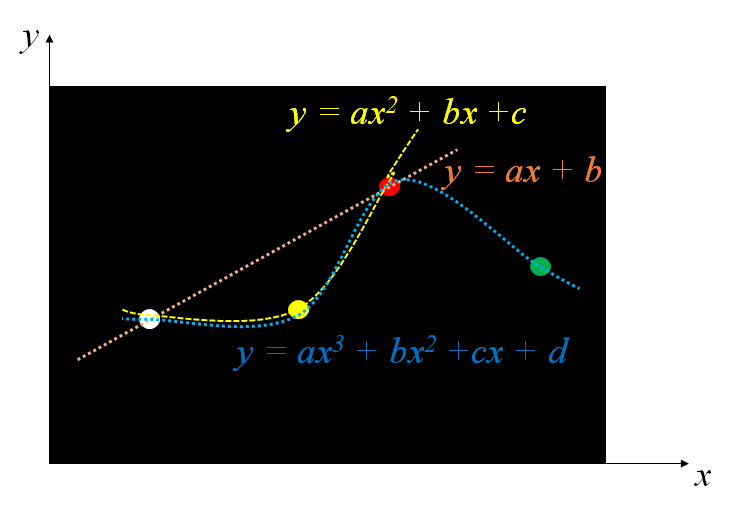
***Đa thức***

Ta biết rằng, trong mặt phẳng Oxy,

. Qua 2 điểm (x1, y1), (x2, y2), xác định duy nhất đường thẳng y = ax + b.

. Qua 3 điểm (x1, y1), (x2, y2), (x3, y3), xác định duy nhất một parabol y = ax2 + bx + c.

. Qua 4 điểm (x1, y1), (x2, y2), (x3, y3), (x4, y4) xác định duy nhất một đa thức bậc 3 y = ax3 + bx2 + cx + d



Một cách tổng quát, qua n + 1 điểm {(x0, y0), (x1, y1),…, (xn, yn)}xác định duy nhất một đa thức bậc n:

***Công thức nội suy Lagrange***

Xét đa thức bậc n có dạng

.

.

.

.

.

…

.

.

Vậy,

,

hay

, (\*)

với

, (\*\*)

và

. (\*\*\*)

**Giao thức (k, n)-threshold**

***Thiết lập bí mật riêng phần***

Để tạo bí mật riêng phần cho bí mật chung S, quản trị hệ thống thực hiện:

(1) Trước hết, chọn 1 số nguyên tố lớn p.

(2) Chọn ngẫu nhiên k – 1 số .

(3) Tính và cấp cho thành viên Pi cặp , với các phân biệt và (mod p).

***Phục hồi bí mật chung***

Để phục hồi bí mật chung, cần ít nhất k thành viên cung cấp các bí mật riêng , và thực hiện

(1) Tính K hệ số Ai theo công thức (\*)

(2) Tính K đa thức Li(xi, yi) theo công thức (\*\*)

(3) Xác định đa thức theo công thức (\*) và tính .

**Ví dụ**

Giả sử cần chia sẻ bí mật S = 4 cho 5 người sao cho có ít nhất 3 người hợp tác mới có thể phục hồi được S.

. Xét (p = 11).

. Do k = 3, ta sẽ tạo ngẫu nhiên parabol y = f(x) = S + a1x + a2x2 ≡ 5 + 2x + x2.

. Sinh ra 5 bí mật thành phần dạng (x, f(x)): {(1, 8), (2, 2), (4, 7), (7, 2), (10, 4)}

***Ví dụ 1***

Nhận được {(1,8), (4, 7), (7,2)}.

.

.

.

***Ví dụ 2***

Nhận được {1, 8), (10, 4)}.

.

. .

***Ví dụ 3***

Nhận được {(1,8), (4, 7), (7,2), (10, 4)}.

.

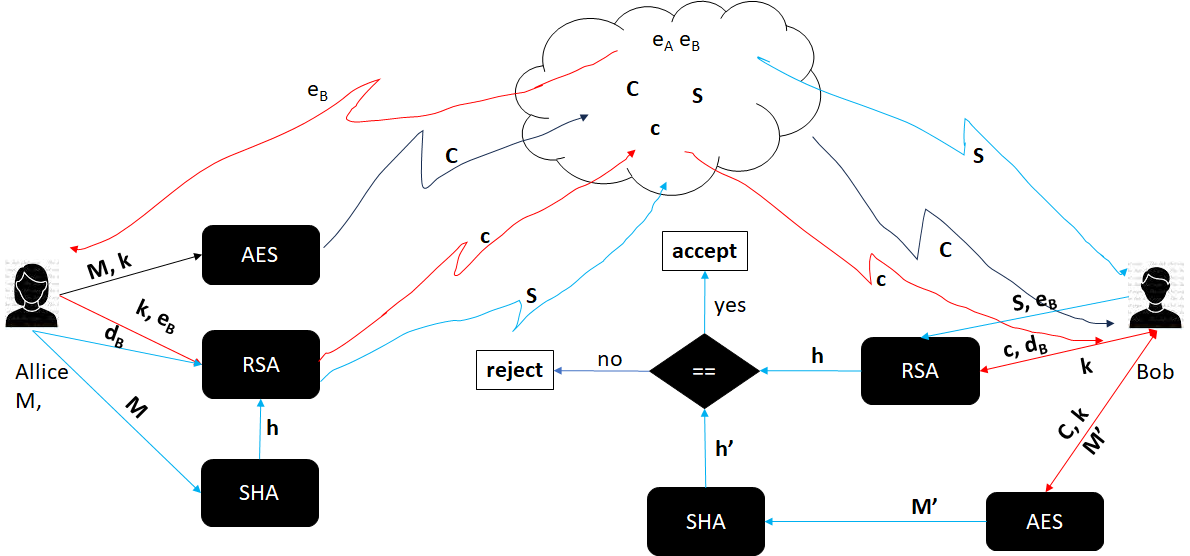
. .

**BẢO MẬT THÔNG TIN**

**Yêu cầu**

Xây dựng hệ thống cung cấp dịch vụ bảo mật thông tin khi trao đổi trên mạng. Các dịch vụ hệ thống cung cấp:

* Bảo mật thông tin dạng tập tin gửi đính kèm qua email.
* Có chức năng ký trên văn bản.
* **Quy trình**
* ***Mã hóa***
* (1) Allice mã hóa văn bản M bằng hệ mã bí mật, ký hiệu **AES**, với khóa bí mật k, và
* (2) mã hóa khóa bí mật k bằng hệ mã công khai, ký hiệu **RSA**, với khóa công khai của eB Bob.
* (3) Chuyển kết quả mã hóa C = AES(k, M) và c = RSA(eB, k) cho Bob.



***Giải mã***

(4) Bob lấy C và c xuống.

(5) Giải mã c dùng khóa cá nhân của Bob để được khóa bí mật k = RSA(dB, c).

(6) Dùng khóa k giải mã C để được văn bản M = AES(k, C).

***Ký***

Trong trường hợp giao dịch có ký, Allice thực hiện

(7) Băm văn bản M bằng hàm băm mật mã, ký hiệu **SHA**.

(8) Sử dụng khóa cá nhân của mình ký trên trị băm nhận được.

(9) Gửi cho Bob chữ ký S = RSA(dA, SHA(M)).

***Xác minh chữ ký***

Để xác minh, Bob thực hiện

(10) Lấy chữ ký S xuống.

(11) Băm văn bản M’ đã giải mã, h’ = SHA(M’).

(12) Sử dụng khóa công khai của Allice để kiểm tra xem RSA(eB, S) == h’.