ATIVIDADE 05

Docente: Robson Calvetti

UC: Sistemas Computacionais e Segurança – SCS

Grupo

Marinna Pereira Carneiro da Silva - RA: 824142121

Mariana Hildebrand Dantas - RA: 824118462

Christian Batista de Lima - RA: 824126605

Beatriz Silva de Jesus – RA: 824219590

Mayara Fernanda Dos Santos – RA: 824227938

Análise e Desenvolvimento de Sistemas – ADS

Trabalho realizado sobre a atividade da aula ocorrida em 24/09/24.

Classe: CryptoRSA.java

```
package exerciciocalvetti_05;


import java.io.*;
import javax.crypto.*;
import java.security.*;
import java.security.spec.*;
import java.security.cert.*;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;



public class CryptoRSA {
    private     byte[]  textoCifrado;
   private     byte[]  textoDecifrado;

   public CryptoRSA()
   {
      textoCifrado = null;
      textoDecifrado = null;
   }

   public void gerarParDeChaves(File fPub, File fPrv)
      throws IOException, NoSuchAlgorithmException, CertificateException,
           KeyStoreException, InvalidAlgorithmParameterException
   {
      final int RSAKEYSIZE = 1024;
      KeyPairGenerator kpg = KeyPairGenerator.getInstance ("RSA");
      kpg.initialize (new RSAKeyGenParameterSpec(RSAKEYSIZE, RSAKeyGenParameterSpec.F4));
      KeyPair kpr = kpg.generateKeyPair();
      PrivateKey  oPriv = kpr.getPrivate();
      PublicKey   oPub  = kpr.getPublic();
      // Gravando a chave publica em formato serializado
      ObjectOutputStream oos = new ObjectOutputStream (new FileOutputStream (fPub));
      oos.writeObject (oPub);
      oos.close();
```

```
    // Gravando a chave privada em formato serializado
    oos = new ObjectOutputStream (new FileOutputStream (fPrv));
    oos.writeObject (oPriv);
    oos.close();
}

        public void geraCifra(byte[] texto, File fPub)
        throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
              IllegalBlockSizeException, BadPaddingException,
              InvalidAlgorithmParameterException, IOException, ClassNotFoundException
    {
    ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fPub));
    PublicKey iPub = (PublicKey) ois.readObject();
    ois.close();
    Cipher rsacf = Cipher.getInstance ("RSA");
    rsacf.init (Cipher.ENCRYPT_MODE, iPub);
    textoCifrado = rsacf.doFinal (texto);
}

    public byte[]  getTextoCifrado() throws   Exception
    {
       return   textoCifrado;
    }

public void geraDecifra(byte[] texto, File fPrv)
    throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
          IllegalBlockSizeException, BadPaddingException,
          InvalidAlgorithmParameterException, IOException, ClassNotFoundException
{
    ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fPrv));
    PrivateKey iPrv = (PrivateKey) ois.readObject();
    ois.close();
    Cipher rsacf = Cipher.getInstance ("RSA");
    rsacf.init (Cipher.DECRYPT_MODE, iPrv);
    textoDecifrado = rsacf.doFinal (texto);
}

    public byte[]  getTextoDecifrado() throws   Exception
    {
```

```
        return   textoDecifrado;


    }


public void gerarParDeChaves() {
    try {
        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
        keyGen.initialize(2048); // Tamanho da chave
        KeyPair pair = keyGen.generateKeyPair();
        // Aqui você pode armazenar ou retornar as chaves como necessário
        System.out.println("Chaves geradas com sucesso!");
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
}
}
```

Classe: CryptoAES.java

```
package exerciciocalvetti_05;

import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.security.cert.*;

public class CryptoAES {

        private byte[] textoCifrado;
        private byte[] textoDecifrado;

          public CryptoAES()
          {
             textoCifrado = null;
             textoDecifrado = null;
          }

          public void geraChave(File fSim)
             throws IOException, NoSuchAlgorithmException, InvalidAlgorithmParameterException,
                  CertificateException, KeyStoreException
          {

          // Gera uma chave simétrica de 128 bits:
             KeyGenerator kg = KeyGenerator.getInstance("AES");
             kg.init(128);
             SecretKey sk = kg.generateKey();
             // Grava a chave simétrica em formato serializado
             ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fSim));
             oos.writeObject(sk);
             oos.close();
          }


          public void geraCifra(byte[] texto, File fSim)
```

```
    throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
        IllegalBlockSizeException, BadPaddingException,
        InvalidAlgorithmParameterException, IOException, ClassNotFoundException
{
    ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fSim));
    SecretKey iSim = (SecretKey) ois.readObject();
    byte[] chave = iSim.getEncoded();
    ois.close();
    Cipher aescf = Cipher.getInstance ("AES/CBC/PKCS5Padding");
    IvParameterSpec ivspec = new IvParameterSpec (new byte[16]);
    aescf.init (Cipher.ENCRYPT_MODE, new SecretKeySpec (chave, "AES"), ivspec);
    textoCifrado = aescf.doFinal (texto);
}

  public byte[] getTextoCifrado() throws Exception
  {
    return textoCifrado;
  }

  public void geraDecifra(byte[] texto, File fSim)
    throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
        IllegalBlockSizeException, BadPaddingException,
        InvalidAlgorithmParameterException, IOException, ClassNotFoundException
  {
    ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fSim));
    SecretKeySpec iSim = (SecretKeySpec) ois.readObject();
    ois.close();
    Cipher aescf = Cipher.getInstance ("AES/CBC/PKCS5Padding");
    IvParameterSpec ivspec = new IvParameterSpec (new byte[16]);
    aescf.init (Cipher.DECRYPT_MODE, iSim, ivspec);
    textoDecifrado = aescf.doFinal (texto);
}

    public byte[] getTextoDecifrado() throws Exception
  {
    return textoDecifrado;
  }
}
```

Classe: CryptoDummy.java

```
package exerciciocalvetti_05;

import java.io.*;


public class CryptoDummy {
  private byte[] textoCifrado;
  private byte[] textoDecifrado;


  public CryptoDummy() {
     textoCifrado = null;
     textoDecifrado = null;
  }


  // Gera uma chave Dummy simétrica (dk: de 0 a 100):
  public void geraChave(File fDummy) throws IOException {
     int dk = (int) (Math.random() * 101);
     // Grava a chave Dummy simétrica em formato serializado
     ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fDummy));
     oos.writeObject(dk);
     oos.close();
  }


  // Gera cifra a partir de texto e chave armazenada no arquivo
  public void geraCifra(byte[] texto, File fDummy)
     throws IOException, ClassNotFoundException
    {
     ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fDummy));
     int iDummy = (Integer) ois.readObject();
     ois.close();


     textoCifrado = texto;
     for (int i = 0; i < texto.length; i++) {
        textoCifrado[i] = (byte) (textoCifrado[i] + 1 + iDummy);
     }


}
```

```
    public byte[] getTextoCifrado() throws Exception {
  return textoCifrado;
    }

    public void geraDecifra(byte[] texto, File fDummy) throws IOException, ClassNotFoundException {
    ObjectInputStream ois = new ObjectInputStream(new FileInputStream(fDummy));
    int iDummy = (Integer) ois.readObject();
    ois.close();

       textoDecifrado = texto;
       for (int i = 0; i < texto.length; i++) {
       textoDecifrado[i] = (byte) (textoDecifrado[i] - 1 - iDummy);
  }
}

       public byte[] getTextoDecifrado() throws Exception {
          return textoDecifrado;
}

  }
```

Classe: TesteCrypto.java

```
package exerciciocalvetti_05;


import java.io.File;



public class TesteCrypto {
     public static void main(String[] args) throws Exception
   {   String  sMsgClara = "Oi, alunos do IMT!";
       String  sMsgCifrada = null;
       String  sMsgDecifrada = null;
       byte[]  bMsgClara = null;
       byte[]  bMsgCifrada = null;
       byte[]  bMsgDecifrada = null;
       // Instancia objeto da classe Impressora
       Impressora prn = new Impressora();
       // Imprime marcador de bloco
       System.out.println("----------------------------------------------------------------");
       // Imprime Texto
       System.out.println(">>> Imprimindo mensagem original...");
       System.out.println("");
       // Converte o texto String dado no equivalente byte[]
       bMsgClara = sMsgClara.getBytes("ISO-8859-1");
       // Imprime cabecalho da mensagem
       System.out.println("Mensagem Clara (Hexadecimal):");
       // Imprime o texto original em Hexadecimal
       System.out.println(prn.hexBytesToString(bMsgClara));
       System.out.println("");
       // Imprime cabecalho da mensagem
       System.out.println("Mensagem Clara (String):");
       // Imprime o texto original em String
       System.out.println(sMsgClara);
       System.out.println("");



     /*
 * Criptografia Dummy ----------------------------------------------------
```

```
*/
    // Imprime Texto
    System.out.println(">>> Cifrando com o algoritmo Dummy...");
    System.out.println("");
    // Instancia um objeto da classe CryptoDummy
    CryptoDummy cdummy = new CryptoDummy();
    // Gera a chave criptografica Dummy simetrica e nome do arquivo onde sera armazenada

      cdummy.geraChave(new File ("chave.dummy"));
  // Gera a cifra Dummy da mensagem dada, com a chave Dummy simetrica dada
  cdummy.geraCifra(bMsgClara, new File ("chave.dummy"));
  // Recebe o texto cifrado
  bMsgCifrada = cdummy.getTextoCifrado();
  // Converte o texto byte[] no equivalente String
  sMsgCifrada = (new String (bMsgCifrada, "ISO-8859-1"));
  // Imprime cabecalho da mensagem
  System.out.println("Mensagem Cifrada (Hexadecimal):");
  // Imprime o texto cifrado em Hexadecimal
  System.out.println(prn.hexBytesToString(bMsgCifrada));
  System.out.println("");
  // Imprime cabecalho da mensagem
  System.out.println("Mensagem Cifrada (String):");
  // Imprime o texto cifrado em String
  System.out.println(sMsgCifrada);
  System.out.println("");
  // Imprime texto
  System.out.println(">>> Decifrando com o algoritmo Dummy...");
  System.out.println("");

  // Gera a decifra Dummy da mensagem dada, segundo a chave Dummy simetrica gerada
  cdummy.geraDecifra(bMsgCifrada, new File ("chave.dummy"));
  // recebe o texto decifrado
  bMsgDecifrada = cdummy.getTextoDecifrado();
  // Converte o texto byte[] no equivalente String
  sMsgDecifrada = (new String (bMsgDecifrada, "ISO-8859-1"));
  // Imprime cabecalho da mensagem
  System.out.println("Mensagem Decifrada (Hexadecimal):");
  // Imprime o texto decifrado em Hexadecimal
  System.out.println(prn.hexBytesToString(bMsgDecifrada));
```

```
 System.out.println();
 // Imprime cabecalho da mensagem
 System.out.println("Mensagem Decifrada (String):");
 // Imprime o texto decifrado em String
 System.out.println(sMsgDecifrada);
 System.out.println("");

System.out.println(">>> Cifrando com o algoritmo AES...");
System.out.println("");
// Instancia um objeto da classe CryptoAES
CryptoAES   caes = new CryptoAES();
// Gera a chave criptografica AES simetrica e o nome do arquivo onde será armazenada
caes.geraChave(new File ("chave.simetrica"));
// Gera a cifra AES da mensagem dada, com a chave simetrica dada
caes.geraCifra(bMsgClara, new File ("chave.simetrica"));
// Recebe o texto cifrado
bMsgCifrada = caes.getTextoCifrado();

 // Converte o texto byte[] no equivalente String
 sMsgCifrada = (new String (bMsgCifrada, "ISO-8859-1"));
 // Imprime cabecalho da mensagem
 System.out.println("Mensagem Cifrada (Hexadecimal):");
 // Imprime o texto cifrado em Hexadecimal
 System.out.println(prn.hexBytesToString(bMsgCifrada));
 System.out.println("");
 // Imprime cabecalho da mensagem
 System.out.println("Mensagem Cifrada (String):");
 // Imprime o texto cifrado em String
 System.out.println(sMsgCifrada);
 System.out.println("");
 // Imprime texto
 System.out.println(">>> Decifrando com o algoritmo AES...");
 System.out.println("");
 // Gera a decifra AES da mensagem dada, segundo a chave simetrica gerada

 caes.geraDecifra(bMsgCifrada, new File ("chave.simetrica"));
 // recebe o texto decifrado
 bMsgDecifrada = caes.getTextoDecifrado();
 // Converte o texto byte[] no equivalente String
```

```
sMsgDecifrada = (new String (bMsgDecifrada, "ISO-8859-1"));
// Imprime cabecalho da mensagem
System.out.println("Mensagem Decifrada (Hexadecimal):");
// Imprime o texto decifrado em Hexadecimal
System.out.println(prn.hexBytesToString(bMsgDecifrada));
System.out.println();
// Imprime cabecalho da mensagem
System.out.println("Mensagem Decifrada (String):");
// Imprime o texto decifrado em String
System.out.println(sMsgDecifrada);
System.out.println("");

   System.out.println(">>> Cifrando com o algoritmo RSA...");
   System.out.println("");
   // Instancia um objeto da classe CryptoRSA
   CryptoRSA   crsa = new CryptoRSA();
   // Gera as chaves criptograficas RSA publica e privada e os arquivos onde armazenar
   crsa.gerarParDeChaves(new File ("chave.publica"), new File ("chave.privada"));
   // Gera a cifra RSA da mensagem dada, segundo a chave publica gerada
   crsa.geraCifra(bMsgClara, new File ("chave.publica"));
   // Recebe o texto cifrado
   bMsgCifrada = crsa.getTextoCifrado();
   // Converte o texto byte[] no equivalente String
   sMsgCifrada = (new String (bMsgCifrada, "ISO-8859-1"));
   // Imprime cabecalho da mensagem
   System.out.println("Mensagem Cifrada (Hexadecimal):");
   // Imprime o texto cifrado em Hexadecimal

   System.out.println(prn.hexBytesToString(bMsgCifrada));
   System.out.println("");
   // Imprime cabecalho da mensagem
   System.out.println("Mensagem Cifrada (String):");
   // Imprime o texto cifrado em String
   System.out.println(sMsgCifrada);
   System.out.println("");
   // Imprime texto
   System.out.println(">>> Decifrando com o algoritmo RSA...");
   System.out.println("");
   // Gera a decifra RSA da mensagem dada, segundo a chave privada gerada
```

```
        crsa.geraDecifra(bMsgCifrada, new File ("chave.privada"));

        // recebe o texto decifrado
        bMsgDecifrada = crsa.getTextoDecifrado();
        // Converte o texto byte[] no equivalente String
        sMsgDecifrada = (new String (bMsgDecifrada, "ISO-8859-1"));
        // Imprime cabecalho da mensagem
        System.out.println("Mensagem Decifrada (Hexadecimal):");
        // Imprime o texto decifrado em Hexadecimal
        System.out.println(prn.hexBytesToString(bMsgDecifrada));
        System.out.println();
        // Imprime cabecalho da mensagem
        System.out.println("Mensagem Decifrada (String):");
        // Imprime o texto decifrado em String
        System.out.println(sMsgDecifrada);
        System.out.println("");
    }
}
```

Classe: Impressora.java

```
package exerciciocalvetti_05;


public class Impressora {

   public String hexBytesToString(byte[] b)
   {  String   sOut = "";
      String   sBgn = "";
      String   sMdl = "";
      String   sEnd = "";
      String   sSpc = "                                        "; // 48 espacos
      for(int i = 0; i < b.length; i++)
      {  // A cada linha de 16 bytes hexadecimais faz:
         if(i%16==0) sBgn += Integer.toHexString(i&0xFFFF | 0x10000).substring(1,5) + " - ";
         // Monta a String do meio, contendo os bytes lidos
         sMdl += Integer.toHexString(b[i] & 0xFF | 0x100).substring(1,3) + " ";
         // Monta a String do final, contendo os caracteres lidos
         if(b[i] > 32 && b[i] <= 126) sEnd += (char) b[i];
         else sEnd += ".";
         // Monta linha a cada 16 caracteres lidos
         if((i % 16 == 15) || (i == b.length - 1))
         {  sOut += sBgn+sMdl+sSpc.substring(3*((i%16)+1),sSpc.length())+" - "+sEnd+"\n";
            sBgn = sMdl = sEnd = "";
         }
      }
      return   sOut;
   }
}
```

Resultado:

```
----------------------------------------------------------------
>>> Imprimindo mensagem original...

Mensagem Clara (Hexadecimal):
0000 - 4f 69 2c 20 61 6c 75 6e 6f 73 20 64 6f 20 49 4d  - Oi,.alunos.do.IM
0010 - 54 21                                            - T!


Mensagem Clara (String):
Oi, alunos do IMT!

>>> Cifrando com o algoritmo Dummy...

Mensagem Cifrada (Hexadecimal):
0000 - 53 6d 30 24 65 70 79 72 73 77 24 68 73 24 4d 51  - Sm0$epyrsw$hs$MQ
0010 - 58 25                                            - X%


Mensagem Cifrada (String):
Sm0$epyrsw$hs$MQX%

>>> Decifrando com o algoritmo Dummy...

Mensagem Decifrada (Hexadecimal):
0000 - 4f 69 2c 20 61 6c 75 6e 6f 73 20 64 6f 20 49 4d  - Oi,.alunos.do.IM
0010 - 54 21                                            - T!


Mensagem Decifrada (String):
Oi, alunos do IMT!

>>> Cifrando com o algoritmo AES...

Mensagem Cifrada (Hexadecimal):
0000 - 9c 78 95 b5 08 80 72 f5 f9 e6 1d dc 0c a1 e0 b5  - .x....r.........
0010 - b7 a7 e0 83 40 80 3a 75 df 3a 25 12 e1 b7 47 7b  - ....@.:u.:%...G{


Mensagem Cifrada (String):
xµ□rõùæÜ□¡àµ·§à@:uß:%□á·G{
```

```
Mensagem Cifrada (String):
xµ□rõùæÜ□¡àµ·§à@:uß:%□á·G{

>>> Decifrando com o algoritmo AES...

Mensagem Decifrada (Hexadecimal):
0000 - 4f 69 2c 20 61 6c 75 6e 6f 73 20 64 6f 20 49 4d  - Oi,.alunos.do.IM
0010 - 54 21                                            - T!


Mensagem Decifrada (String):
Oi, alunos do IMT!

>>> Cifrando com o algoritmo RSA...

Mensagem Cifrada (Hexadecimal):
0000 - 58 99 d3 43 9e f7 58 64 73 be 0f 55 4b 52 00 cb  - X..C..Xds..UKR..
0010 - 44 f8 83 45 be 6c c9 de e0 0c 14 64 1c e2 9f 8e  - D..E.l.....d....
0020 - 53 a7 ff dc 79 a3 9b a7 2d dd aa bd 5f 18 35 25  - S...y...-..._.5%
0030 - 52 b5 45 4a 27 4d 97 d9 92 5a 58 84 cf c3 7c 54  - R.EJ'M...ZX...|T
0040 - 0e d6 4d aa 09 a3 ec 17 33 23 2a 07 1e 4f 07 7b  - ..M.....3#*..O.{
0050 - 29 ca 42 63 a4 5e df fb cb 01 81 67 67 6d 9b fb  - ).Bc.^.....ggm..
0060 - 97 5b 86 be fd 13 62 d8 00 36 b5 b4 66 32 15 5c  - .[....b..6..f2.\
0070 - a9 e7 cb 7b ac 96 50 22 c1 de 74 d5 0d 99 ef 7e  - ...{..P"..t....~


Mensagem Cifrada (String):
XÓC÷Xds¾□UKR ËDøE¾1ÉÞà□□dâS§ÿÜy£§-Ý ª¼_□5%RµEJ'MÙZXÏÃ|T□ÖMª      £ì□3#*□O□{)ÊBc¤^ßûË□ggmû[¾ý□bØ 6µ´f2□\0çË{¬P"ÁÞtÕ
ï~

>>> Decifrando com o algoritmo RSA...

Mensagem Decifrada (Hexadecimal):
0000 - 4f 69 2c 20 61 6c 75 6e 6f 73 20 64 6f 20 49 4d  - Oi,.alunos.do.IM
0010 - 54 21                                            - T!


Mensagem Decifrada (String):
Oi, alunos do IMT!
```