BÀI THỰC HÀNH SỐ 3

Môn: MẬT MÃ & AN NINH MẠNG

-o0o-

**Họ tên: Lê Bảo Khánh**

**MSSV: 1911363**

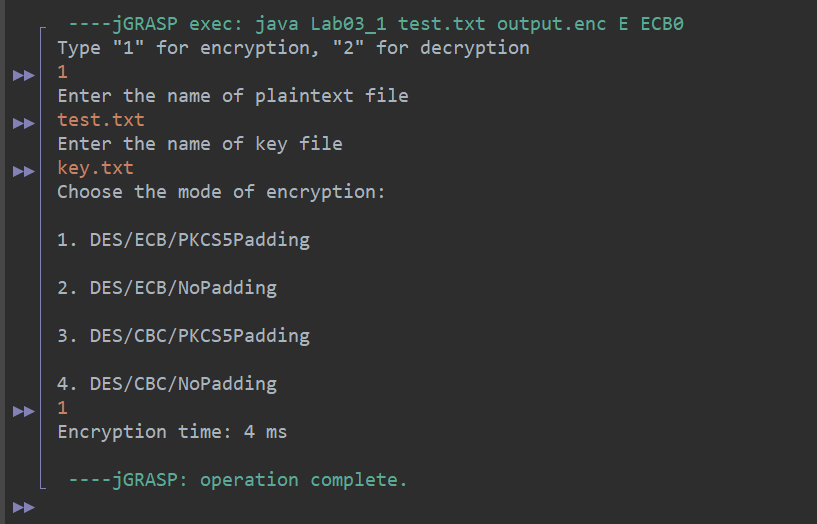
**Nhóm: L01**

**Câu 1:**

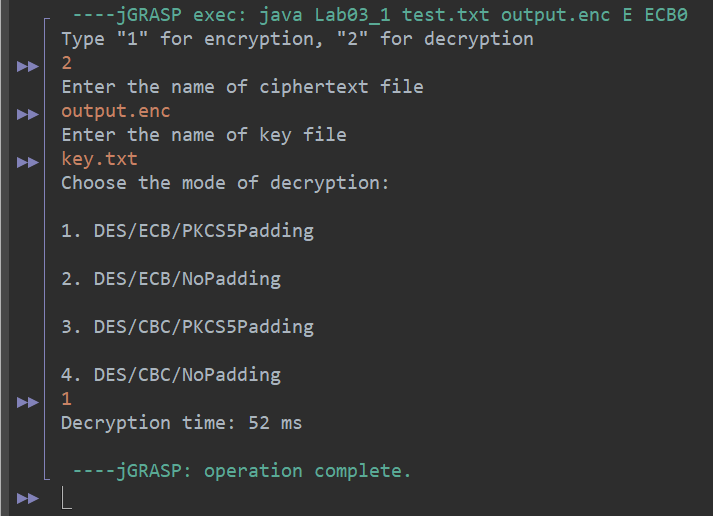
a.

Chương trình Lab03\_1.java

***Encryption:***

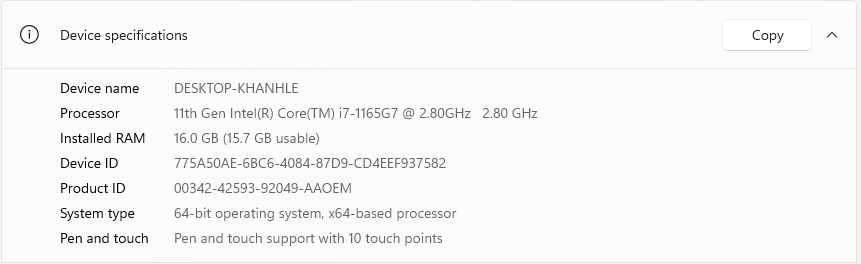


***Decryption:***



b.

Cấu hình máy tính



***Thời gian cần thiết để mã hóa:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước file | Thời gian cần thiết để mã hóa (ms) | | | |
| **DES/ECB/PKCS5Padding** | **DES/ECB/NoPadding** | **DES/CBC/PKCS5Padding** | **DES/CBC/NoPadding** |
| 10 MB | 2539 | 2487 | 2447 | 2708 |
| 100 MB | 29215 | 26069 | 27133 | 27586 |
| 1 GB | 316531 | 314913 | 319242 | 312145 |

***Thời gian cần thiết để giải mã:***

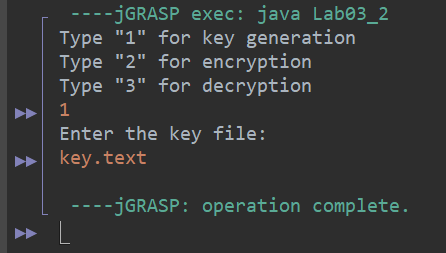
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước file | Thời gian cần thiết để giải mã (ms) | | | |
| **DES/ECB/PKCS5Padding** | **DES/ECB/NoPadding** | **DES/CBC/PKCS5Padding** | **DES/CBC/NoPadding** |
| 10 MB | 2484 | 2673 | 2594 | 2514 |
| 100 MB | 25711 | 27646 | 29142 | 27663 |
| 1 GB | 300055 | 323459 | 312924 | 32001 |

**Câu 2:**

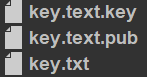
**\* Chương trình**

Chương trình Lab03\_2.java

***Key generation:***

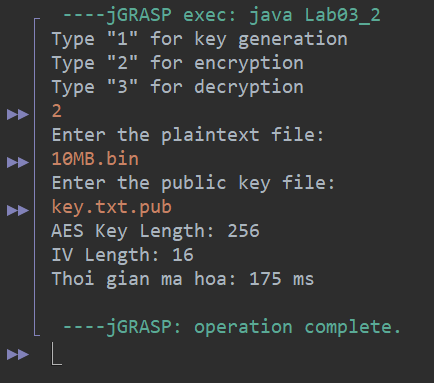


Từ key.txt, ta sinh được public key “key.txt.pub” và private key “key.txt.key”



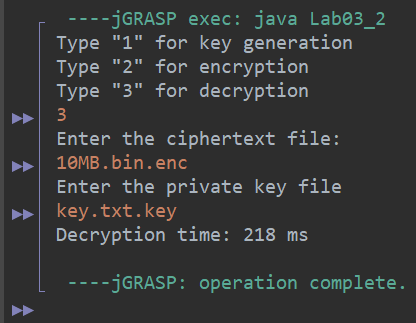
***Encryption:***

Sử dụng public key cho plaintext 10MB.bin, ta được ciphertext 10MB.bin.enc



***Decryption:***

Sử dụng private key “key.txt.key” để giải mã cho ciphertext 10MB.bin.enc, ta được plaintext 10MB.bin.enc.ver



**\* So sánh:**

So sánh với giải thuật: **DES/ECB/PKCS5Padding** thì giải thuật RSA:

* Mã hóa nhanh gấp lần
* Giải mã nhanh gấp lần

**Câu 3:**

Chương trình Lab03\_3.java

a. sign()

static public byte[] sign(String data, String pvtKeyFile)

   throws InvalidKeyException, Exception {

      byte[] bytes = Files.readAllBytes(Paths.get(pvtKeyFile));

      PKCS8EncodedKeySpec ks = new PKCS8EncodedKeySpec(bytes);

      KeyFactory kf = KeyFactory.getInstance("RSA");

      PrivateKey pvt = kf.generatePrivate(ks);

      Signature dsa = Signature.getInstance("SHA1withRSA");

      dsa.initSign(pvt);

      dsa.update(data.getBytes());

      return dsa.sign();

   }

b. verifySignature()

static public boolean verifySignature(byte[] data, byte[] signature, String pubKeyFile)

   throws Exception {

      byte[] bytes = Files.readAllBytes(Paths.get(pubKeyFile));

      X509EncodedKeySpec ks = new X509EncodedKeySpec(bytes);

      KeyFactory kf = KeyFactory.getInstance("RSA");

      PublicKey pub = kf.generatePublic(ks);

      Signature sig = Signature.getInstance("SHA1withRSA");

      sig.initVerify(pub);

      sig.update(data);

      return sig.verify(signature);

   }

c. signAndEncrypt()

static private void signAndEncrypt(String fileName, String pvtKeyFile, String DESKeyFileName)

      throws InvalidKeyException, Exception {

      String message = readFile(fileName);

      //Sign

      byte[] signStr = sign(message, pvtKeyFile);

      String str = Base64.getEncoder().encodeToString(signStr);

      System.out.println("Message: " + message);

      System.out.println("Sign: " + str);

      writeFile(fileName.substring(0, fileName.length() - 4) + "\_s.txt", str);

      encryptDES(fileName, DESKeyFileName);

      encryptDES(fileName.substring(0, fileName.length() - 4) + "\_s.txt", DESKeyFileName);

   }

d. DecryptAndVerify()

static private void DecryptAndVerify(String fileName, String signFileName, String pubKeyFile, String DESKeyFileName)

      throws Exception {

      decryptDES(fileName, DESKeyFileName);

      decryptDES(signFileName, DESKeyFileName);

      String message = readFile(fileName.substring(0, fileName.length() - 3) + "dec");

      String str = readFile(signFileName.substring(0, signFileName.length() - 3) + "dec");

      byte[] signStr = Base64.getDecoder().decode(str);

      //Verify

      boolean result = verifySignature(message.getBytes(), signStr, pubKeyFile);

      System.out.println("Result: " + result);

   }