Câu 31: Trình bày sự hỗ trợ mật mã bởi thư viện Crypto++.

Crypto++ hỗ trợ đầy đủ kiến trúc 32 bit và 64 bit cho nhiều hệ điều hành khác nhau như Apple(MAC OS X và iOS),BSD, Linux, Solaris và Windows.

Ngôn ngữ hỗ trợ C++

Khai báo trước khi sử dụng: include "duong dan den thu vien", using cryptoPP::x (x là lớp mà bạn muốn sử dụng được hỗ trợ trong thư viên crypto++

Thư viện Crypto++ hỗ trợ các thuật toán được sử dụng trong mật mã như:

- Hàm băm hỗ trợ các thuật toán như SHA-1,SHA-2,Tiger... Ví dụ như trong SHA-1 có các hàm DigestSize() count trả về kích thước sau khi băm, byte * CreateUpdateSpace(size_t &size) yêu cầu không gian để thêm đầu vào.
- Mật mã đối xứng: AES,RC6... Một số hàm trong AES:

  byte pass(AES::BlockSize) khai báo một biến pass có độ dài bằng độ dài của block là 16 bit, CryptoPP::AES::Decryption aesDecryption(key,AES::DEFAULT_KEYLENGTH); tạo đối tượng giải mã.
- Mật mã khóa công khai: RSA, DSA, ElGamal... Một số hàm hỗ trợ trong RSA hàm tạo cặp khóa public và private RSA::PrivateKey privateKey(params), RSA::PublicKey publicKey(params).
- Các bộ sinh số giả ngẫu nhiên (PRNG): ANSI X9.17, RandomPool... ví dụ AutoSeededRandomPool rng Tạo một chuỗi bit ngẫu nhiên

Câu 29:

Cho hệ mật RSA 32 bít và khối dữ liệu: D=0x1234ABCD. Áp dụng hàm OS2IP() để chuyển khối dữ liệu trên thành số nguyên. Áp dụng hàm I2OSP() cho kết quả vừa tìm được để tìm lại khối dữ liệu ban đầu.

TL: Hàm OS2IP(): chuyển từ octect string sang số nguyên

tách D thành 4 byte D1= 12h = 18, D2=34h=52, D3=ABh= 171, D4=CDh= 205

⇨ $d= 18*256^3 + 53*256^2 + 171*256 + 205*256^0 =3054411741$

Hàm I2OSP() :chuyển từ số nguyên sang octect string

Thực hiện chia số nguyên cho 256 lấy phần dư => đổi sang cơ số 16 và viết ngược từ dưới lên

305411741%256 = 205 = CDh
1193131%256 = 171 = ABh
4660%256= 52 = 34h
18 = 12h
D= 0x1234ABCD

Câu 28: 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.

❖ **TL**: Công thức mã hóa và giải mã của hệ mật RSA

$c = m^e \bmod n$

$m = c^d \bmod n$

❖ Từ 2 công thức trên:

- 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 $m^e < n$

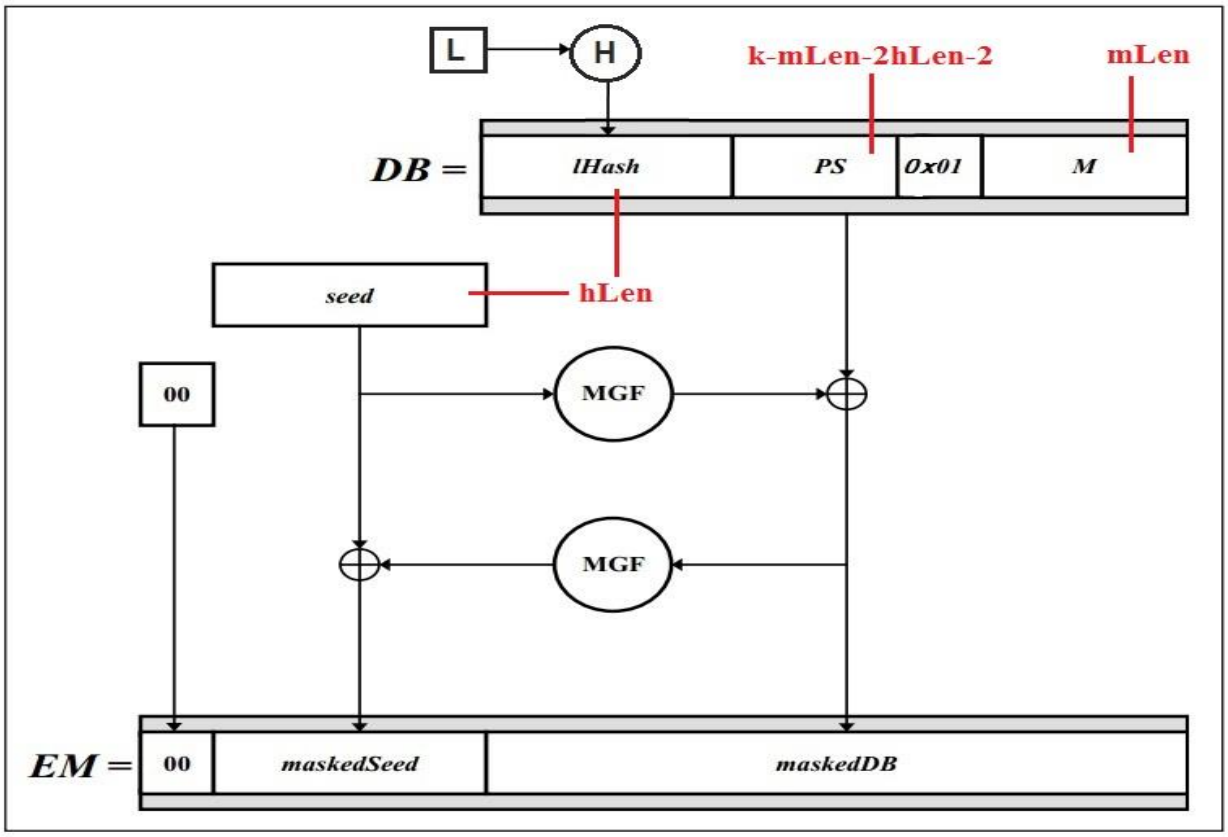
⇒Có thể tìm ra m bằng cách khai căn bậc e của c.

- Ngoài ra, với hệ mật khóa công khai RSA kẻ tấn công có thể thực hiện tấn công lựa chọn bản rõ hay tấn công lựa chọn bản mã để tìm ra được bản rõ ban đầu.

Lược đồ chuyển đổi bản rõ OAEP

Mã hóa

❖



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.

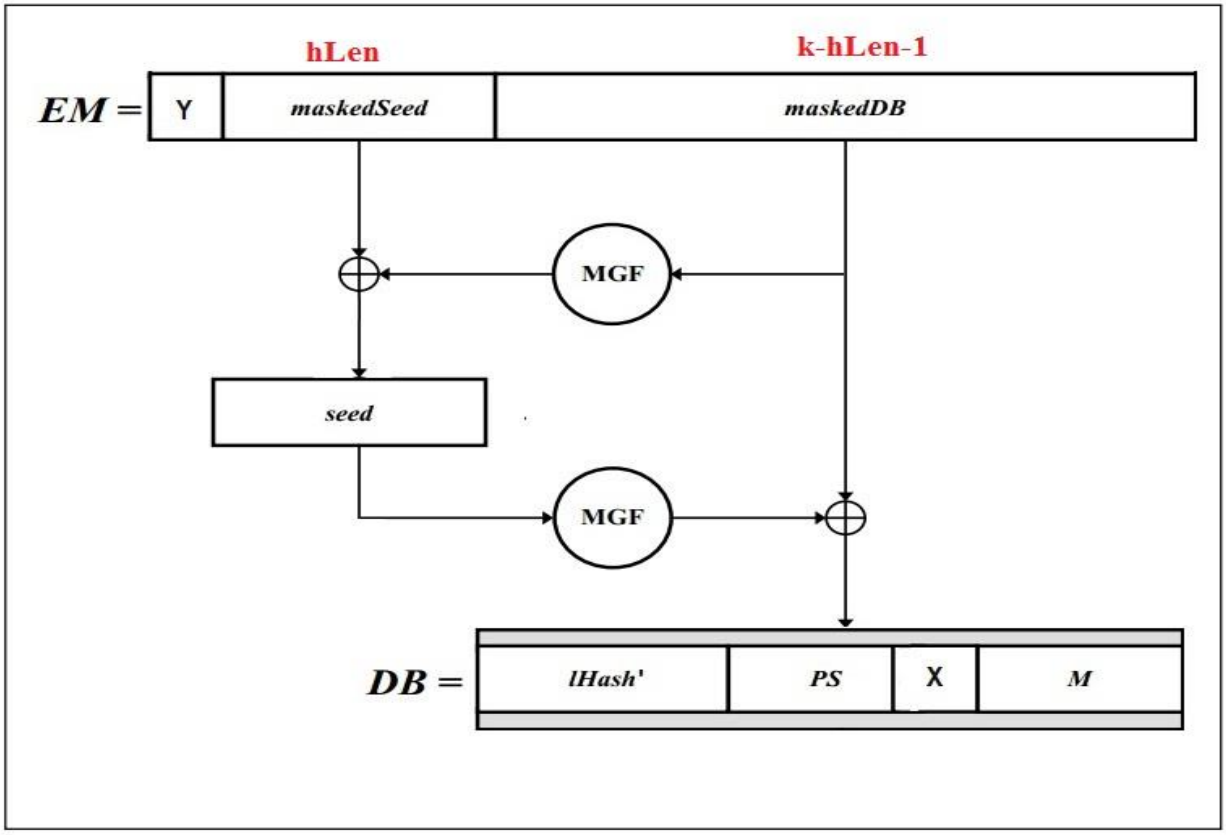
Output:

- 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.

Giải mã



❖ *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 33: Trình bày các nhóm hàm của thư viện GMP (hoặc các nhóm phương thức của lớp BigInteger trong Java) cần thiết cho việc cài đặt thuật toán RSA.

TL:

1. *void mpz_sub_ui (mpz_t rop, const mpz_t op1, unsigned long int op2)*
   + Hàm này để thực hiện phép trừ một số kiểu mpz_t cho một số nguyên: rop = op1 -op2.
   + Dùng để tính p-1 và q-1.

2. *void mpz_ramdom*
   void mpz_powm (mpz_t rop, const mpz_t base, const mpz_t exp, const mpz_t mod)
   + Dùng để tính rop=(base^exp)mode(mode)

3. *void mpz_nextprime (mpz_t rop, const mpz_t op)*
   + Dùng để tính số nguyên tố tiếp theo sau của một số op và lưu vào rop

4. *void mpz_gcd (mpz_t rop, const mpz_t op1, const mpz_t op2)*
   + Hàm này để tính op1/op2 và lưu vào rop để kiểm tra nguyên tố

cùng nhau

5. *int mpz_invert (mpz_t rop, const mpz_t op1, const mpz_t op2)*
+ Hàm tính mod^-1: rop = (op1^-1)mod(op2).

6. *void mpz_urandomb (mpz_t rop, gmp_randstate_t state, mp_bitcnt_t n*)
+ Tạo ra một số ngẫu nhiên phân bố đồng đều trong khoảng 0 tới 2n - 1.
+ Tình trạng biến phải được khởi tạo bằng cách gọi một trong những hàm gmp_randinit trước khi gọi hàm này

Câu 25: Hãy cho biết chức năng của giao thức SSH. Trình bày cách thức client xác thực server trong giao thức SSH. Trong giao thức SSH, có những cách nào cho phép server xác thực client? Lấy ví dụ về SSH server và SSH client. Chỉ ra vai trò của mật mã khóa công khai và mật mã đối xứng trong giao thức SSH

TL:

- Chức năng của SSH:

Cung cấp giao diện Shell điều khiển cho người dùng truy cập từ xa. SSH mã hóa mọi thông tin truyền, nhận giữa máy chủ truy cập (SSH Server) và máy khách từ xa (SSH Client), nhằm ngăn chặn các phương pháp tấn công nghe lén trên đường truyền.

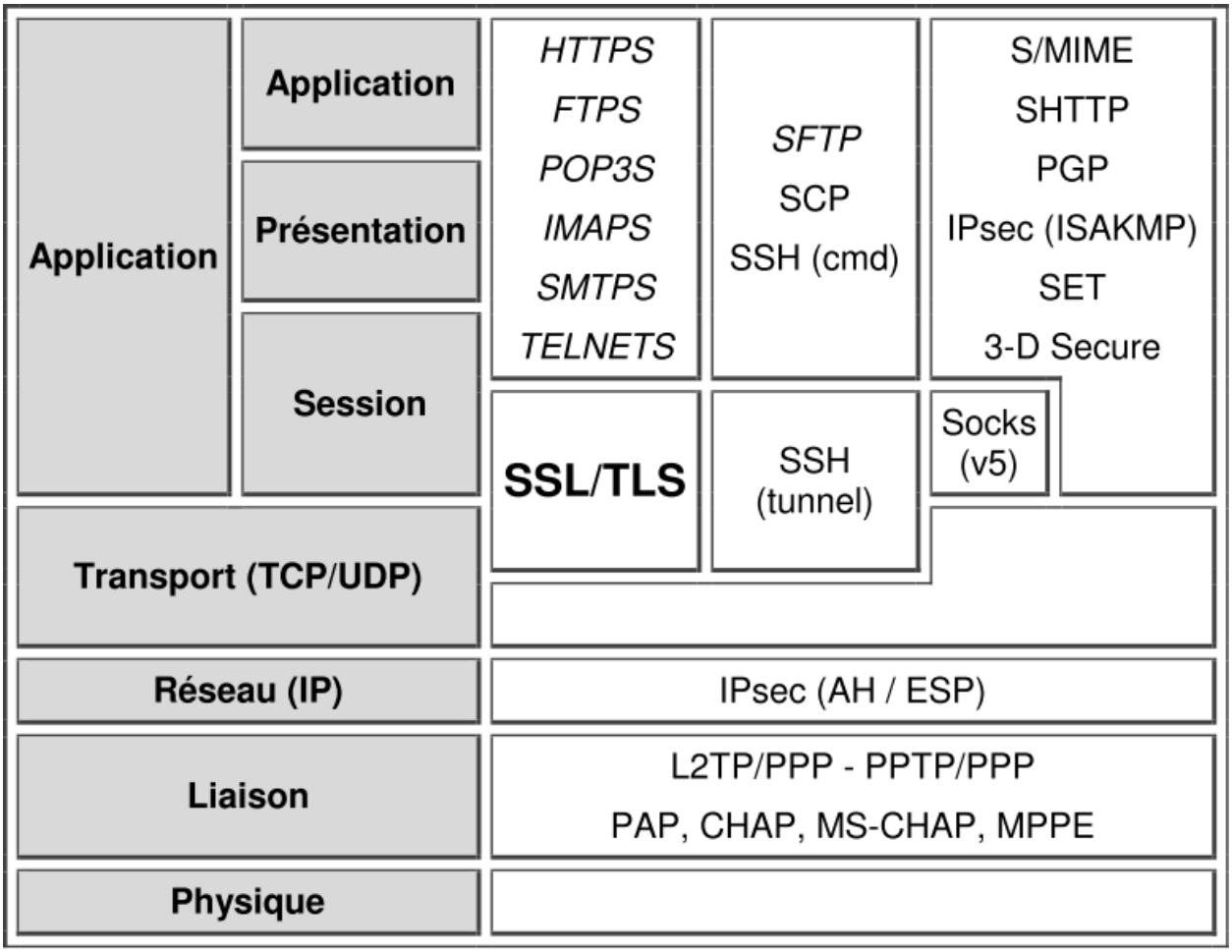
- Cách thức client xác thực server:
- Có 4 cách để server xác thực client:

  Methods
  • public key
  • password
  • host based
  • keyboard-interactive (RFC 4256)

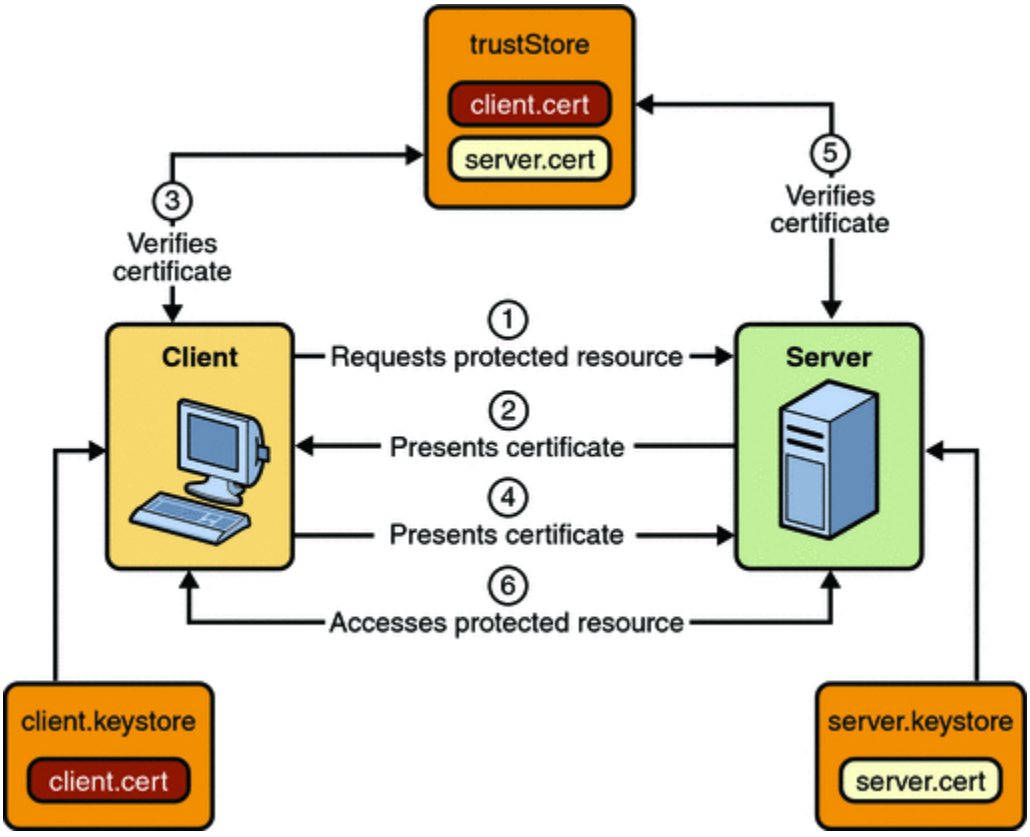
Câu 26: Hãy cho biết chức năng của giao thức HTTPS. Vẽ sơ đồ thể hiện vị trí của giao thức SSL/TLS trong mô hình phân lớp. Trình bày cách thức client xác thực server trong giao thức HTTPS. Giao thức HTTPS có cho phép server xác thực client không? Chỉ ra vai trò của mật mã khóa công khai và mật mã đối xứng trong giao thức HTTPS.

TL:

- Chức năng của giao thức HTTPS là đảm bảo an toàn cho thông tin truyền qua môi trường internet
  - Tính bí mật: sử dụng encryption để đảm bảo thông điệp truyền giữa client và server không bị đọc trộm
  - Tính toàn vẹn: sử dụng hashing để đảm bảo thông điệp k bị sửa đổi, mất mát
  - Xác thực : sử dụng *digital certificate* để giúp client có thể tin tưởng rằng server/website mà họ đang truy cập thực sự là server/website mà họ mong muốn vào, chứ không phải bị giả mạo.
- Sơ đồ vị trí của SSL/TLS trong mô hình phân lớp:



- Cách thức client xác thực server:



Các bước:

1. Client gửi request cho một secure page (có URL bắt đầu với *https://*)
2. Server gửi lại cho client certificate của nó.
3. Client gửi certificate này tới CA (mà được ghi trong certificate) để kiểm chứng.

   Giả sử certificate đã được xác thực và còn hạn sử dụng hoặc client vẫn cố tình truy cập mặc dù Web browser đã cảnh báo rằng không thể tin cậy được certificate này (do là dạng self-signed SSL certificate hoặc certificate hết hiệu lực, thông tin trong certificate không đúng…) thì mới xảy ra bước 4 sau.

4. Client tự tạo ra ngẫu nhiên một *symmetric encryption key*, rồi sử dụng *public key* (trong certificate) để mã hóa symmetric key này và gửi về cho server.
5. Server sử dụng *private key* (tương ứng với public key trong certificate ở trên) để giải mã ra symmetric key ở trên.
6. Sau đó, cả server và client đều sử dụng symmetric key đó để mã hóa/giải mã các thông điệp trong suốt phiên truyền thông.

Và tất nhiên, các symmetric key sẽ được tạo ra ngẫu nhiên và có thể khác nhau trong mỗi phiên làm việc với server. Ngoài *encryption* thì cơ chế hashing sẽ được sử dụng để đảm bảo tính *Integrity* cho các thông điệp được trao đổi.

Câu 32: Trình bày sự hỗ trợ mật mã bởi nền tảng .NET Framework

TL:
Thư viện mật mã do .Net hỗ trợ:
Môi trường: .Net Frame work.
Ngôn ngữ được hỗ trợ: Visual C++, Visual Basic, C#.
Khai báo thư viện trước khi sử dụng: using System.Security.Cryptography;

1. .Net hỗtrợ *Hàmbăm*: Class HashAlgorithm

   Ví dụ:
   HashAlgorithm ha = HashAlgorithm.Create("MD5");
2. Mật mã đối xứng: Class SymmetricAlgorithm

   Ví dụ:
   SymmetricAlgorithmsa = SymmetricAlgorithm.Create("DES");
3. Mật mã KCK: Class AsymmetricAlgorithm

   Ví dụ:
   AsymmetricAlgorithma = AsymmetricAlgorithm.Create("RSA");
4. Sinh khóa từ mật khẩu: Class DeriveBytes

Có 2 Class kế thừa từ Class DeriveBytes là PasswordDeriveBytes và Rfc2898DeriveBytes;
Ví dụ:
Rfc2898DeriveBytesrdb = new Rfc2898DeriveBytes(“pasword”,salt);// salt là mảng byte số ngẫu nhiên

5. Sinh số ngẫu nhiên: Class RandomNumberGenerator

Class RNGCryptoServiceProvider kế thừa từ Class RandomNumberGenerator.
Ví dụ:
byte[] salt = new byte[10];// tạo một mảng byte salt 10 phầntử
RNGCryptoServiceProviderrngc = new RNGCryptoServiceProvider();
rngc.GetBytes(salt);

- Câu 30: Môi trường JDK(java development kit) 1.4 trở lên
- Ngôn ngữ Java
- Khai báo thư viện trước khi sử dụng:
  `java.security,javax.crypto,javax.crypto.spec, và javax.crypto.interfaces`
- Các nhà cung cấp: Sun, SunRsaSign, SunJCE,
- Hỗ trợ:
  - Hàm băm: MessageDigest
    - Đầu tiên, đưa dữ liệu vào khởi tạo hàm băm:

      byte[] dataBytes = “This is test data”.getBytes();

    - Tiếp theo lớp MessageDigest khởi tạo đối tượng “md” bằng thuật toán SHA:

```
MessageDigest md =
MessageDigest.getInstance("SHA");
```

- Tiếp theo "md" được cập nhật với các dữ liệu đầu vào và tạo dữ liệu băm: md.update(dataBytes);

```
byte[] digest = md.digest();
```

- Chữ kí số: Signature
  - Signature dsa = Signature.getInstance("SHA1withDSA");

- Tạo các khóa bí mật: KeyGenertor

```
KeyGenertor kg= KeyGenertor. getInstance("DES");
```

Câu 27:

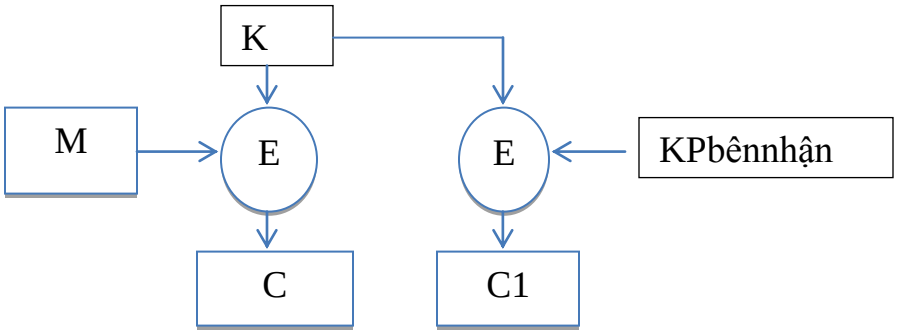
Chức năng của chuẩn SMIME.

Giao thức truyền mail trên mạng là giao thức SMTP, SMTP định dạng dữ liệu truyền trên mạng là dạng dữ liệu 7bit. Chuẩn MIME ra đời để khắc phục nhược điểm đó, MIME sẽ chuyển đổi tất cả các dạng dữ liệu không phải dạng 7bit như âm thanh hình ảnh, dữ liệu nhị phân,…thành dạng dữ liệu có thể truyền bằng SMTP. Nhưng ở đây các dữ liệu của chuẩn MIME được biểu diễn ở dạng rõ hoặc dạng có thể dễ dàng chuyển đổi thành dạng rõ. Chuẩn SMIME có vai trò:
-Kí số lên thông điệp để đảm bảo tính xác thực, toàn vẹn và tính chống chối bỏ.
-Mã hóa thông điệp nhằm đảm bảo tính bí mật.

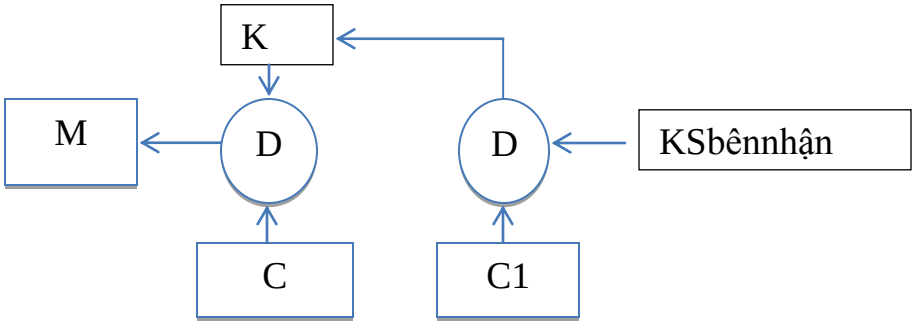
-Kí số và mã hóa để đảm bảo tính xác thực, toàn vẹn và chống chối bỏ cũng như tính bí mật của thông điệp.

1.1. Chỉ mã hóa

-Sơ đồ:



Hình 1. Tạo tin nhắn.



Hình 2. Nhận tin nhắn.

M: là thông điệp dạng MIME
M': Bản tin băm tóm lược đã được mã hóa
M'': Thông điệp M được băm bởi bên nhận
M1'': Thông điệp giải mã bản mã M' bởi bên nhận
E: Quá trình mã hóa
D: Quá trình giải mã
K: Khóa phiên ngẫu nhiên được bên gửi tạo ra.
KP: Khóa công khai
KS: Khóa bí mật
C: Bản mã thông điệp
C1: Bản mã khóa phiên
Trước đó bên gửi và bên nhận đã trao đổi chứng thư số khóa công khai và thuật toán dùng cho quá trình mã hóa và giải mã cũng như thuật toán băm (Hash).

- Quá trình tạo tin nhắn:

-Bên gửi sinh ra một số ngẫu nhiên K làm khóa phiên, dùng khóa phiên K này để mã hóa thông điệp dạng MIME bằng mật mã khóa bí mật.

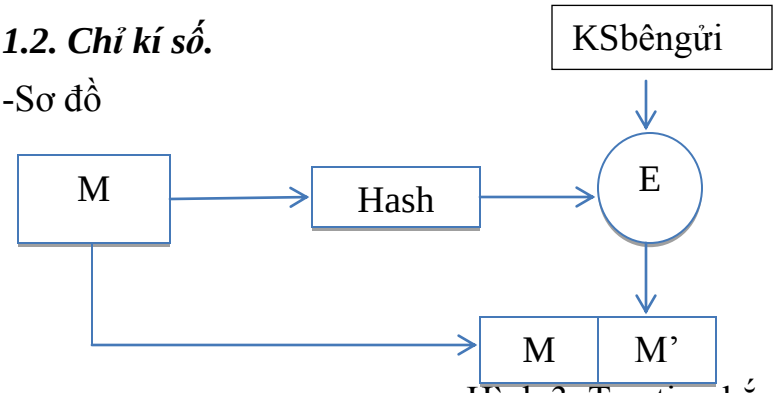
-Bên gửi lấy khóa công khai KP của bên nhận để mã hóa khóa phiên bằng mật mã khóa công khai.
-Sau đó bên gửi tiến hành đóng gói vào chuẩn MIME với tiêu đề thích hợp và phần dữ liệu này được xem như một file đính kèm với filename là smime.p7m

➢ Quá trình nhận tin nhắn:

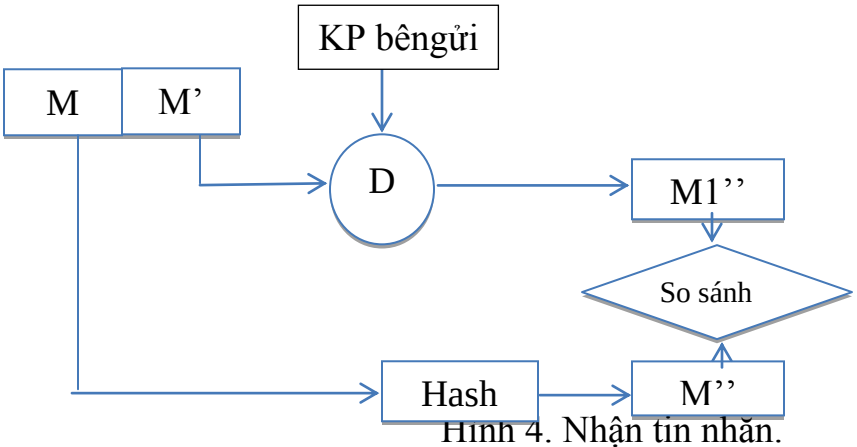
Bên nhận nhận được khối mã theo dạng MIME, bên nhận sẽ xử lí và thu được các khối mã.
-Bên nhận dùng khóa bí mật KS của mình để giải mã khối mã khóa phiên C1 bằng mật mã khóa công khai thu được K.
-Bên nhận dùng khóa K để giải mã khối mã thông điệp C thu được thông điệp mà bên gửi đã gửi cho mình.

1.2. Chỉ kí số.

-Sơ đồ



Hình 3. Tạo tin nhắn.



Hình 4. Nhận tin nhắn.

➢ Quá trình tạo tin nhắn:

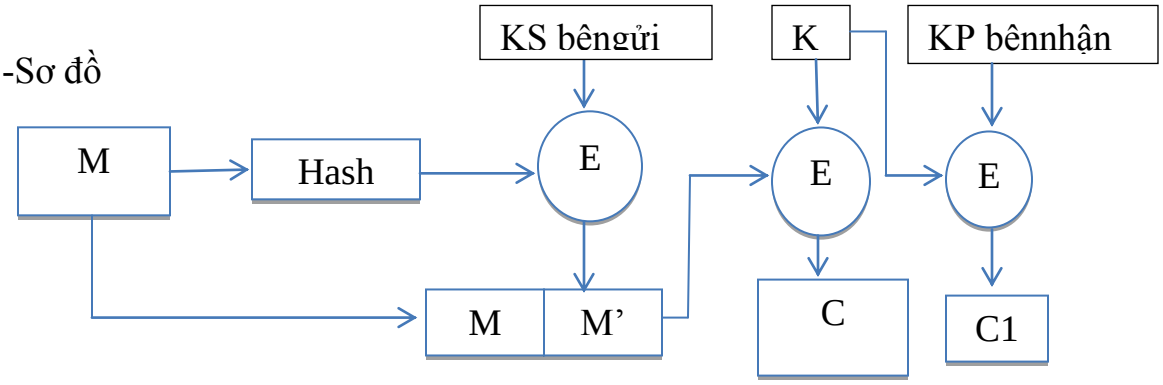
-Thông điệp được băm (Hash) thu được bản tin tóm lược.
-Bên gửi dùng khóa bí mật KS của mình để mã hóa bản tin tóm lược bằng mật mã khóa công khai thu được bản mã M'.
-Sau đó thông điệp rõ ban đầu M và bản mã M' được đóng gói vào chuẩn MIME với dạng như một tập tin đính kèm có filename là smime.p7m

➢ Quá trình nhận tin:

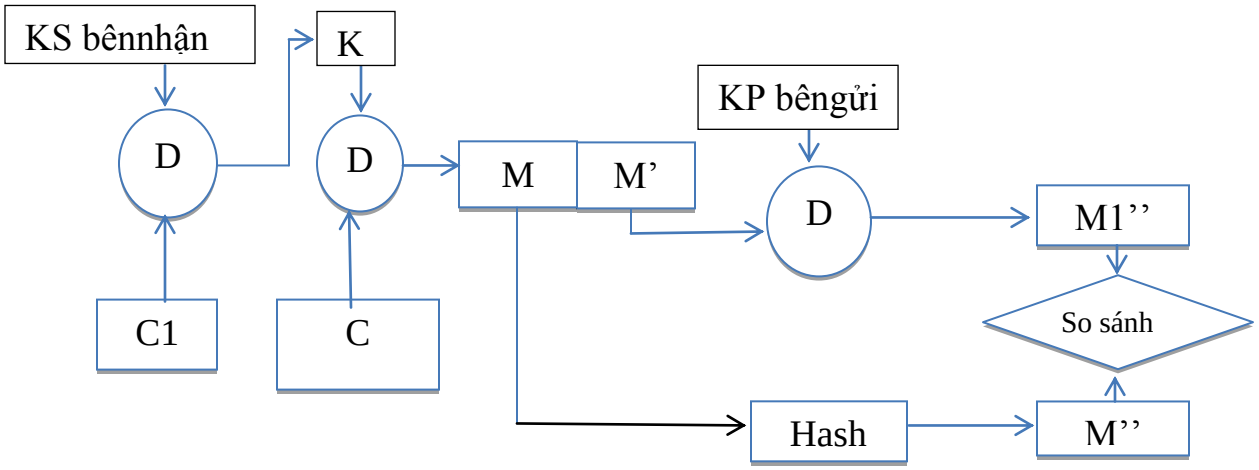
-Bên nhận nhận được phần mã hóa M’ cũng như thông điệp ở dạng rõ qua chuẩn MIME.
-Bên nhận tiến hành băm (Hash) bản tin ở dạng rõ được M’’, đồng thời bên nhận dùng khóa công khai KP của bên gửi để giải mã M’ thu được M1’’.
-Bên nhận tiến hành kiểm tra chữ kí số xem có hợp lệ hay không và hiển thị phần thông điệp.

1.3. Cả mã hóa và kí số.

-Sơ đồ



Hình 5. Tạo tin nhắn.



Hình 6. Nhận tin nhắn.

➢ Quá trình tạo tin nhắn:

-Thông điệp được băm (Hash) thu được bản tin tóm lược.
-Bên gửi dùng khóa bí mật KS của mình để mã hóa bản tin tóm lược bằng mật mã khóa công khai thu được bản mã M’. Thông điệp ban đầu M được ghép với bản mã M’.
-Bên gửi sinh ra một số ngẫu nhiên K làm khóa phiên, dùng khóa phiên K này để mã hóa M ghép với M’ bằng mật mã khóa bí mật được bản mã C.

-Bên gửi lấy khóa công khai KP của bên nhận để mã hóa khóa phiên K bằng mật mã khóa công khai được bản mã C1.
-Sau đó bên gửi tiến hành đóng gói C và C1 vào chuẩn MIME với tiêu đề thích hợp và phần dữ liệu này được xem như một file đính kèm với filename là smime.p7m

- Quá trình nhận tin:

Bên nhận nhận được khối mã theo dạng MIME, bên nhận sẽ xử lí và thu được các khối mã.
-Bên nhận dùng khóa bí mật KS của mình để giải mã khối mã khóa phiên C1 bằng mật mã khóa công khai thu được K.
-Bên nhận dùng khóa K để giải mã khối mã thông điệp C thu được M ghép với M'.
-Bên nhận tách phần thông điệp M và M' ra.
-Bên nhận tiến hành băm (Hash) thông điệp M ở dạng rõ được M'', đồng thời bên nhận dùng khóa công khai KP của bên gửi để giải mã M' thu được M1''.
-Bên nhận tiến hành kiểm tra chữ kí số xem có hợp lệ hay không và hiển thị phần thông điệp.

1.4. Vai trò mật mã.

- Mật mã khóa đối xứng:

Với khả năng xử lý nhanh mật mã khóa đối xứng được dùng trong quá trình mã hóa toàn bộ những thông điệp cần gửi tới bên nhận như thông điệp và chữ kí số, những định danh về người dùng cũng như thuật toán dùng để mã hóa,… Ở đây bên gửi và bên nhận phải thỏa thuận một khóa chung cho cả 2 bên. Khóa chung nà trong SMIME sẽ được bên gửi sinh ra và gửi cho bên nhận thông qua việc sử dụng mật mã khóa công khai. Như vậy ở đây mật mã khóa bí mật có vai trò đảm bảo tính bí mật cho thông điệp.

- Mật mã khóa công khai:

Với ưu điểm của mật mã khóa công khai là không cần trao đổi khóa chung giữa 2 bên gửi và nhận nên mật mã khóa công khai được dùng để trao đổi khóa phiên dùng chung cho bên gửi và bên nhận.
Mật mã khóa công khai có vai trò trong việc tạo chữ kí số để đảm bảo tính xác thực, toàn vẹn và chống chối bỏ đối với thông điệp.