**MỤC LỤC**

[**Câu 1: Trình bày tính chất của hàm băm. Giải thích ứng dụng trong kiểm tra tính toàn vẹn/ký số.** 2](#_Toc28192957)

[**Câu 2: Phân tích ưu, nhược điểm của mật mã đối xứng và mật mã bất đối xứng/mật mã khóa công khai. Từ đó đề xuất mô hình sử dụng kết hợp mật mã đối xứng và mật mã bất đối xứng để trao đổi thông tin mật qua môi trường không an toàn.** 3](#_Toc28192958)

[**Câu 3: Vẽ và giải thích sơ đồ thực hiện ký và kiểm tra chữ ký sử dụng MM khóa công khai** 4](#_Toc28192959)

[**Câu 4: Định nghĩa mã dòng. Vẽ sơ đồ và giải thích hoạt động của một hệ mã dòng tổng quát** 5](#_Toc28192960)

[**Câu 5: Cho sơ đồ sinh dòng khóa của hệ mật RC4** 6](#_Toc28192961)

[**Câu 6: Các chế độ hoạt động của mã khối (ECB, CBC, OFB, CFB, CTR).** 8](#_Toc28192962)

[**Câu 7: Hệ mật AES** 14](#_Toc28192963)

[**Câu 8: Mật mã trên đường cong elliptic: nhóm hữu hạn trên đường cong elliptic, mật mã trên đường cong elliptic (sinh khóa, mã hóa, giải mã, nhúng số)** 16](#_Toc28192964)

[**Câu 9: Chuẩn mật mã RSA (theo PKCS #1)** 18](#_Toc28192965)

[**Câu 10: Các khái niệm: Tính sống của thực thể; Tính tươi của thông điệp; Xác thực thực thể; Xác thực thông điệp; Phân biệt xác thực bằng mật khẩu và xác thực bằng thách đố - giải đố.** 23](#_Toc28192966)

[**Câu 11: Xác thực bằng mật khẩu:** 24](#_Toc28192967)

[**Câu 12: Các giao thức khác: trình bày hoạt động của giao thức. Giải thích quyết định của bên xác thực.** 29](#_Toc28192968)

[**Câu 13: Cộng điểm trên đường cong Elliptic** 30](#_Toc28192969)

[**Câu 14: Phương pháp nhúng số lên điểm thuộc đường cong trên Fp** 32](#_Toc28192970)

[**Câu 15: Hệ mật ElGamal** 34](#_Toc28192971)

# **Câu 1: Trình bày tính chất của hàm băm. Giải thích ứng dụng trong kiểm tra tính toàn vẹn/ký số.**

1. **Tính chất của hàm băm:**

Hàm băm là một ánh xạ từ không gian có độ dài tùy ý sang không gian có độ dài cố định. Hàm băm có những tính chất sau:

* Tính nén: hàm băm h ánh xạ một đầu vào x có độ dài bit tùy ý thành một đầu ra h(x) có độ dài bit cố định n.
* Tính dễ tính toán: cho trước h và một đầu vào x, việc tính toán h(x) là dễ dàng.
* Tính kháng tiền ảnh: với hầu hết toàn bộ các đầu ra xác định trước, ta không thể tính toán để tìm ra đầu vào ban đầu, nghĩa là từ h(x) đã biết không tìm ra được đầu vào x ban đầu.
* Tính kháng tiền ảnh thứ hai: cho trước đầu vào x thì không thể tìm ra được nghịch ảnh thứ hai x’ ≠ x mà h(x) = h(x’).
* Tính kháng va chạm: không thể tìm được cặp đầu vào x và y phân biệt nào mà băm thành cùng một đầu ra được, nghĩa là không có cặp x ≠ y nào mà có đầu ra h(x) = h(y).

1. **Ứng dụng trong kiểm tra tính toàn vẹn/ký số:**

Bản rõ m được đưa qua hàm băm h thu được h(m), h(m) được mã hóa bởi khóa bí mật của người gửi tạo thành chữ ký số của bên gửi, sau đó chữ ký số này được gắn kèm với bản rõ m và được gửi cho bên nhận.

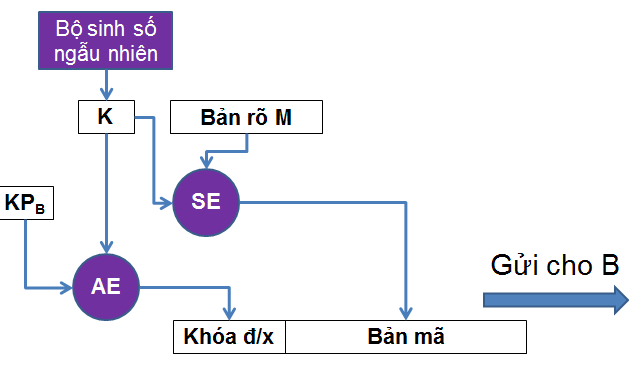
Sau khi nhận được bản tin, người nhận sẽ thực hiện tách bản rõ m và chữ ký số. Bản rõ m được băm qua hàm h để thu được h(m). Chữ ký số đưa qua hàm giải mã sử dụng khóa công khai của bên gửi, thu được h’. Tiến hành so sánh h(m) và h’. Nếu h’ trùng với h(m) thì thông điệp nhận được không bị sửa đổi trên đường truyền. Ngược lại, thông điệp không phải được gửi từ bên gửi hoặc đã bị sửa đổi trên đường truyền, nghĩa là đảm bảo được tính toàn vẹn. Có thể tin tưởng như vậy vì hàm băm có tính kháng va chạm và tính kháng tiền ảnh thứ hai, nghĩa là sẽ không có bản rõ m’ nào mà m’ ≠ m lại có thể qua hàm băm h cho được giá trị h(m’) = h(m). Do đó có thể kiểm tra được tính toàn vẹn của dữ liệu.

# **Câu 2: Phân tích ưu, nhược điểm của mật mã đối xứng và mật mã bất đối xứng/mật mã khóa công khai. Từ đó đề xuất mô hình sử dụng kết hợp mật mã đối xứng và mật mã bất đối xứng để trao đổi thông tin mật qua môi trường không an toàn.**

1. **Ưu, nhược điểm của mật mã đối xứng và mật mã bất đối xứng/mật mã khóa công khai:**

|  |  |  |
| --- | --- | --- |
|  | **Mật mã đối xứng** | **Mật mã bất đối xứng/Mật mã khóa công khai** |
| **Ưu điểm** | - Phương pháp tính toán đơn giản, tốn ít tài nguyên  - Tốc độ cao  - Đảm bảo toàn vẹn dữ liệu  - Đảm bảo bí mật dữ liệu | - Đảm bảo được bí mật, toàn vẹn dữ liệu  - Chống chối bỏ  - Xác thực được người dùng.  - Dễ quản lý khóa |
| **Nhược điểm** | - Vấn đề trao đổi khóa phải được thực hiện trên kênh an toàn  - Khó quy trách nhiệm khi bị lộ khóa cho một trong hai bên giữ khóa | - Cần người quản trị tin cậy để sinh khóa công khai và bí mật cho từng người  - Phép tính phức tạp hơn  - Tốc độ chậm |

1. **Mô hình kết hợp MM đối xứng và MM bất đối xứng:**

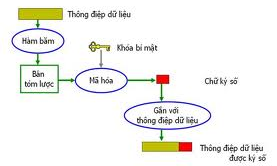


* Tại bên gửi: Bản rõ M được mã hóa bằng khóa K (khóa K được sinh ra từ bộ sinh khóa ngẫu nhiên) tạo thành bản mã SE, sau đó khóa công khai (KPb) được sử dụng để mã hóa khóa K (EK). EK được gắn với bản mã SE và gửi cho B.
* Sau khi B nhận được thông điệp trên sẽ sử dụng khóa bí mật Ks để giải mã EK thu được khóa K. Tiếp theo dùng khóa K để giải mã bản mã SE thu được bản rõ.

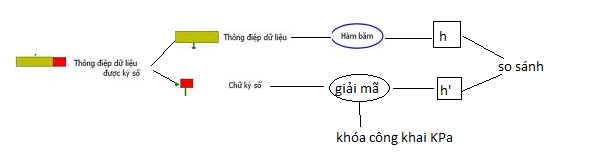
# **Câu 3: Vẽ và giải thích sơ đồ thực hiện ký và kiểm tra chữ ký sử dụng MM khóa công khai**

1. **Sơ đồ:**

Bên gửi:



Bên nhận:



1. **Giải thích sơ đồ:**

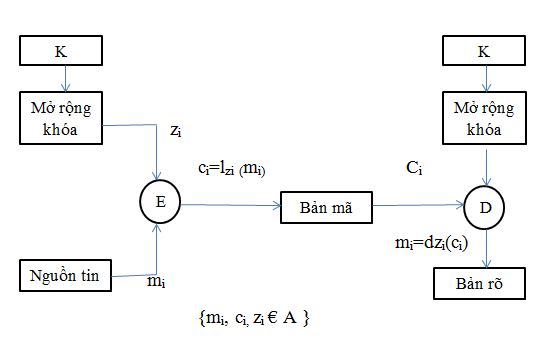
* Bên gửi: bản rõ (m) được đưa qua hàm băm H thu được h. h được mã hóa bởi khóa bí mật của bên gửi KSA tạo thành chữ ký số của A. sau đó được gắn kèm với bản rõ (m) và được gửi cho bên nhận.
* Bên nhận: sau khi nhận được bản tin sẽ thực hiện tách bản rõ (m) và chữ ký số.Với bản rõ (m): m được băm qua hàm H để thu được h. Với chữ ký số: đưa qua hàm giải mã sử dụng khóa công khai của A thu được h’.
* Tiến hành so sánh: h và h’. Nếu h’ trùng với h: thông điệp nhận được đúng là được gửi từ bên gửi và không bị sửa đổi trên đường truyền. Nếu ngược lại, thông điệp không phải được gửi từ bên gửi hoặc đã bị sửa đổi trên đường truyền.

# **Câu 4: Định nghĩa mã dòng. Vẽ sơ đồ và giải thích hoạt động của một hệ mã dòng tổng quát**

1. **Định nghĩa:**

Một hệ mã dòng là một hệ mật đối xứng, trong đó các kí tự rõ được kết hợp với một dòng kí tự khóa giả ngẫu nhiên. Trong mã dòng, từng kí tự rõ được mã hóa riêng rẽ bởi một kí tự tương ứng trong dòng khóa để cho ra một kí tự mã. Dòng khóa được sinh ra từ khóa đầu vào bởi bộ mở rộng khóa, bộ mở rộng khóa này quyết định độ an toàn và là đặc trưng để phân biệt các hệ mã dòng khác nhau.

1. **Sơ đồ tổng quát:**



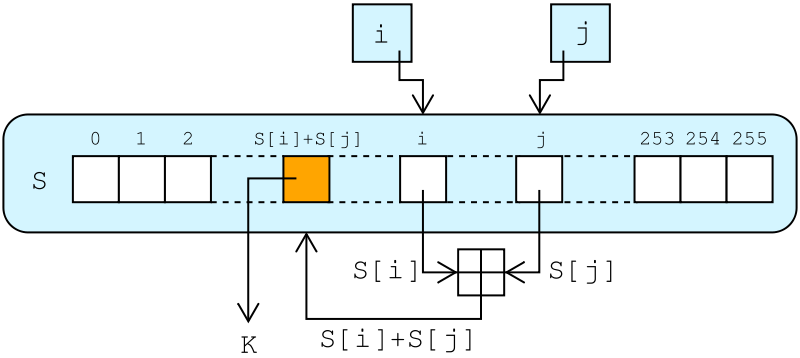
1. **Giải thích sơ đồ:**

Hai bên A và B muốn truyền tin cho nhau.

* Trước hết A và B thỏa thuận trước 1 khóa ngẫu nhiên K và chung 1 hàm mở rộng khóa. Đầu tiên, bên người gửi A cho khóa K đi qua hàm mở rộng khóa thu được 1 dòng khóa Zi. Zi có kích thước bằng kích thước gói tin cần mã hóa.
* Đem XOR từng bit của Zi (Z1,Z2,…Zi ) với lần lượt từng bit của tin Mi (m1,m2,…mi ). Ta sẽ thu được bản mã Ci có kích thước bằng kích thước dòng khóa Zi và thông điệp Mi.
* Bản mã Ci được đem gửi cho bên người nhận B
* Tương tư, bên người nhận B. dùng khóa K đã thỏa thuận cho đi qua bộ mở rộng khóa cũng thu được Zi.
* Thực hiện XOR lần lượt từng bit của dòng khóa Zi với từng bit của bản mã nhận được Ci ta sẽ giải mã ra được thông điệp Mi

Mã dòng dùng chung một khóa K, đi qua bộ mở rộng khóa (bộ sinh số giả ngẫu nhiên), tạo ra dòng khóa z để phục vị mã hóa từng kí tự trong bản rõ m. Mỗi Mi đc mã hóa bằng Zi tương ứng trong dòng khóa. Bản mã c gồm các kí tự Ci, đi qua bộ giải mã, kết hợp với Zi trong dòng khóa Z để giải mã thu được từng ký tự bản rõ Mi của bản rõ m.

# **Câu 5: Cho sơ đồ sinh dòng khóa của hệ mật RC4**



1. **Nêu đặc trưng của hệ mật RC4:**

* RC4 là hệ mã dòng với độ dài khóa không cố định (40-2048 bit)
* Thuật toán RC4 không dựa trên LFSR (thanh ghi dịch phản hồi tuyến tính), nó dựa trên việc sử dụng những hoán vị ngẫu nhiên nên có thể vận hành rất nhanh trong phần mềm. Với mỗi bước sinh khóa, RC4 sẽ sinh ra được 1 byte khóa.
* RC4 có tính phi tuyến cao, chu kỳ không lặp lại. Độ an toàn cao, có thể chống lại tấn công thám mã tuyến tính và thám mã vi sai.
* Thuật toán có thể tổng quát với các trường hợp độ dài lớn hơn và kích thước mảng S lớn hơn.
* Thích hợp cho việc cài đặt bằng phần mềm.
* Được ứng dụng rộng rãi.

1. **Mô tả quá trình sinh dòng khóa:**

* Thuật toán RC4 sẽ bắt đầu bằng việc tạo một mảng S gồm các hoán vị ngẫu nhiên 256 phần tử từ 0 đến 255. Sau khi có mảng S rồi sẽ tiến hành việc sinh dòng khóa lần lượt từng byte một.
* Đầu tiên, khởi tạo hai biến i, j với giá trị = 0. Để tính toán các 1 byte của dòng khóa ta thực hiện các bước sau:
* Tính toán giá trị mới của i và j:

i:= 0; j:=0;

i : = (i+1) mod 256;

j : = (j+S[i]) mod 256;

* Đổi chỗ S[i] và S[j] để tăng tính ngẫu nhiên.
* Gọi byte tiếp theo của dòng khóa từ S ở vị trí thứ t với

swap (S[i] ;S[j]);

t = (S[i]+S[j]) mod 256;

k = S[t];

* Mỗi 1 lần biến chạy i tăng thêm 1 thì 1 byte khóa sẽ được sinh ra i chạy đến 255 thì ta sẽ được 256 byte khóa.

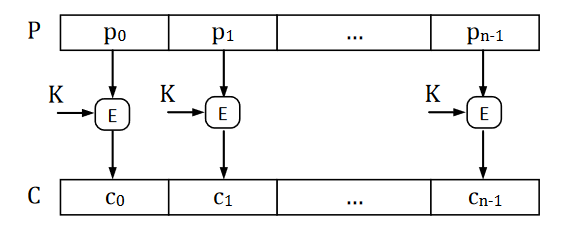
1. **Nhận xét sự thay đổi của mảng S:**

Trong quá trình sinh khóa, mảng S bị thay đổi và thay đổi chậm, mỗi lần biến i tăng lên là S thay đổi một lần , các giá trị trong mảng thay đổi lần lượt (tại mỗi bước sinh khóa chỉ có 2 ô [i], [j] thay đổi giá trị, các ô còn lại không ảnh hưởng),chỉ hoán đổi vị trí 2 ô với nhau mà không làm thay đổi tập hợp. Biến i làm cho các phần tử của S đều thay đổi và biến j làm cho sự thay đổi đó ngẫu nhiên hơn.

# **Câu 6: Các chế độ hoạt động của mã khối (ECB, CBC, OFB, CFB, CTR).**

1. **Electronic Codebook Mode (ECB):**

Sơ đồ mã hóa:



Sơ đồ giải mã:

A close up of a clock

Description automatically generated

Giải thích:

* Ở mô hình ECB, mỗi khối bản rõ được mã hóa một cách riêng rẽ, dùng chung 1 khóa K.
* Đầu vào: khóa K, các khối bản rõ n-bit p0, p1,…, p(n-1)
* Kết quả: tạo ra các khối mã c0, c1, …,c(n-1) ; giải mã để tìm ra bản rõ.
* Mã hóa: với 0≤j≤n-1, cj ← EK(pj)
* Giải mã: với 0≤j≤n-1, pj ← DK-1(cj)

Tính chất:

* Đầu vào giống nhau dẫn đến đầu ra giống nhau
* Mỗi bit lỗi trong khối mã dẫn đến mất 1 khối tương ứng khi giải mã => không có tính lan sai
* Có thể xử lý các khối song song

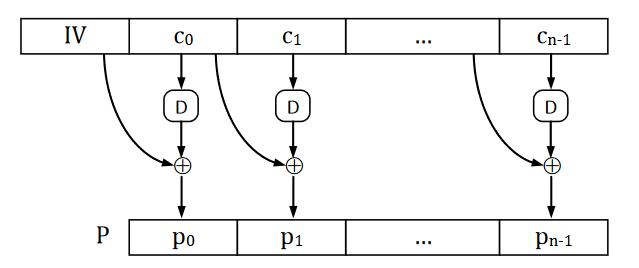
1. **Cipher Block Chaining (CBC):**

Sơ đồ mã hóa:

A picture containing sky, screenshot

Description automatically generated

Sơ đồ giải mã:



Giải thích:

* Trong mô hình CBC, bản mã của mỗi lần mã hóa được sử dụng cho lần mã hóa tiếp theo:

Ci = E (Pi Ci-1, K) với i = 1, 2, 3,…, n-1

* Do đó để mã hóa khối đầu tiên, người ta dùng một khối dữ liệu giả được gọi là vector khởi tạo IV (initialization vector) và được chọn ngẫu nhiên:

C0 = E(P0  IV, K)

* Để giải mã, làm ngược lại:

P0 = D(C0, K) IV

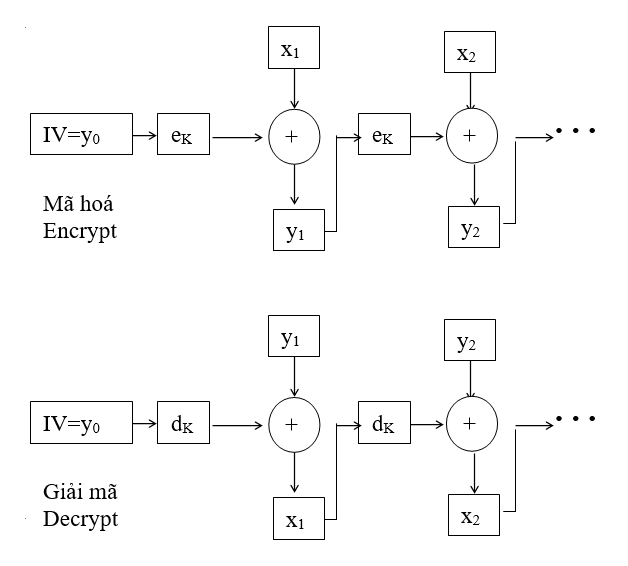
Pi = D(Ci, K) Ci-1 với i = 1, 2, 3,…, n-1

Tính chất:

* Khi các khối bản rõ giống nhau thì các khối bản mã vẫn khác nhau
* Mỗi bit lỗi trong mã thứ j dẫn đến mất khối j và j+1 khi giải mã
* Không thể mã hóa song song các khối, chỉ có thể giải mã song song

1. **Cipher Feedback Mode (CFB)**

Sơ đồ:



Giải thích:

* Trong chế độ CFB, bản mã của một khối được sử dụng để tạo khóa mã hóa khối kế tiếp, ta bắt đầu với y0 = IV (là một véc tơ khởi tạo) và tạo phần tử zi của dòng khoá bằng cách mã hoá khối sử dụng hàm mã hóa E bản mã trước đó. Tức zi = eK(yi-1), i ≥1. Bản rõ được mã hóa trực tiếp bằng phép XOR giữa zi và xichứ không phải bằng thao tác mã hóa của mã khối.
* Khi giải mã, ta bắt đầu với y0 = IV và tạo ra phần tử zi bằng cách sử dụng hàm giải mã D. Bản rõ được tạo ra bằng phép XOR giữa zi và yi chứ không phải bằng thao tác giải mã của mã khối.

Tính chất:

* Hai khối bản rõ giống nhau không cho ra hai khối bản mã giống nhau.
* Có tính lan sai, một khối bản rõ xi bị thay đổi sẽ khiến cho khối bản mã yi và tất cả các khối bản mã tiếp theo bị ảnh hưởng.
* Không thể thực hiện song song khi mã hóa nhưng có thể thực hiện song song khi giải mã

1. **Output Feedback Mode (OFB)**

Sơ đồ:

A close up of a sign

Description automatically generated

Giải thích:

* Trong Ouput Feedback mode, đầu ra của mã khối được sử dụng để tạo khóa mã hóa khối kế tiếp.
* Trong pha mã hóa:
  + Bản rõ P được chia thành các khối Pi. Để mã hóa khối thứ nhất (P1), vector IV được mã hóa bằng khóa K thu được kết quả là z1, s bit đầu tiên của z1 sẽ được sử dụng để XOR với P1 và thu được khối bản mã C1.
  + Để mã hóa khối thứ i:
    - Thực hiện dịch trái s bit với thanh ghi dịch. Ghi s bit đầu tiên của khối đầu ra vào thanh ghi dịch.
    - Mã hóa giá trị thanh ghi dịch bởi khóa K được zi. Lấy s bit đầu tiên của kết quả mã hóa zi trên để XOR với khối Pi và thu được khối Ci.

Tính chất:

* Khi bản rõ là giống nhau thì bản mã vẫn khác nhau
* Đây là mã dòng đồng bộ
* Không có tính lan sai: Một bit lỗi trong bản mã dẫn đến một bit lỗi tương ứng khi giải mã
* Bản rõ không chẵn kích thước khối vẫn không cần đệm

1. **Counter Mode (CTR)**

Sơ đồ:

A close up of text on a white background

Description automatically generated

Giải thích:

* Counter mode sử dụng một bộ đếm (counter) thay vì một bộ IV truyền thống. Bộ đếm có các thuộc tính bổ sung bao gồm: 1 nonce và 1 bộ đếm ban đầu. Ở chế độ này không yêu cầu phải đệm thêm vào bản rõ để đạt kích thước khối của bản mã.
* Do tính đối xứng của phép XOR, quá trình mã hóa và giải mã thực chất được thực hiện các bước giống nhau.
  + Ta quy ước: Ci là khối bản mã thứ i; Pi là khối bản rõ thứ i; Oi là khối mã hóa thứ i; Ii là bộ đếm thứ i bao gồm Nonce và Counter; K là khóa; E là thuật toán mã hóa.
  + Ta có công thức mã hóa: Ci = Pi Oi
  + Ta có công thức giải mã: Pi = Ci Oi
  + Trong đó Oi = Ek (Ii); Ii = Ii-1 + 1.

Tính chất:

* Khi bản rõ là giống nhau thì bản mã vẫn khác nhau
* Không có tính lan sai: Một bit lỗi trong bản mã dẫn đến một bit lỗi tương ứng khi giải mã
* Bản rõ không chẵn kích thước khối vẫn không cần đệm
* Là mã dòng đồng bộ
* Có thể cài đặt song song.

# **Câu 7: Hệ mật AES**

1. **Lịch sử ra đời:**

Từ cuối thập niên 1980, đầu thập niên 1990, xuất phát từ những lo ngại về độ an toàn và tốc độ thấp khi áp dụng bằng phần mềm, giới nghiên cứu đã đề xuất khá nhiều thuật toán mã hóa khối để thay thế DES. Những ví dụ tiêu biểu bao gồm: RC5, Blowwfish, IDEA (International Data Encryption Algorithm: Thuật toán mã hóa dữ liệu quốc tế), NewDES, SAFER và FEAL. Hầu hết những thuật toán này có thể sử dụng từ khóa 64 bit của DES mặc dù chúng thường được thiết kế hoạt động với từ khóa 64 bit hay 128 bit. Bản thân DES cũng cải tiến để có thể được sử dụng an toàn hơn.

Năm 2001, sau một cuộc thi quốc tế, NIST đã chọn ra một thuật toán mới là Tiêu chuẩn mã hóa tiên tiến AES (Advanced Encryption Standard) để thay thế cho DES. Thuật toán được trình diện dưới tên là Rijndael. Những thuật toán khác có tên trong danh sách cuối cùng của cuộc thi AES gồm: RC6, Serpent, MARS và Twofish. AES là thuật toán mã hóa khối được chính phủ Hoa Kỳ áp dụng làm tiêu chuẩn mã hóa thay cho tiêu chuẩn DES trước đó. Giống như tiêu chuẩn DES, AES được kỳ vọng áp dụng trên phạm vi toàn thế giới và đã được nghiên cứu rất kỹ lưỡng. AES được chấp thuận làm tiêu chuẩn liên bang bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) trong một quá trình tiêu chuẩn hóa kéo dài 5 năm. Thuật toán được thiết kế bởi hai nhà mật mã học người Bỉ: Joan Daemen và Vincent Rijmen. Thuật toán được đặt tên là "Rijndael" khi tham gia cuộc thi thiết kế AES theo cách ghép tên của hai đồng tác giả. Thuật toán được dựa trên bản thiết kế Square có trước đó của Daemen và Rijmen; còn Square lại được thiết kế vdựa trên Shark. Khác với DES sử dụng mạng Feistel, Rijndael sử dụng mạng thay thế-chuyển vị. AES có thể dễ dàng thực hiện với tốc độ cao bằng phần mềm hoặc phần cứng và không đòi hỏi nhiều bộ nhớ. Do là một tiêu chuẩn mã hóa mới, AES đang được triển khai sử dụng rộng rãi hàng loạt.

1. **Đặc trưng của AES:**

* Kích thước khóa là 128/192/256 bit
* Kích thước khối dữ liệu là 128 bit
* Số vòng lặp: 10, 12, 14 vòng

1. **Mô tả thuật toán**

Mặc dù 2 tên AES và Rijndael vẫn thường được gọi thay thế cho nhau nhưng trên thực tế thì 2 thuật toán không hoàn toàn giống nhau. AES chỉ làm việc với khối dữ liệu 128 bit và khóa có độ dài 128, 192 hoặc 256 bit trong khi Rijndael có thể làm việc với dữ liệu và khóa có độ dài bất kỳ là bội số của 32 bit nằm trong khoảng từ 128 tới 256 bit. Các khóa con sử dụng trong các chu trình được tạo bởi quá trình tạo khóa con Rijndael. Hầu hết các phép toán trong thuật toán AES đều thực hiện trong một trường hữu hạn. AES làm việc với từng khối dữ liệu 4×4 bytes (tiếng Anh: state, khối trong Rijndael có thể có thêm cột). Quá trình mã hóa gồm 4 bước:

* AddRoundKey: mỗi byte của khối được kết hợp với khóa con, các khóa con này được tạo ra từ quá trình tạo khóa con Rijndael.
* SubBytes: đây là phép thế (phi tuyến) trong đó mỗi byte sẽ được thế bằng một byte khác theo bảng tra (Rijndael S-box).
* ShiftRows: đổi chỗ, các hàng trong khối được dịch vòng.
* MixColumns: quá trình trộn làm việc theo các cột trong khối theo một phép biến đổi tuyến tính. Tại chu trình cuối thì bước MixColumns được thay thế bằng bước AddRoundKey.

# **Câu 8: Mật mã trên đường cong elliptic: nhóm hữu hạn trên đường cong elliptic, mật mã trên đường cong elliptic (sinh khóa, mã hóa, giải mã, nhúng số)**

1. **Nhóm hữu hạn trên đường cong elliptic:**

- Định nghĩa: Nhóm hữu hạn là một tập hữu hạn các phần từ cùng với một phép toán hai ngôi xác định trên nó. Phép toán 2 ngôi được thực hiện với các phần tử trong nhóm hữu hạn đó.

- Nhóm hữu hạn trên đường cong elliptic:

* Cho p là số nguyên tố
* Fp  là trường hữu hạn các số nguyên theo modulo p
* Đường cong E trên P được xd bởi: (1) với a,b thuộc p,



* Fp = (G,+,\*), trong đó G = (0,1,2….,p-1): tập hữu hạn các phần tử từ 0 tới p-1, các phép toán + và \* thực hiện theo modulo p.
* Một cặp (x,y) Fp là 1 điểm E nếu thỏa mãn (1)



* Luật nhóm:
  + Xét tập các điểm trên đường cong E
  + Phép (+) điểm trên tập E là phép mà sao cho (E,+) là một nhóm hữu hạn.
  + Ta có:



Luật 1: P + = P với mọi PE ( là ptu trung hòa)



Luật 2: nếu x1 x2, tính m= (y2-y1)/(x2-x1)



x3= - x2 – x1



y3= m(x1-x3)-y1

Luật 3 : Nếu x1=x2 và y1y2 thì P1+P2=P3=



Vì , x1=x2 thì => y1=-y2=> y1+y2=



Luật 4: Nếu P1=P2 và y10 thì: m=, x3= -2x1, y3=m(x1-x3)-y1



Luật 5 : P1=P2 và y1=0 thì P1+P2=



* Tính chất: Giao hoán, kết hợp, tồn tại phần tử trung hòa, tồn tại phần tử đối (P+P’=P’+P=)



* Phần tử đối: P= => -P=; P=P(x,0)=> -P=P; P=P(x,y)=> -P=P’(x,-y)



1. **Mật mã trên đường cong elliptic:**

**- Sinh cặp khóa:**

* **IN:** tham số hệ thống <p, E(a,b), P, n >
* **OUT:** khóa công khai Q và khóa bí mật d
  + Chọn ngẫu nhiên d [1, n-1]



* + Tính Q = dP
  + Kết quả là: (Q, d)

**- Mã hóa:**

* **IN:** Tham số hệ thống (p, E(a,b), P, n); Khóa công khai Q; Thông điệp m
* **OUT:** bản mã (C1, C2)
  + Biểu diễn m thành điểm M E



* + Chọn ngẫu nhiên k [1, n-1]



* + Tính C1 = kP
  + Tính C2 = M + kQ
  + Kết quả là C = (C1, C2)

**- Giải mã:**

* **IN:** Tham số hệ thống (p, E(a,b), P, n); Khóa bí mật d; Bản mã C = (C1, C2)
* **OUT:** Thông điệp ban đầu m
  + Tính M = C2 – dC1
  + Trích xuất m từ M
  + Kết quả là M

**- Nhúng số vào điểm thuộc đường cong Elliptic:**

Giả sử cần những m vào đường cong E :



* Coi m là hoành độ của điểm cần nhúng, thay m vào phương trình của E
* Tính f(m)= , thực hiện tìm căn bậc hai của f(m) theo thuật toán tìm căn bậc hai sẽ thu được tọa độ của điểm có thể nhúng m vào đó.



* Chỉ có một nửa số phần tử của Fp là thặng dư bậc 2(có căn bậc 2) nên tỷ lệ thành công khi nhúng chỉ là 50-50.

Để giải quyết vấn đề này, người ta đề xuất giải pháp sau:

* Chọn một giới hạn<< p đối với m
* Chọn số nguyên lớn nhất sao cho , tức là



Số nguyên được gọi là hệ số nhúng;



* Đối với mỗi giá trị*m* cần nhúng, ánh xạ*m* thành số

sao cho là thặng dư bậc 2. Xác suất để không tìm thấy như thế là . Công thức xác suất này chính là lý do để chọn << p.



* Từ điểm đã nhúng, tìm lại *m* bằng công thức

# **Câu 9: Chuẩn mật mã RSA (theo PKCS #1)**

1. **Các phép biến đổi cơ sở:**

**Khóa RSA:**

+Khóa công khai:



+Khóa bí mật bộ 2:

+Khóa bí mật bộ 5:



**Hàm chuyển đổi dữ liệu cơ sở:**

**OS2IP: Octet String To Integer Primitive: x = OS2IP(X)**



**I2OSP: Integer To Octet String Primitive: X = I2OSP(x, xLen),x≥0**

**Phép mã hóa, giải mã cơ bản(RSA Encryption/Decryption Primitive):**

Mã hóa: c = RSAEP(KP, m), c = me mod n

Giải mã: m = RSADP(KS, m), m = cd mod n

Giải mã dùng khóa bộ 5:



1. **Lược đồ EME-OAEP, mã hóa và giải mã**

**EM = EME-OAEP-ENCODE(M, L)**

**Kích thước cácthành phần (octet):**

* *emLen = k*, *seedLen=hLen*, *dbLen = k - hLen -1*
* *lHash = Hash(L)(kích thước phụ thuộc vào hàm băm)*
* *PS = 0x00.00....01*
* *min(psLen)=1*, *max(mLen)=k-2hlen-2*

**Các bước biến đổi M**

* *Ktra kích thước M*, *xác định psLen và PS, tính lHash*, *sinh ngẫu nhiên Seed*
* *dbMash=MGF(Seed, dbLen)*, *maskedDB=DB*⊕*dbMask*
* *seedMask=MGF(maskedDB,seedLen)*, *maskedSeed =Seed*⊕*seedMask*
* *EM=00||maskedSeed||maskedDB*

**Các bước tìm lại M:**

* Kiểm tra octet đầu tiên có phải 00
* Kiểm tra hLen octet tiếp theo (độ dài maskedSeed)
* Seed = SeedMask ⊕ MaskedSeed, DB = MaskedDB ⊕ DbMask
* Kiểm tra lHash: kiểm tra giá trị băm có phải của L không
* Kiểm tra PS: sau lHash, kiểm tra giá trị khác 0 đầu tiên có phải 01, nếu là 01 thì sau đó là M, nếu k phải 01 thì k thỏa mãn.

**RSA-OAEP-ENCRYPT:**

1. Kiểm tra kích thước của M
2. EM = EME-OAEP-ENCODE(M, L)
3. m = OS2IP(EM)
4. c = RSAEP(Kp, m)
5. C = I2OSP(c, k)

**RSA-OAEP-DECRYPT:**

1. c = OS2IP(C)
2. m = RSADP(Ks, c)
3. EM = I2OSP(m,k)
4. M = EME-OAEP-DECODE(EM, L)

1. **Lược đồ EMSA-PSS, ký số và kiểm tra chữ ký**

**Thành phần của lược đồ**: *M: Message*, *salt: ngẫu nhiên- kích thước sLen*, *pad1: 8 octet 00*, *pad2: 00.00...00.01*, *Hash()*, *MGF()*

**Kích thước các thành phần***: mLen = bất kì, sLen = tùy chọn, hLen, dbLen=emLen-hLen-1*

**EM=EMSA-PSS-ENCODE(M, emBits) (emBits:độ dài tối đa của OS2IP(EM)).**

* mHash = Hash(M), salt = sinh ngẫu nhiên, M’ = pad1||mHash||salt
* H = Hash(M’)
* Pad2 = emLen – hLen – 1 – sLen, padding2 min = 1
* DB = pad2||salt
* dbMask = MGF(H,dbLen)
* MaskedDB = DB ⊕ dbMask
* EM = MaskedDB||H||bc (1octet)

**EM= EMSA-PSS-VERIFY(M,EM,emBits)**

* Kiểm tra emLen < hLen+sLen+2?: nhỏ hơn thì k thỏa mãn
* Kiểm tra octet ngoài cùng bên phải có phải 0xbc
* H=EM[…]: tính từ bit thứ 2 bên phải hLen octet là độ dài hLen
* DB = maskedDB ⊕ MGF(H,dbLen)
* DB hợp lệ?: Ktra octet đầu tiên khác 0 từ trái qua phải của DB, nếu = 01 thì là hợp lệ, ngay sau 01 là giá trị salt.
* M = salt||mHash||pad1
* H’ = Hash(M’)
* So sánh H và H’, nếu trùng thì ta tìm M từ giá trị H’ đó.

**Tạo chữ ký:**

1. EM = EMSA-PSS-ENCODE(M, modBits-1), giả sử n=1024 thì EM->m:1023 bit. Đảm bảo m<n, vì mọi phép toán thực hiện theo modulo n.
2. m = OS2IP(EM)
3. s = RSASP(m)
4. S = I2OSP(s, k)

**Kiểm tra chữ ký:**

1. s = OS2IP(S)
2. m = RSAVP(Kp, s)
3. EM = I2OSP(m, emLen); emLen=⎡(modBits-1)/8⎤
4. EMSA-PSS-VERIFY(M, EM, modBits-1)

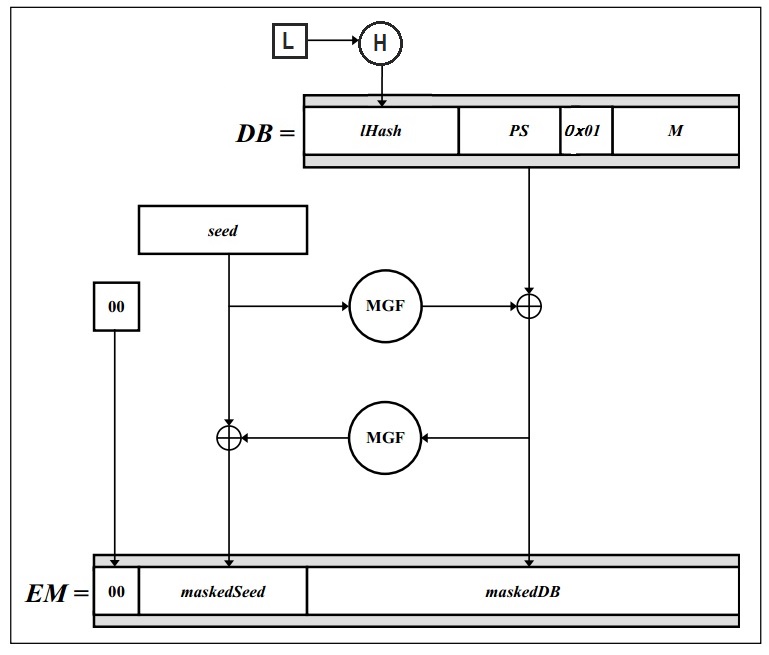
1. **Tại sao cần chuyển đổi bản rõ trước khi áp dụng nguyên thủy mật mã RSA? Vẽ và trình bày lược đồ chuyển đổi bản rõ OAEP.**

Trả lời:

* Công thức mã hóa và giải mã của hệ mật RSA
* c = me mod n
* m = cd mod n
* Từ 2 công thức trên, ta thấy:
* Nếu m = 0 hoặc m = 1 sẽ tạo ra c có giá trị tương ứng là 0, 1.
* Nếu thực hiện mã hóa với số e, m có giá trị nhỏ thì m e< n. Ta có thể tìm ra m bằng cách khai căn bậc e của c.
* Ngoài ra, với hệ mật khóa công khai🡪 kẻ tấn công có thể thực hiện tấn công lựa chọn bản rõ hay tấn công lựa chọn bản mã để lấy được bản rõ ban đầu.

Để tránh gặp phải những vấn đề trên, RSA trên thực tế sẽ sử dụng một hình thức chuyển đổi ngẫu nhiên bản rõ m trước khi thực hiện mã hóa. Quá trình chuyển đổi này phải đảm bảo cho m không rơi vào các giá trị không an toàn.

* **Lược đồ chuyển đổi bản rõ OAEP:**



* *OAEP\_encrypt(k, M, L):*

*Input:*

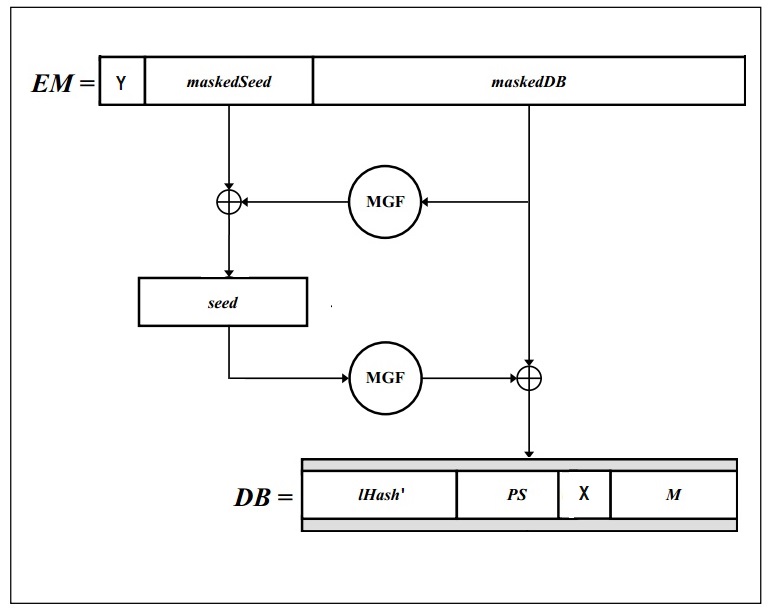
* k: Độ dài của n (n=k octet)
* M: Dài mLen octet
* L: Nhãn tùy chọn liên quan đến M.

*Output:*

* EM: Bản mã có độ dài k octet

*Các bước thực hiện:*

* Kiểm tra độ dài của L.
* Kiểm tra mLen <= k-2hLen-2.
* lHash = Hash (L)
* Sinh PS chứa k-mLen-2hLen byte 0.
* DB = lHash || PS || 0x01 || M
* Sinh chuỗi ngẫu nhiên Seed có độ dài hLen
* dbMask = MGF(seed, k-hLen-1)
* maskedDB = DB⊕dbMask.
* seedMask = MGF(maskedDB, hLen)
* maskedSeed = seed⊕ seedMask.
* EM = 0x00 || maskedSeed || maskedDB.
* **Lược đồ giải chuyển đổi bản rõ OAEP:**



* *OAEP\_decrypt (k, L):*

*Input:*

* k: là độ dài của n (n=k octet)
* L: : nhãn tùy chọn liên quan đến M, có độ dài bằng độ dài hằm băm sử dụng.

*Output:*

* m: thông điệp có độ dài mLen <= k-2hLen-2.

*Các bước thực hiện:*

* Thực hiện kiểm tra độ dài của L.
* Kiểm tra k phải >2hLen + 2
* lHash = Hash(L)
* EM = Y || maskedSeed || maskedDB.
* seedMask = MGF(maskedDB, hLen)
* seed = maskedSeed⊕seedMask.
* dbMask = MGF(seed, k – hLen - 1).
* DB = maskedDB⊕ dbMask.
* DB = lHash’ || PS || 0x01 || M.

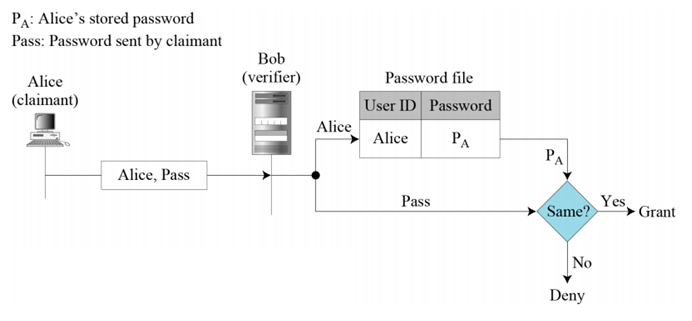
# **Câu 10: Các khái niệm: Tính sống của thực thể; Tính tươi của thông điệp; Xác thực thực thể; Xác thực thông điệp; Phân biệt xác thực bằng mật khẩu và xác thực bằng thách đố - giải đố.**

1. **Tính sống của thực thể:** Tính sống của thực thể A đối với một thực thể B là tính chất cho biết rằng thực thể A đang tham gia vào liên lạc với thực thể B.
2. **Tính tươi của thông điệp:** là khi thực thể nhận thông điệp cho rằng khoảng thời gian giữa thời điểm gửi và thời điểm nhận thông điệp và đủ nhỏ. Với các giao thức thời gian thực, tính tươi của thông điệp được tính bằng vài giây; Với các hệ thống liên lạc mật mã thì tính bằng khóa ngày, khóa giờ…
3. **Xác thực thực thể:** là việc một thực thể thiết lập một giao tiếp sống với một thực thể thứ hai mà định danh của thực thể này đúng là định danh mà thực thể thứ nhất yêu cầu.
4. **Xác thực thông điệp:** là một cơ chế cho phép khẳng định rằng thông điệp không bị thay đổi trong quá trình truyền và bên nhận có thể kiểm tra được nguồn gốc của thông điệp.
5. **Phân biệt xác thực bằng mật khẩu và xác thực bằng thách đố, giải đố:**

|  |  |
| --- | --- |
| **Xác thực bằng mật khẩu** | **Xác thực bằng thách đố - giải đố** |
| - Xác thực bằng mật khẩu nhằm xác thực thực thể.  - Xác thực dựa vào “cái gì đó mà người dùng biết” | - Xác thực thách đố - giải đố kiểm tra tính sống của thực thể.  - Xác thực thông qua tính tươi của thông điệp gửi đi. |
| 2 loại mật khẩu  • Mật khẩu cố định  • Mật khẩu một lần | • Thách đố (challenge) là một giá trị thay đổi theo thời gian không phụ thuộc ý muốn của claimant (được sinh ngẫu nhiên bởi verifier; hoặc giá trị bộ đếm). Challenge là nhãn thời gian tránh được sự tương tác nên thích hợp cho những ứng dụng không tương tác (ví dụ như thư điện tử) nhưng khó khăn trong việc đồng bộ đồng hồ  • Giải đố (response) là kết quả biến đổi challenge và yếu tố bí mật bằng một hàm nào đó |
| Claimant chứng minh bản thân bằng cách cung cấp yếu tố bí mật (mật khẩu). Yếu tố này được truyền trực tiếp qua kênh không an toàn. | Claimant chứng mình biết yếu tố bí mật bằng việc gửi một hàm biến đổi từ yếu tố bí mật (không cần gửi trực tiếp yếu tố bí mật). |

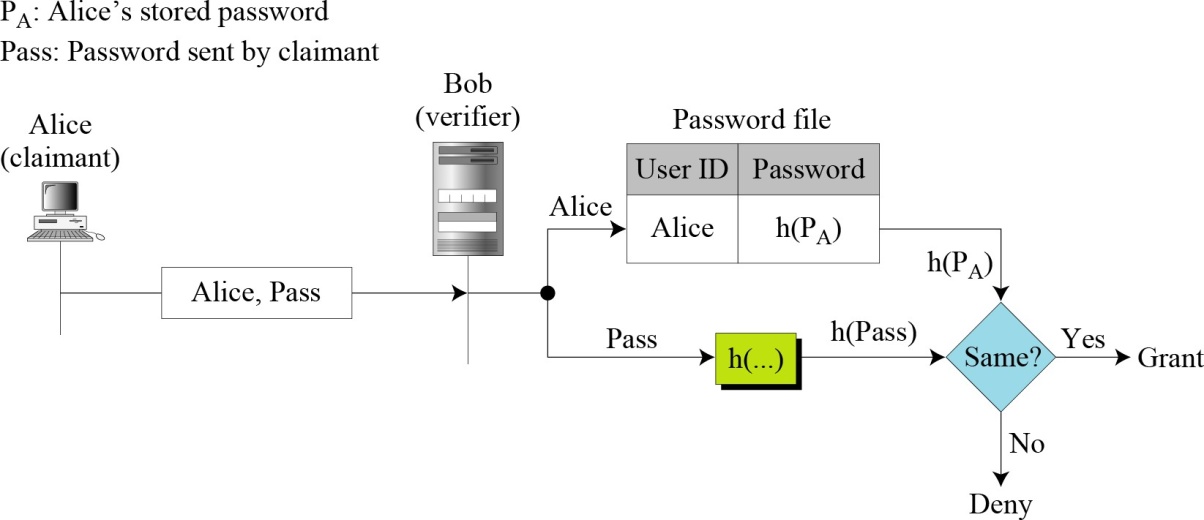
# **Câu 11: Xác thực bằng mật khẩu:**

1. **Lưu mật khẩu dạng rõ:**



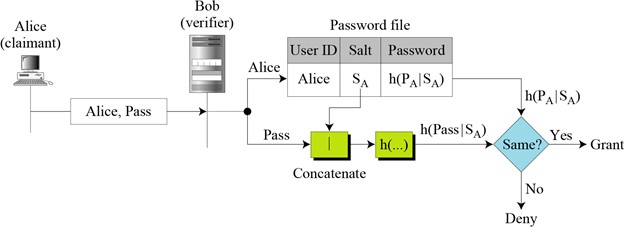
* Điều kiện ban đầu của giao thức:
  + Alice là bên phải xác thực, Bob là bên thực hiện xác thực.
  + Alice và Bob thống nhất với nhau thông tin để xác thực. Trong đó Alice sẽ ghi nhớ một định danh và mật khẩu, phía Bob lưu trữ thông tin này vào cơ sở dữ liệu dưới dạng bản rõ.
* Các bước hoạt động của giao thức:
  + Bước 1: Alice muốn liên lạc với Bob sẽ gửi một thông điệp gồm có định danh và mật khẩu để xác thực.
  + Bước 2: Bob nhận được thông điệp yêu cầu xác thực của Alice, sẽ lấy định danh so sánh với cơ sở dữ liệu nếu không tồn tại thì xác thực thất bại. Nếu 2 giá trị này giống nhau thì xác thực thành công, cho phép truy cập. Nếu không xác thực thất bại, truy cập bị từ chối.
* Cơ sở cho quyết định cuối cùng của bên xác thực:
  + Mật khẩu lưu dưới dạng rõ, chỉ dựa vào sự so sánh 2 bản rõ mật khẩu của Alice gửi và mật khẩu lưu trên CSDL của Bob.
* Khả năng tấn công
  + Chặn bắt mật khẩu
  + Chiếm file chứa mật khẩu

1. **Lưu mật khẩu dạng băm:**



* Điều kiện ban đầu của giao thức:
  + Alice là bên phải xác thực, Bob là bên thực hiện xác thực.
  + Alice và Bob thống nhất với nhau thông tin để xác thực. Trong đó Alice sẽ ghi nhớ một định danh và mật khẩu, phía Bob lưu trữ thông tin này vào cơ sở dữ liệu (mật khẩu được lưu dưới dạng băm h(PA)).
* Các bước hoạt động của giao thức:
  + Bước 1: Alice muốn liên lạc với Bob sẽ gửi một thông điệp gồm có định danh và mật khẩu để xác thực.
  + Bước 2: Bob nhận được thông điệp yêu cầu xác thực của Alice, sẽ lấy định danh so sánh với cơ sở dữ liệu nếu không tồn tại thì xác thực thất bại. Nếu tồn tại Bob sẽ lấy mật khẩu sau khi đi qua hàm băm h(Pass) so sánh với giá trị mật khẩu h(PA) lưu trong cơ sở dữ liệu. Nếu 2 giá trị này giống nhau thì xác thực thành công, cho phép truy cập. Nếu không xác thực thất bại, truy cập bị từ chối.
* Cơ sở cho quyết định cuối cùng của bên xác thực:
  + Căn cứ vào tính chất kháng tiền ảnh thứ 2 của hàm băm. Khi đem so sánh nếu H(pass)=H(PA) chứng minh rằng Pass= PA, PA là giá trị bí mật chỉ được chia sẻ giữa Alice và Bob (Alice lại chỉ ra được giá trị bí mật PA) nên người này chính là Alice.
* Tấn công từ điển sử dụng bảng băm sẵn (precomputed hash table) lên cơ sở dữ liệu tài khoản người dùng trong trường hợp có được cơ sở dữ liệu đó. Precomputed hash table là một danh sách chứa các giá trị hash tương ứng với giá trị mật khẩu nào đó (đã biết). Khi ta có được cơ sở dữ liệu cần tấn công ta sẽ so sánh các hàm hash trong đó với Precomputed hash table. Nếu 2 hàm hash trùng nhau sẽ suy ngược được mật khẩu cần tìm.

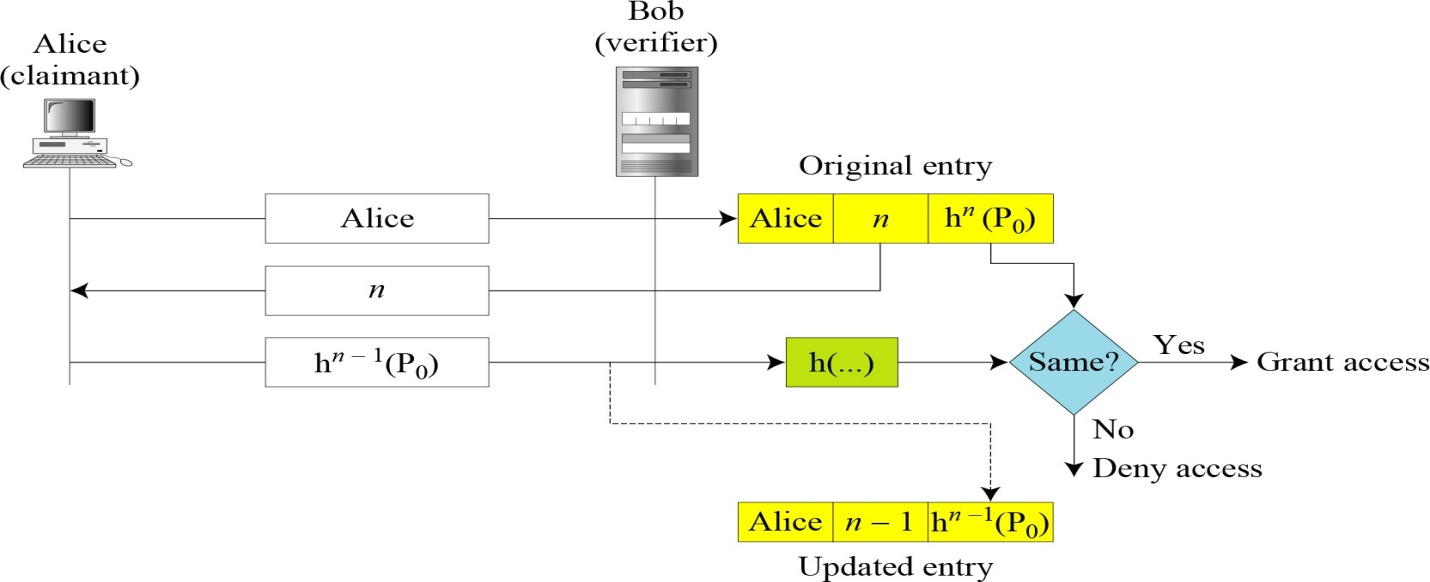
1. **Lưu mật khẩu dạng băm có salt:**



* Điều kiện ban đầu của giao thức:
  + Alice là bên phải xác thực, Bob là bên thực hiện xác thực.
  + Alice và Bob thống nhất với nhau thông tin để xác thực. Trong đó Alice sẽ ghi nhớ một định danh và mật khẩu, phía Bob lưu trữ thông tin này vào cơ sở dữ liệu (mật khẩu được lưu dưới dạng băm kết hợp với giá trị ngẫu nhiên SA: h(PA| SA)). SA là giá trị ngẫu nhiên được lưu trong cơ sở dữ liệu của Bob.
* Các bước hoạt động của giao thức:
  + Bước 1: Alice muốn liên lạc với Bob sẽ gửi một thông điệp gồm có định danh và mật khẩu để xác thực
  + Bước 2: Bob nhận được thông điệp yêu cầu xác thực của Alice, sẽ lấy định danh so sánh với cơ sở dữ liệu nếu không tồn tại thì xác thực thất bại. Nếu tồn tại Bob sẽ lấy giá trị Pass Alice gửi nối với SA và cho đi qua hàm băm – H(Pass| SA) sau đó so sánh giá trị này với giá trị h(PA| SA) lưu trong cơ sở dữ liệu. Nếu hai giá trị này giống nhau thì xác thực thành công, cho phép truy cập. Nếu không xác thực thất bại, truy cập bị từ chối.
* Cơ sở cho quyết định cuối cùng của bên xác thực: Căn cứ vào tính chất kháng tiền ảnh thứ 2 của hàm băm. Nếu H(Pass| SA)=h(PA| SA) thì Bob khẳng định rằng Pass= PA. PA là giá trị bí mật chỉ được chia sẻ giữa Alice và Bob. Nên người gửi chính là Alice.
* Ý nghĩa của việc sử dụng salt trong giao thức:

Sử dụng “Salt” trong giao thức làm tăng độ phức tạp tạo ra một mật khẩu mạnh hơn để ngăn ngừa các tấn công.

1. **Mật khẩu một lần Lamport**



* Điều kiện ban đầu của giao thức:
  + - Alice là bên phải xác thực, Bob là bên thực hiện xác thực.
    - Alice và Bob thống nhất với nhau thông tin để xác thực. Trong đó Alice sẽ ghi nhớ một định danh và mật khẩu, phía Bob lưu trữ thông tin này vào cơ sở dữ liệu (mật khẩu được lưu ở dạng băm n lần của P0 : hn(P0)), giá trị n được lưu trữ trong cơ sở dữ liệu của Bob.
  + Các bước hoạt động của giao thức:
    - Bước 1: Alice muốn liên lạc với Bob sẽ gửi một thông điệp gồm có định danh và mật khẩu để xác thực.
    - Bước 2: Bob nhận được thông điệp yêu cầu xác thực của Alice, sẽ lấy định danh so sánh với cơ sở dữ liệu nếu không tồn tại thì xác thực thất bại. Nếu tồn tại Bob đọc trong cơ sở dữ liệu tương ứng giá trị n và gửi cho Alice giá trị n này.
    - Bước 3: Alice nhận được giá trị n sẽ băm mật khẩu (n-1) lần : hn-1(P0) và gửi cho Bob.
    - Bước 4: Bob nhận được kết quả sẽ đem băm thêm 1 lần sau đó so sánh kết quả này với giá trị hn(P0) trong cơ sở dữ liệu. Nếu 2 giá trị này giống nhau thì xác thực thành công, cho phép truy cập đồng thời Bob cập nhật lại cơ sở dữ liệu giá trị n sẽ thay thế bằng giá trị (n-1), hn(P0) thay thế bằng hn-1(P0). Nếu không xác thực thất bại, truy cập bị từ chối.
  + Cơ sở cho quyết định cuối cùng của bên xác thực: Dựa trên tính chất kháng tiền ảnh thứ 2 của hàm băm. Nếu hn-1(Pass)= hn-1(P0) vậy chứng tỏ Pass= P0. P0 là giá trị bí mật chỉ được chia sẻ giữa Alice và Bob. Nên người gửi chính là Alice.
  + Tấn công kiểu man in the middle lên giao thức:

Khi Bob gửi cho Alice giá trị n, kẻ tấn công sẽ chặn gói tin này và thay đổi giá trị n thành (n-1) sau đó gửi lại cho Alice. Alice sẽ gửi lại cho Bob giá trị hn-2(P0). Kẻ tấn công sẽ lấy giá trị này sửa đổi thành hn-1(P0) sau đó gửi lại cho Bob đồng thời lưu lại giá trị hn-2(P0). Cứ như vậy kẻ tấn công sẽ có được mọi giá trị mật khẩu trong mỗi phiên giao dịch của Alice và Bob.

# **Câu 12: Các giao thức khác: trình bày hoạt động của giao thức. Giải thích quyết định của bên xác thực.**

1. **Xác thực lẫn nhau 3 bước ISO:**

* Giả thiết:
  + A có chứng chư số khóa công khai CertA
  + B có chứng thư số khóa công khai CertB
* Yêu cầu: A và B đạt được sự xác thực lẫn nhau
* Cách thức:
  + B → A: NB;
  + A → B: CertA, TokenAB
  + B → A: CertB, TokenBA
  + Trong đó:
    - TokenAB = NA || NB || B || sigA(NA || NB || B)
    - TokenBA = NB || NA || A || sigB(NB || NA || A)

1. **Giao thức Woo-Lam:**

* Giả thiết:
  + A và B cùng tin cậy bên thứ ba T
  + A và T chia sẻ khóa KAT
  + B và T chia sẻ khóa KBT
* Yêu cầu: B xác thực được A
* Cách thức:
* A → B: "Alice"
* B → A: NB
* A → B: {NB}KAT
* B → T: {"Alice", {NB}KAT}KBT
* T → B: {NB}KBT
* B giải mã thông điệp của T
  + Thu được NB thì A được xác thực;
  + Ngược lại, A bị từ chối.

1. **Thêm phụ gia vào giao thức:**

* Giả thiết: U và H đã thỏa thuận mật khẩu PU
* Yêu cầu:
  + H và U xác thực lẫn nhau
  + H và U thỏa thuận được khóa K bí mật
  + Không truyền PU và H(PU) qua kênh không an toàn.
* Cách thức:
* U → H: IDU, {εU}PU //εU = ngẫu nhiên
* H → U: {{K}εU }PU //K = ngẫu nhiên
* U → H: {NU}K
* H → U: {NU, NH}K
* U → H: {NH}K //kiểm tra NU trước
* H giải mã và kiểm tra NH

1. **Thỏa thuận khóa có xác thực:**

* Giả thiết:
  + Tham số hệ thống
    - Nhóm hữu hạn với phần tử sinh a
    - Thuật toán mật mã đối xứng E
  + Alice, Bob có chứng thư số CertA, CertB
* Yêu cầu:
  + Alice, Bob xác thực lẫn nhau
  + Alice, Bob trao đổi khóa bí mật K
* Cách thức thực hiện:
* Alice → Bob: ax
* Bob → Alice: ay, CertB, EK(sigB(ax, ay))
* Alice → Bob: Cert A, EK(sigA(ax, ay))
  + K = axy = ayx

# **Câu 13: Cộng điểm trên đường cong Elliptic**

**Ví dụ 1:**Cho đường cong Elliptic . Giả sử *P* = (5,3) và *Q =*(11*,3*) thuộc đường cong trên. Tính các giá trị  và 2*P*.

**Lời giải:**

- Tính giá trị :

+ Với *P* = (5,3) và *Q =*(11*,3*) là hai điểm trên đường cong E17(3,5). Ta có:



+ Vì  nên ta có:



+ Với , ta có:



+ Vậy .

- Tính giá trị :

+ Với *P* = (5,3) là một điểm trên đường cong E17(3,5). Ta có 

+ Tính giá trị :



+ Với , ta có:



+ Vậy .

**Ví dụ 2:**Cho đường cong E23(1,1). Giả sử *P* = (0,1) và *Q* = (7,11) là các điểm nằm trên cong Eliptic E23(1,1). Hãy tìm giá trị *k* để .

**Lời giải:**

Với  và . Để tìm giá trị *k* sao cho  thì ta phải nhân *P* lên nhiều lần cho tới khi bằng *Q*:

- Tính 

+ Tính giá trị :



+ Với , ta có:



+ Vậy .

- Tính 

+ Tính giá trị :



+ Với , ta có:



+ Vậy .

- Tính 

+ Tính giá trị :



+ Với , ta có:



+ Vậy .

- Tính 

+ Tính giá trị :



+ Với , ta có:



+ Vậy .

- Tính 

+ Tính giá trị :



+ Với , ta có:



+ Vậy .

|  | (nếu)  (nếu) |  |  |  |
| --- | --- | --- | --- | --- |
| 1 |  |  |  | (0,1) |
| 2 | 12 | 6 | 19 | (6,19) |
| 3 | 3 | 3 | 13 | (3,13) |
| 4 | 4 | 13 | 16 | (13,16) |
| 5 | 10 | 18 | 3 | (18,3) |
| 6 | 18 | 7 | 11 | (7,11) |

- Kết luận: *k* = 6

# **Câu 14: Phương pháp nhúng số lên điểm thuộc đường cong trên Fp**

Trên , ta có thể biến đổi để đưa mọi đường cong về dạng ().

, với điều kiện  và 

Giả sử cần chuyển số  thành một điểm trên đường cong , ta coi  là hoành độ của điểm cần tìm, thay vào () được . Tiếp đó sử dụng thuật toán Tonelli-Shank (xem phần 4) để tìm tung độ .

Nhúng một số lên một điểm thuộc đường cong elliptic

Xét đường cong  thuộc  và số cần nhúng là .

Thay  vào phương trình đường cong, ta tìm được . Sử dụng thuật toán Tonelli-Shank, tính được hai căn bậc hai của 35 là 13 và 54. Vậy có hai điểm có thể được sử dụng để nhúng  là  và .

Tuy nhiên, khó khăn ở đây là chỉ có khoảng một nửa số phần tử của  là thặng dư bậc 2 (có căn bậc 2). Do vậy, nếu số cần nhúng nằm trong khoảng từ 0 đến  thì sẽ có khoảng một nửa trường hợp không thể nhúng thành công. Để giải quyết vấn đề này, người ta đề xuất giải pháp sau.

1. Chọn một giới hạn nmax = p đối với ;
2. Chọn số nguyên  lớn nhất sao cho , tức . Số nguyên  được gọi là hệ số nhúng;
3. Đối với mỗi giá trị  cần nhúng, ánh xạ  thành số  sao cho  là thặng dư bậc 2. Xác suất để không tìm thấy  như thế là . Công thức xác suất này chính là lý do để chọn .
4. Từ điểm đã nhúng, tìm lại  bằng công thức .

Nhúng điểm với hệ số nhúng

Xét đường cong  thuộc .

Giả sử cần nhúng số  thuộc tập hợp  lên các điểm thuộc đường cong  đã cho. Tức là ở đây ta có nmax = 5.

Nếu ta không sử dụng hệ số nhúng, mà tìm  thì kết quả sẽ như trong bảng dưới đây.

*Bảng. Nhúng số trong trường hợp không sử dụng hệ số nhúng*

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | 2 | không tồn tại |
|  | 6 | 26, 41 |
|  | 16 | 4, 63 |
|  | 38 | không tồn tại |
|  | 11 | không tồn tại |
|  | 8 | không tồn tại |

Có thể thấy rằng trong số 6 trường hợp giá trị của  thì chỉ nhúng thành công trong 2 trường hợp, còn lại là không thành công.

Nếu sử dụng hệ số nhúng thì giá trị của hệ số này sẽ là . Việc nhúng các số  sẽ được thực hiện như trong bảng. Trong bảng này chỉ xét một giá trị căn bậc hai của .

*Bảng Nhúng số trong trường hợp có sử dụng hệ số nhúng*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | 0 | 10 | 1 | 6 | 26 | (1, 26) |  |
|  | 11 | 21 | 11 | 26 | 19 | (11, 19) |  |
|  | 22 | 32 | 24 | 29 | 37 | (24, 37) |  |
|  | 33 | 43 | 33 | 59 | 40 | (33, 40) |  |
|  | 44 | 54 | 47 | 49 | 60 | (47, 60) |  |
|  | 55 | 65 | 55 | 47 | 39 | (55, 39) |  |

Kết quả cho thấy mọi giá trị của  đều có thể được nhúng thành công vào một điểm nhất định thuộc đường cong đã cho.

Việc xác định giá trị của  được thực hiện như sau:

Các bước tìm 

**Bước 1**. Chọn 

**Bước 2**. Tính kí hiệu Legendre . Nếu  thì chuyển đến Bước 3; nếu không thì  chính là giá trị cần tìm, chuyển đến Bước 4.

**Bước 3**. Nếu  thì thông báo "Không tồn tại " và chuyển đến Bước 4; nếu không thì tăng  thêm 1 và trở lại Bước 2.

**Bước 4**. Kết thúc.

# **Câu 15: Hệ mật ElGamal**

Là thuật toán mật mã khóa công khai dựa trên thuật toán Diffie-Hellman

* **Sinh cặp khóa cho Alice:**
  + Chọn số nguyên tố **p**, phần tử sinh **g** của **Z\*p**
  + Chọn ngẫu nhiên **x ∈ [1, p-1]**
  + Tính **h = gx**
  + Khóa bí mật là **KSA = (p, g, x)**
  + Khóa công khai là **KPA = (p, g, h)**
* **Mã hóa (thông điệp gửi cho Alice)**
  + Biết khóa công khai là **KPA = (p, g, h)**
  + Chọn ngẫu nhiên **y∈ [1, p-1]**
  + Tính khóa chung **s = hy**
  + Bản mã: **c = (c1, c2) = (gy, m**⋅**s)**
* **Giải mã (bởi Alice)**
  + Bản mã **c = (c1, c2)**
  + Dùng khóa bí mật là **KSA = (p, g, x)**
  + Tính khóa chung **s = c1x**
  + Bản rõ: **m = c2⋅s-1**

**Lược đồ kí số ElGamal**: tuy cùng tên với thuật toán mã hóa ElGamal nhưng bản chất thuật toán rất khác biệt

* **Sinh cặp khóa cho Alice:**
  + Chọn số nguyên tố **p**, phần tử sinh **g** của **Z\*p**
  + Chọn ngẫu nhiên **x ∈ [1, p-1]**
  + Tính **h = gx**
  + Khóa bí mật là **KSA = (p, g, x)**
  + Khóa công khai là **KPA = (p, g, h)**
* **Thực hiện kí số (bởi Alice)**

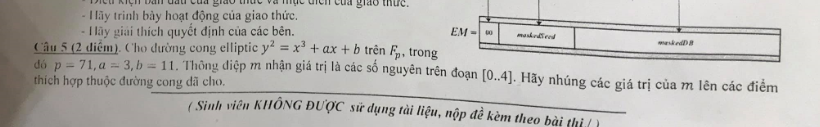
Thông điệp **m**, sử dụng **KSA = (p, g, x)**

* 1. Sinh ngẫu nhiên: **1<k<p-1; (k,p-1)=1**
  2. Tính **r=gk (mod p)**
  3. Tính **s = (m-xr)k-1 (mod p-1)**
  4. Nếu **s=0** thì trở lại bước 1
  5. Chữ kí số lên thông điệp **m** là **(r,s)**
* **Kiểm tra chữ kí (bởi bất kì ai)**

Thông điệp **m**, sử dụng **KPA = (p, g, h)**

* 1. Kiểm tra: **0< r<p; 0< s <p-1**

Kiểm tra: **gm = hr⋅rs (mod p)**

****

**m = {0,1,2,3,4}**

**mmax = 4 << p=79  
  
hệ số nhúng sẽ là** 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| m | lm | lm+l-1 | m’ | f(m’) |  | M |  |
|  |  |  |  |  |  | (, ) |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |