BÀI THỰC HÀNH SỐ 3
Môn: AN NINH MẠNG
-o0o-

I. MỤC TIÊU

Mục tiêu của bài thực hành này cung cấp kiến thức về kiến trúc mật mã trên JDK (Java Development Toolkit) và kỹ năng lập trình trên các thư viện sẵn có trên kiến trúc này.
Các nội dung chính trong bài thực hành gồm 3 phần:

- Tìm hiểu kiến trúc mật mã trên JDK.
- Minh họa cách sử dụng các thư viện sẵn có trên kiến trúc nói trên.
- Viết các chương trình đơn giản dùng các thư viện sẵn có:
    - Mã hoá/giải mã tập tin với *DES*
    - Mã hoá/giải mã tập tin với *RSA*
    - Ký và xác minh chữ ký số với *SHA1withRSA*

II. CHUẨN BỊ TRƯỚC KHI THỰC HIỆN BÀI THỰC HÀNH

Trang thiết bị và các phiên bản phần mềm sử dụng:

- 01 máy tính.
- Hệ điều hành: MS Windows XP SP2, Vista, 7, 8, 10, Linux, Mac OS
- JDK: JDK phiên bản 1.6 trở lên.
- Java Development Kit (JDK) 1.6 trở lên.
- Môi trường hỗ trợ lập trình Java (Ví dụ: jGRASP) hay một môi trường lập trình tương tự như Netbean, Eclipse.

III. CÁCH THỨC VÀ HẠN CHÓT NỘP BÀI

- Tên các tập tin tương ứng với các chương trình phải viết là: *Lab03_1.java, Lab03_2.java, Lab03_3.java*. Đóng gói các tập tin này vào một tập tin nén có tên là **<MSSV>_lab03.zip** và nộp bài theo deadline được thông báo ở Bkel, không nhận bài nộp qua email hay các hình thức khác.
- Thời gian để thực hiện bài Lab là 14 ngày.

IV. NỘI DUNG THỰC HIỆN

1. Tìm hiểu kiến trúc mật mã trên JDK 1.6

Sinh viên tham khảo tài liệu *Java_Cryptography_Architecture.pdf* đính kèm hay từ URL:
http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html
Kiến trúc mật mã trên JDK 1.6 như sau:

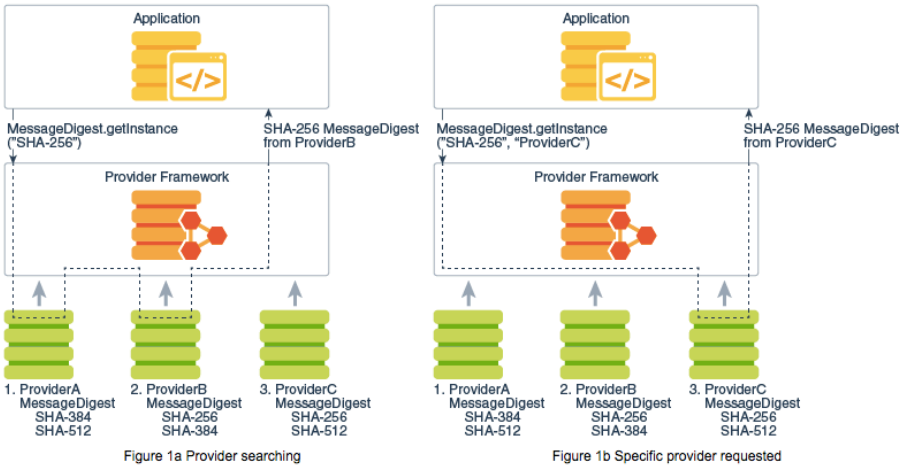

Figure 1a Provider searching

Figure 1b Specific provider requested

Các lớp cần xem xét:

- **SecureRandom**: Dùng để tạo số ngẫu nhiên.
- **MessageDigest**: Dùng để tính toán tóm tắt thông điệp dựa trên các hàm băm.
- **Signature**: Khởi tạo với các khóa và dùng chúng tạo chữ ký số và xác minh chữ ký số.
- **Cipher**: Khởi tạo với các khóa và dùng chúng cho mã hóa/giải mã dữ liệu.
- **Message Authentication Codes (MAC)**: Như MessageDigests nó cũng tạo ra giá trị băm nhưng được mã hóa vì vậy trước tiên nó được khởi tạo với khóa để bảo vệ tính toàn vẹn của thông điệp.
- **KeyGenerator**: dùng để tạo một khóa bí mật với một hệ mã đối xứng đã quy định.
- **KeyPairGenerator**: dùng để tạo một cặp khóa gồm khóa công khai và khóa riêng với một hệ mã khóa công khai đã quy định.
- **KeyFactory**: được dùng để chuyển đổi các khóa mã hóa của loại Key thành các đặc tả khóa và ngược lại.
- **SecretKeyFactory**: được dùng để chuyển đổi các khóa mã hóa của loại SecretKey thành các đặc tả khóa bí mật và ngược lại.

Các tài liệu tham khảo đầy đủ cho các gói có liên quan:

- java.security
- java.security.spec
- java.security.interfaces
- javax.crypto
- javax.crypto.spec
- javax.crypto.interfaces
- java.security.cert

2. Minh họa cách sử dụng

a. Mã hóa và giải mã một tập tin dùng DES

```
// Create a Secret key
byte key[] = "abcdEFGH".getBytes();
SecretKeySpec secretKey = new SecretKeySpec(key,"DES");

// Create Cipher object
Cipher encrypt = Cipher.getInstance("DES/ECB/PKCS5Padding");
encrypt.init(Cipher.ENCRYPT_MODE, secretKey);
```

Biên dịch và thực thi **EncryptFile.java** và **DecryptFile.java**

b. Mã hóa và giải mã một chuỗi dữ liệu dùng RSA

```
// Create a RSA key pair
KeyPairGenerator keygen = KeyPairGenerator.getInstance("RSA");
keygen.initialize(2048);
keyPair = keygen.generateKeyPair();


// Encrypt data with RSA
PublicKey key = keyPair.getPublic();
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.ENCRYPT_MODE, key);
byte[] ciphertext = cipher.doFinal(plaintext.getBytes("UTF8"));
// Decrypt data with RSA
PrivateKey key = keyPair.getPrivate();
Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.DECRYPT_MODE, key);
byte[] plaintext = cipher.doFinal(ciphertext);
```

Biên dịch và thực thi **testRSA.java**

c. Tạo mã xác thực thông điệp dùng HMAC

```
// Generate a key for the HMAC-MD5 keyed-hashing algorithm
KeyGenerator keyGen = KeyGenerator.getInstance("HmacMD5");
SecretKey key = keyGen.generateKey();

// Create a MAC object using HMAC-MD5 and initialize with key
Mac mac = Mac.getInstance(key.getAlgorithm());
mac.init(key);

// Encode a string into bytes using utf-8 and digest it
byte[] utf8 = str.getBytes("UTF8");
byte[] digest = mac.doFinal(utf8);
```

Biên dịch và thực thi **testMAC.java**

d. Mã hóa và giải mã tập tin dùng RSA

Tham khảo: https://www.novixys.com/blog/rsa-file-encryption-decryption-java/

- *Tạo khóa công khai và khóa riêng*

```
// Create RSA keys
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
kpg.initialize(2048);
KeyPair kp = kpg.generateKeyPair();

// Get RSA keys
PublicKey pub = kp.getPublic();
PrivateKey = kp.getPrivate();
```

- *Lưu trữ và khôi phục các khóa*

```
// Save RSA private key
FileOutputStream out1 = new FileOutputStream(fileBase + ".key");
out1.write(kp.getPrivate().getEncoded());
```

```
// Save RSA public key
FileOutputStream out2 = new FileOutputStream(fileBase + ".pub");
out2.write(kp.getPublic().getEncoded());

// Restore RSA public key
byte[] bytes = Files.readAllBytes(Paths.get(pubKeyFile));
X509EncodedKeySpec ks = new X509EncodedKeySpec(bytes);
KeyFactory kf = KeyFactory.getInstance("RSA");
PublicKey pub = kf.generatePublic(ks);

// Restore RSA private key
byte[] bytes = Files.readAllBytes(Paths.get(pvtKeyFile));
PKCS8EncodedKeySpec ks = new PKCS8EncodedKeySpec(bytes);
KeyFactory kf = KeyFactory.getInstance("RSA");
PrivateKey pvt = kf.generatePrivate(ks);
```

- *Mã hóa tập tin với RSA*

```
// Encrypt a file using a RSA key
PublicKey pub = ..;
Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
cipher.init(Cipher.ENCRYPT_MODE, pub);
FileInputStream in = new FileInputStream(inFile);
FileOutputStream out = new FileOutputStream(encFile);
processFile(ci, in, out);

// Method processFile()
static private void processFile(Cipher ci,InputStream
in,OutputStream out)
    throws javax.crypto.IllegalBlockSizeException,
           javax.crypto.BadPaddingException,
           java.io.IOException
{
    byte[] ibuf = new byte[1024];
    int len;
    while ((len = in.read(ibuf)) != -1) {
        byte[] obuf = ci.update(ibuf, 0, len);
        if ( obuf != null ) out.write(obuf);
    }
    byte[] obuf = ci.doFinal();
    if ( obuf != null ) out.write(obuf);
}
```

- *Giải mã một tập tin(đã mã hóa) với RSA*

```
// Encrypt a file using a RSA other key
PrivateKey pvt = ..;
Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
cipher.init(Cipher.DECRYPT_MODE, pvt);
FileInputStream in = new FileInputStream(encFile);
FileOutputStream out = new FileOutputStream(verFile);
processFile(ci, in, out);
```

- *Thực thi chương trình và kiểm chứng*
  - Biên dịch và chạy tập tin **RSA.java** (đính kèm).
  - Tạo cặp khóa.

- Tạo một tập tin test.txt để mã hóa với chương trình.
- Giải mã tập tin (đã mã hóa) với chương để kiểm chứng.

e. Ký và xác minh chữ ký với SHA1withRSA

- *Ký trên một thông điệp*

```
byte[] sign(String data, String priKeyFile) throws
InvalidKeyException, Exception{
   ..
   PrivateKey pvt = ..;
   Signature dsa = Signature.getInstance("SHA1withRSA");
   dsa.initSign(pvt);
   dsa.update(data.getBytes());
   return dsa.sign();
}
```

- *Xác minh chữ ký trên một thông điệp*

```
boolean verifySignature(byte[] data, byte[] signature, String
pubKeyFile) throws Exception {
   ..
   PublicKey pub = ..;
   Signature sig = Signature.getInstance("SHA1withRSA");
   sig.initVerify(pub);
   sig.update(data);
   return sig.verify(signature);
}
```

3. Bài tập

Câu 1. Tham khảo các ví dụ đã cho ở phần trên:

a. Viết chương trình đơn giản dùng thuật toán mã hoá DES để mã hoá và giải mã một tập tin theo các chế độ khác nhau:
- DES/ECB/PKCS5Padding
- DES/ECB/NoPadding
- DES/CBC/PKCS5Padding
- DES/CBC/NoPadding

b. Thử nghiệm chương trình đã viết để thực hiện mã hoá và giải mã các tập tin có kích thước khác nhau: 10MB, 100MB, 1GB (Download tại https://fastest.fish/test-files). Hãy cho biết cấu hình máy tính của bạn và thời gian cần thiết để mã hoá/giải mã các tập tin nói trên theo các chế độ như ở câu a là bao nhiêu?

Lưu ý:

- Chương trình cho phép nhập tên tập tin cần mã hoá/giải mã và tên tập tin chứa khoá từ màn hình console.
- Xuất kết quả ra file **output.enc** đối với chương trình mã hoá, và **output.dec** đối với chương trình giải mã.
- Thực hiện lấy giờ của hệ thống trước và sau khi thực hiện hàm mã hoá/giải mã để tính toán thời gian xử lý mã hoá/giải mã tập tin

Câu 2. Viết chương trình dùng thuật toán mã hoá RSA để mã hoá và giải mã một tập tin (chương trình cho phép nhập tên tập tin cần mã hoá/giải mã và tên tập tin chứa khoá public/private từ màn hình console). Tính toán thời gian cần thiết để mã hoá và giải mã tập tin 10MB (Download tại https://fastest.fish/test-files), so sánh với thời gian khi dùng giải thuật DES ở câu 1?

Câu 3. Dựa trên các đoạn mã tham khảo ở phần **2e)**, hãy hoàn thiện chương trình và hiện thực các phương thức sau theo sơ đồ:

a. Phương thức để ký trên một thông điệp:
`sign()`

b. Phương thức để xác minh chữ ký trên một thông điệp:
`verifySignature()`

c. Phương thức để ký trên một thông điệp và sau đó mã hoá toàn bộ (dùng giải thuật DES với giả sử khoá đối xứng đã được chia sẻ giữa bên gởi và bên nhận):
`signAndEncrypt()`

d. Phương thức để giải mã toàn bộ (dùng giải thuật DES với giả sử khoá đối xứng đã được chia sẻ giữa bên gởi và bên nhận) và xác minh chữ ký số:
`DecryptAndVerify()`

-HẾT-