ECM251 – Linguagens de Programação I

Aula 17 – L1/1 e L2/1

Engenharia da Computação – 3ª série

Criptografia em Java
(L1/1 – L2/1)

2023

Aula 17 – L1/1 e L2/1

Horário

Terça-feira: 2 aulas/semana

- L1/1 (07h40min-09h20min): *Prof. Calvetti*;
- L2/1 (07h40min-09h20min): *Prof. Igor Silveira*;

Tópico

- Criptografia em Java

Definição

- Palavra originada pela junção de duas outras, da língua grega: *kryptós* (escondido) e *gráphein* (escrita);
- É o estudo dos princípios e técnicas pelas quais uma informação pode ser transformada de sua forma original, legível, de texto claro (*plaintext*), para outra forma cifrada (*cyphertext*), supostamente ilegível;
- Somente com a posse da respectiva "**chave criptográfica secreta**", o receptor dessa informação cifrada poderá decifrá-la, ou seja, transformá-la, novamente, em sua forma original e, portanto, legível.

Definição

- Há duas técnicas, bastante difundidas, que podem ser utilizadas para cifrar e decifrar uma informação, com relativa segurança e sucesso, através de seus respectivos algoritmos criptográficos:
  - ✓ Chave Simétrica; e
  - ✓ Chaves Assimétricas.

Definição

- A Criptografia por Chave Simétrica, também denominada Criptografia de Chave Única, Criptografia de Chave Secretas, ou Sistema de Chave Simétrica, utiliza chave criptográfica relacionada para cifrar e decifrar as informações;
- A operação com chave simétrica é, normalmente, mais simples, pois seus algoritmos se utilizam de uma mesma chave criptográfica secreta para realizar ambas as operações;
- Esta chave, na prática, representa um segredo, compartilhado entre duas ou mais partes, a fim de se manter um canal confidencial para a transferência da informação;

Definição

- Utiliza-se, então, uma única chave, compartilhada por todos os interlocutores, na premissa de que esta é conhecida apenas por eles;
- Pode ser dividida em:
  - ✓ Cifras de Fluxo, ou Contínuas, que cifram os bits da mensagem, um a um; e
  - ✓ Cifras por Bloco que pegam um número determinado de bits e os cifram como uma única unidade.

Definição

- Existem muitos desses algoritmos que se utilizam de blocos de 64bits, porém, os melhores e mais seguros, no momento, utilizam blocos de 128 bits ou mais;
- Em relação aos recursos computacionais necessários, esses algoritmos geralmente os consomem em menor quantidade, quando comparados aos que se utilizam de chaves assimétricas;
- Na prática, tem-se que um algoritmo de chave simétrica, de qualidade, pode ser muitas vezes mais rápido que um algoritmo de chave assimétrica de qualidade equivalente;

Definição

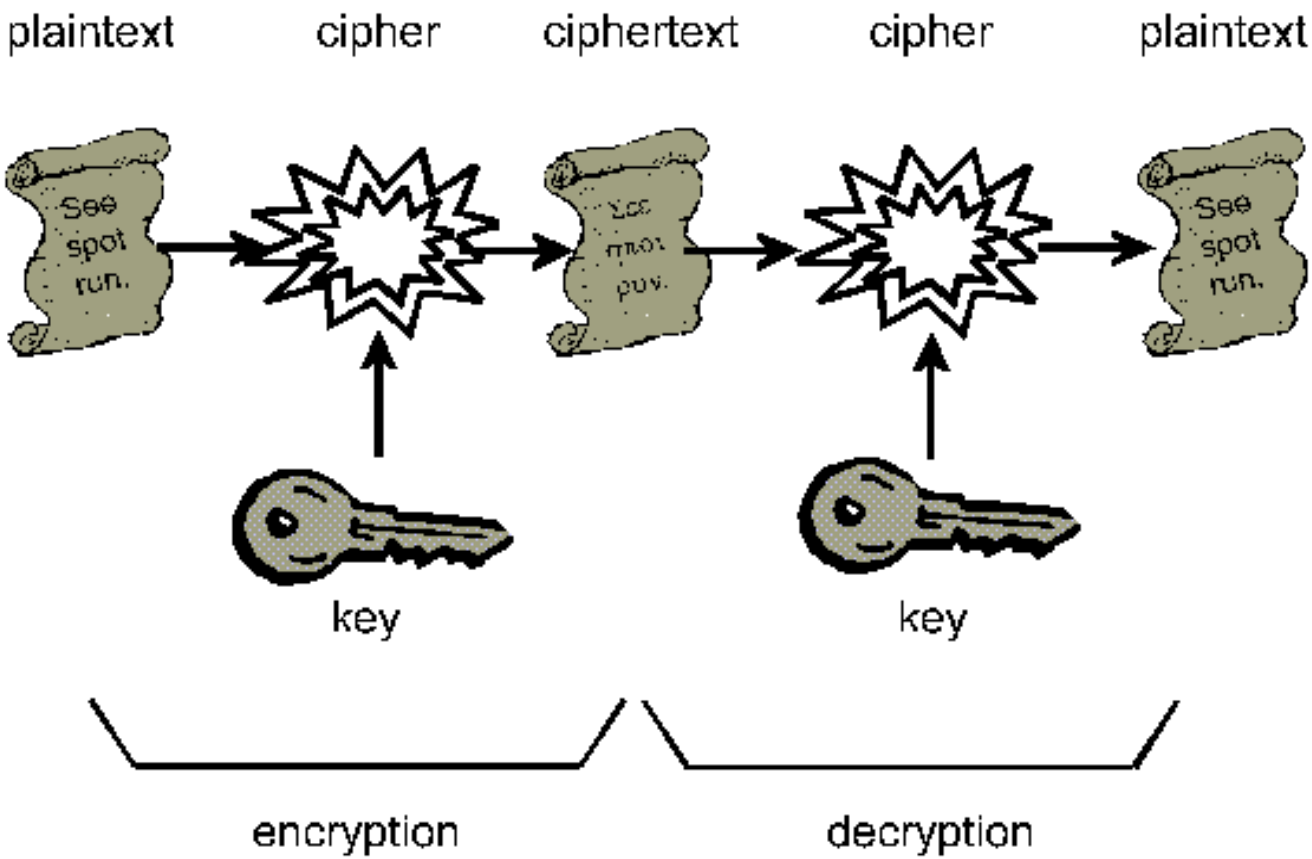
- Quanto à segurança, os algoritmos por chave simétrica são tão seguros quanto forem seguras suas próprias chaves simétricas geradas e o meio no qual são armazenadas e distribuídas para todos os seus interlocutores;
- A desvantagem dos algoritmos criptográficos por chave simétrica está na exigência de compartilhamento de sua chave secreta, ficando uma cópia dela em cada uma das extremidades da comunicação;
- Essas chaves estarão sujeitas às descobertas potenciais, por adversários criptográficos;

Definição

- Por isso, elas necessitam de mudanças frequentes e de serem mantidas seguras durante sua distribuição e serviço;
- Essa exigência de escolher, distribuir e armazenar as chaves sem erro e sem perda é conhecida como “gerenciamento de chave”;
- Alguns dos algoritmos simétricos mais populares e bem reputados: **Twofish**, **Serpent**, **AES**, **Blowfish**, **CAST5**, **RC4**, **3DES** e **IDEA**.

Criptografia por Chave Simétrica

Definição



Definição

- A Criptografia por Chaves Assimétricas, também denominada de Sistema de Chaves Pública/Privada, utiliza chaves criptográficas diferentes para cifrar e decifrar as informações;
- A operação com chaves assimétricas é, normalmente, mais complexa, pois seus algoritmos se utilizam de chaves criptográficas diferentes para realizar as respectivas operações;
- A **chave pública**, na prática, é compartilhada entre duas ou mais partes, a fim de se manter um canal confidencial para o envio da informação cifrada;

Definição

- Utiliza-se, então, uma única **chave privada**, diferente da pública, conhecida apenas pelo receptor da informação, para decifrar a informação cifrada recebida;
- Num algoritmo de criptografia assimétrica, uma mensagem cifrada com a chave pública somente pode ser decifrada pela sua chave privada correspondente;
- Com essa técnica, todos os participantes têm acesso às chaves públicas fornecidas;
- As chaves privadas são geradas localmente, por participante da comunicação e, portanto, nunca precisam ser distribuídas;

Definição

- Desde que a chave privada de um usuário permaneça protegida e secreta, a comunicação recebida estará protegida;
- A qualquer momento, podem ser alteradas as chaves pública e privada de um sistema de comunicação, bastando, então, publicar-se a nova chave pública correspondente, em substituição à antiga chave publicada;

Definição

- Esses algoritmos podem ser utilizados para:
  - ✓ **Confidencialidade**: A chave pública é usada para cifrar mensagens e, com isto, apenas o dono da chave privada pode decifrá-la, evitando assim que terceiros possam ler a mensagem;
  - ✓ **Autenticidade**: A chave privada é usada para cifrar a mensagem e, com isto, garante-se que apenas o dono da chave poderia tê-la editado.

Componentes da Criptografia por Chaves Assimétricas

Definição

- Um sistema de criptografia por **chaves assimétricas** possui os seguintes **componentes**:

  ✓ **Texto Claro**:

  - Mensagem ou dados legíveis, para serem alimentados no algoritmo como entrada;

  ✓ **Algoritmo de Cifra**:

  - Realiza várias transformações no texto claro;

Componentes da Criptografia por Chaves Assimétricas

Definição

- Um sistema de criptografia por **chaves assimétricas** possui os seguintes **componentes**:

  ✓ **Chaves Pública e Privada**:

  - Par de chaves selecionado para a realização das operações de cifra e de decifra, respectivamente, além das transformações exatas realizadas pelo algoritmo dependerem da chave pública e da chave privada fornecidas como entrada;

Componentes da Criptografia por Chaves Assimétricas

Definição

- Um sistema de criptografia por **chaves assimétricas** possui os seguintes **componentes**:

  ✓ **Texto Cifrado**:

  - Mensagem codificada, produzida como saída, que depende do texto claro e da chave criptográfica, sendo que para uma determinada mensagem, duas ou mais chaves diferentes produzirão dois ou mais textos cifrados diferentes; e

  ✓ **Algoritmo de Decifra**:

  - Aceita o texto cifrado e a chave correspondente para reproduzir o texto claro original.

Etapas da Criptografia por Chaves Assimétricas

Definição

- As etapas essenciais de operação com os sistemas de criptografia assimétricas, em relação às chaves públicas são:

1. Cada usuário gera um par de chaves, pública e privada, para serem usadas na cifra e decifra das mensagens a serem transmitidas;
2. Cada usuário coloca a chave pública gerada em um registro público ou arquivo acessível às outras partes envolvidas na comunicação;
3. Cada usuário mantém, consigo mesmo, em segurança e segredo a chave privada gerada.

Definição

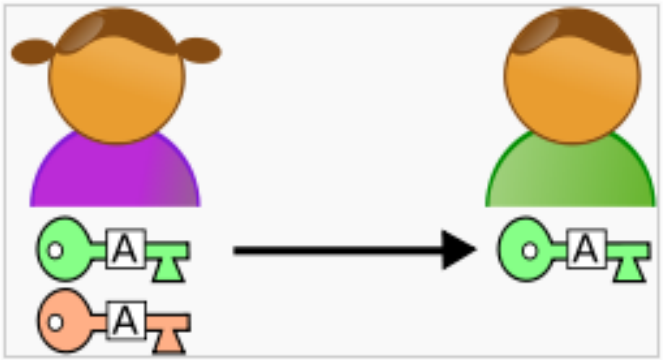
- Em relação aos recursos computacionais necessários, esses algoritmos geralmente os consomem em maior quantidade, quando comparados aos que se utilizam de chaves simétricas;
- Na prática, tem-se que um algoritmo de chave assimétrica, de qualidade, pode ser muitas vezes mais lento que um algoritmo de chave simétrica de qualidade equivalente.

Definição

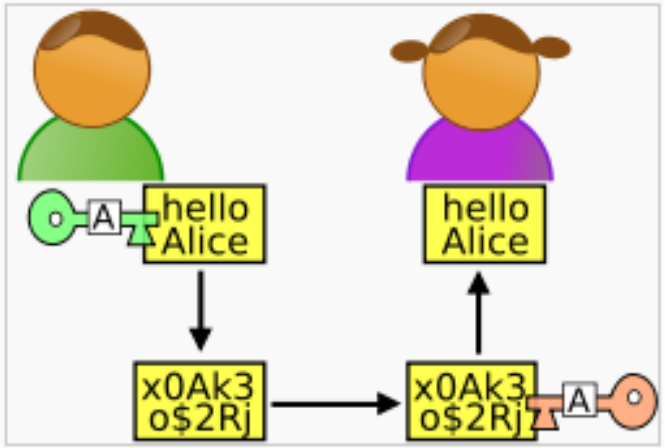
- Quanto à segurança, os algoritmos por chave assimétrica são tão seguros quanto forem seguras suas próprias chaves privadas geradas e o meio no qual são armazenadas;
- Alguns dos algoritmos assimétricos mais populares e bem reputados: **GNUPG**, **PGP**, **RSA** e **ECC**.

Criptografia por Chaves Assimétricas

Definição



Etapa 1



Etapa 2

Exemplo

- Execute as atividades descritas abaixo, exatamente na ordem na qual elas se apresentam:

  1. Copie as classes dadas em anexo para o ambiente de desenvolvimento adotado (IDE):
     - ✓ **CryptoDummy.java**;
     - ✓ **CryptoAES.java**;
     - ✓ **CryptoRSA.java**;
     - ✓ **Impressora.java**; e
     - ✓ **TesteCrypto.java**.

Exemplo

- Execute as atividades descritas abaixo, exatamente na ordem na qual elas se apresentam:

  2. Compile todas as classes dadas em anexo até que não existam erros de compilação;

  3. Verifique, na classe **TesteCrypto.java**, o conteúdo do texto claro, a ser cifrado e decifrado pelos algoritmos dados;

Exemplo

- Execute as atividades descritas abaixo, exatamente na ordem na qual elas se apresentam:

4. Execute o programa da classe **TesteCrypto.java** e verifique a operação de cifra e decifra do texto claro dado, nos seguintes algoritmos dados:

   4.1. **CryptoDummy.java**: Algoritmo de criptografia por soma de um número randômico, com grau de segurança muito baixo, que se utiliza de chave simétrica, criada e armazenada no arquivo ***chave.dummy***, gerado durante a execução do programa **TesteCrypto.java**;

Exemplo

- Execute as atividades descritas abaixo, exatamente na ordem na qual elas se apresentam:

  4.2. **CryptoAES.java**: Algoritmo de criptografia AES, com grau de segurança alto, que se utiliza de chave simétrica, criada e armazenada no arquivo ***chave.simetrica***, gerado durante a execução do programa **TesteCrypto.java**; e

Exemplo

- Execute as atividades descritas abaixo, exatamente na ordem na qual elas se apresentam:

  4.3. **CryptoRSA.java**: Algoritmo de criptografia RSA, com grau de segurança alto, que se utiliza de chaves assimétricas, criadas e armazenadas nos arquivos ***chave.publica*** e ***chave.privada***, gerados durante a execução do programa **TesteCrypto.java**;

Exemplo

- Execute as atividades descritas abaixo, exatamente na ordem na qual elas se apresentam:

  5. Renomeie os arquivos ***chave.dummy***, ***chave.simetrica***, ***chave.publica*** e ***chave.privada*** para, respectivamente, ***chave1.dummy***, ***chave1.simetrica***, ***chave1.publica*** e ***chave1.privada***;

Exemplo

- Execute as atividades descritas abaixo, exatamente na ordem na qual elas se apresentam:

6. Repita a execução do programa da classe ***Teste.Crypto.java***, e verifique se as chaves criptográficas constantes em seus respectivos arquivos são as mesmas da execução anterior (se ***chave*** é igual à ***chave1***); e

7. Verifique, para cada método de criptografia dado, se é possível decifrar uma mensagem cifrada, utilizando-se uma chave antiga equivalente (***chave1***).

Criptografia em Java

Exemplo

```
1  // Classe "CryptoDummy.java"
2  import java.io.*;
3  public   class CryptoDummy
4  {  private     byte[]   textoCifrado;
5     private     byte[]   textoDecifrado;
6     public   CryptoDummy()
7     {  textoCifrado = null;
8        textoDecifrado = null;
9     }
10    public void geraChave(File fDummy) throws IOException
11    {  // Gera uma chave Dummy simetrica (dk: 0 a 100):
12       int   dk = (int) (Math.random()*101);
13       // Grava a chave Dummy simetrica em formato serializado
14       ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fDummy));
15       oos.writeObject(dk);
16       oos.close();
17    }
18    public  void  geraCifra(byte[] texto, File fDummy)
19       throws   IOException, ClassNotFoundException
20    {  ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fDummy));
21       int iDummy = (Integer) ois.readObject();
22       ois.close();
23       textoCifrado = texto;
24       for(int i = 0; i < texto.length; i++)
25       {  textoCifrado[i] = (byte) (textoCifrado[i] + i + iDummy);
26       }
27    }
```

Criptografia em Java

Exemplo

```
28    public  byte[]   getTextoCifrado() throws   Exception
29    {  return   textoCifrado;
30    }
31    public  void  geraDecifra(byte[] texto, File fDummy)
32       throws   IOException, ClassNotFoundException
33    {  ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fDummy));
34       int iDummy = (Integer) ois.readObject();
35       ois.close();
36       textoDecifrado = texto;
37       for(int i = 0; i < texto.length; i++)
38       {  textoDecifrado[i] = (byte) (textoDecifrado[i] - i - iDummy);
39       }
40    }
41    public  byte[]   getTextoDecifrado()  throws   Exception
42    {  return   textoDecifrado;
43    }
44 }
45
```

Criptografia em Java

Exemplo

```
1  // Classe "CryptoAES.java"
2  import java.io.*;
3  import javax.crypto.*;
4  import javax.crypto.spec.*;
5  import java.security.*;
6  import java.security.cert.*;
7  public   class CryptoAES
8  {  private     byte[]   textoCifrado;
9     private     byte[]   textoDecifrado;
10    public   CryptoAES()
11    {  textoCifrado = null;
12       textoDecifrado = null;
13    }
14    public void geraChave(File fSim)
15       throws   IOException, NoSuchAlgorithmException, InvalidAlgorithmParameterException,
16                CertificateException, KeyStoreException
17    {  // Gera uma chave simetrica de 128 bits:
18       KeyGenerator kg = KeyGenerator.getInstance("AES");
19       kg.init(128);
20       SecretKey sk = kg.generateKey();
21       // Grava a chave simetrica em formato serializado
22       ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(fSim));
23       oos.writeObject(sk);
24       oos.close();
25    }
```

Criptografia em Java

Exemplo

```
26    public  void  geraCifra(byte[] texto, File fSim)
27       throws   NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
28                IllegalBlockSizeException, BadPaddingException,
29                InvalidAlgorithmParameterException, IOException, ClassNotFoundException
30    {  ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fSim));
31       SecretKey iSim = (SecretKey) ois.readObject();
32       byte[] chave = iSim.getEncoded();
33       ois.close();
34       Cipher aescf = Cipher.getInstance ("AES/CBC/PKCS5Padding");
35       IvParameterSpec ivspec = new IvParameterSpec (new byte[16]);
36       aescf.init (Cipher.ENCRYPT_MODE, new SecretKeySpec (chave, "AES"), ivspec);
37       textoCifrado = aescf.doFinal (texto);
38    }
39    public  byte[]   getTextoCifrado() throws   Exception
40    {  return   textoCifrado;
41    }
42    public  void  geraDecifra(byte[] texto, File fSim)
43       throws   NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
44                IllegalBlockSizeException, BadPaddingException,
45                InvalidAlgorithmParameterException, IOException, ClassNotFoundException
46    {  ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fSim));
47       SecretKeySpec iSim = (SecretKeySpec) ois.readObject();
48       ois.close();
49       Cipher aescf = Cipher.getInstance ("AES/CBC/PKCS5Padding");
50       IvParameterSpec ivspec = new IvParameterSpec (new byte[16]);
51       aescf.init (Cipher.DECRYPT_MODE, iSim, ivspec);
52       textoDecifrado = aescf.doFinal (texto);
53    }
```

Criptografia em Java

Exemplo

```
54    public  byte[]   getTextoDecifrado()  throws   Exception
55    {  return   textoDecifrado;
56    }
57 }
58
```

Criptografia em Java

Exemplo

```
1  // Classe "CryptoRSA.java"
2  import java.io.*;
3  import javax.crypto.*;
4  import java.security.*;
5  import java.security.spec.*;
6  import java.security.cert.*;
7  public class CryptoRSA
8  {  private     byte[]   textoCifrado;
9     private     byte[]   textoDecifrado;
10    public CryptoRSA()
11    {  textoCifrado = null;
12       textoDecifrado = null;
13    }
14    public void geraParDeChaves(File fPub, File fPvk)
15       throws   IOException, NoSuchAlgorithmException, CertificateException,
16                KeyStoreException, InvalidAlgorithmParameterException
17    {  final int RSAKEYSIZE = 1024;
18       KeyPairGenerator kpg = KeyPairGenerator.getInstance ("RSA");
19       kpg.initialize (new RSAKeyGenParameterSpec(RSAKEYSIZE, RSAKeyGenParameterSpec.F4));
20       KeyPair kpr = kpg.generateKeyPair();
21       PrivateKey   oPriv = kpr.getPrivate();
22       PublicKey    oPub  = kpr.getPublic();
23       //-- Gravando a chave publica em formato serializado
24       ObjectOutputStream oos = new ObjectOutputStream (new FileOutputStream (fPub));
25       oos.writeObject (oPub);
26       oos.close();
27       //-- Gravando a chave privada em formato serializado
28       oos = new ObjectOutputStream (new FileOutputStream (fPvk));
29       oos.writeObject (oPriv);
30       oos.close();
31    }
```

Criptografia em Java

Exemplo

```
32    public  void  geraCifra(byte[] texto, File fPub)
33       throws   NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
34                IllegalBlockSizeException, BadPaddingException,
35                InvalidAlgorithmParameterException, IOException, ClassNotFoundException
36    {  ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fPub));
37       PublicKey iPub = (PublicKey) ois.readObject();
38       ois.close();
39       Cipher rsacf = Cipher.getInstance ("RSA");
40       rsacf.init (Cipher.ENCRYPT_MODE, iPub);
41       textoCifrado = rsacf.doFinal (texto);
42    }
43    public  byte[]   getTextoCifrado() throws   Exception
44    {  return   textoCifrado;
45    }
46    public  void  geraDecifra(byte[] texto, File fPrv)
47       throws   NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException,
48                IllegalBlockSizeException, BadPaddingException,
49                InvalidAlgorithmParameterException, IOException, ClassNotFoundException
50    {  ObjectInputStream ois = new ObjectInputStream (new FileInputStream (fPrv));
51       PrivateKey iPrv = (PrivateKey) ois.readObject();
52       ois.close();
53       Cipher rsacf = Cipher.getInstance ("RSA");
54       rsacf.init (Cipher.DECRYPT_MODE, iPrv);
55       textoDecifrado = rsacf.doFinal (texto);
56    }
57    public  byte[]   getTextoDecifrado()  throws   Exception
58    {  return   textoDecifrado;
59    }
60 }
61
```

Criptografia em Java

Exemplo

```
1  // Classe "Impressora.java"
2  public class Impressora
3  {  public String hexBytesToString(byte[] b)
4     {  String   sOut = "";
5        String   sBgn = "";
6        String   sMdl = "";
7        String   sEnd = "";
8        String   sSpc = "                                                "; // 48 espacos
9        for(int i = 0; i < b.length; i++)
10       {  // A cada linha de 16 bytes hexadecimais faz:
11          if(i%16==0) sBgn += Integer.toHexString(i&0xFFFF | 0x10000).substring(1,5) + " - ";
12          // Monta a String do meio, contendo os bytes lidos
13          sMdl += Integer.toHexString(b[i] & 0xFF | 0x100).substring(1,3) + " ";
14          // Monta a String do final, contendo os caracteres lidos
15          if(b[i] >= 32 && b[i] <= 126) sEnd += (char) b[i];
16          else  sEnd += ".";
17          // Monta linha a cada 16 caracteres lidos
18          if((i % 16 == 15) || (i == b.length - 1))
19          {  sOut += sBgn+sMdl+sSpc.substring(3*((i%16)+1),sSpc.length())+" - "+sEnd+"\n";
20             sBgn = sMdl = sEnd = "";
21          }
22       }
23       return   sOut;
24    }
25 }
26
```

Criptografia em Java

Exemplo

```
1  // Classe "TesteCrypto.java"
2  import java.io.File;
3  public class TesteCrypto
4  {  public static void main(String[] args) throws Exception
5     {  String   sMsgClara = "Oi, alunos do IMT!";
6        String   sMsgCifrada = null;
7        String   sMsgDecifrada = null;
8        byte[]   bMsgClara = null;
9        byte[]   bMsgCifrada = null;
10       byte[]   bMsgDecifrada = null;
11       // Instancia objeto da classe Impressora
12       Impressora prn = new Impressora();
13       // Imprime marcador de bloco
14       System.out.println("----------------------------------------------------------------");
15       // Imprime Texto
16       System.out.println(">>> Imprimindo mensagem original...");
17       System.out.println("");
18       // Converte o texto String dado no equivalente byte[]
19       bMsgClara = sMsgClara.getBytes("ISO-8859-1");
20       // Imprime cabecalho da mensagem
21       System.out.println("Mensagem Clara (Hexadecimal):");
22       // Imprime o texto original em Hexadecimal
23       System.out.print(prn.hexBytesToString(bMsgClara));
24       System.out.println("");
25       // Imprime cabecalho da mensagem
26       System.out.println("Mensagem Clara (String):");
27       // Imprime o texto original em String
28       System.out.println(sMsgClara);
29       System.out.println("");
```

Criptografia em Java

Exemplo

```
30      /*
31       * Criptografia Dummy ------------------------------------------------------------
32       */
33      // Imprime Texto
34      System.out.println(">>> Cifrando com o algoritmo Dummy...");
35      System.out.println("");
36      // Instancia um objeto da classe CryptoDummy
37      CryptoDummy cdummy = new CryptoDummy();
38      // Gera a chave criptografica Dummy simetrica e nome do arquivo onde sera armazenada
39      cdummy.geraChave(new File ("chave.dummy"));
40      // Gera a cifra Dummy da mensagem dada, com a chave Dummy simetrica dada
41      cdummy.geraCifra(bMsgClara, new File ("chave.dummy"));
42      // Recebe o texto cifrado
43      bMsgCifrada = cdummy.getTextoCifrado();
44      // Converte o texto byte[] no equivalente String
45      sMsgCifrada = (new String (bMsgCifrada, "ISO-8859-1"));
46      // Imprime cabecalho da mensagem
47      System.out.println("Mensagem Cifrada (Hexadecimal):");
48      // Imprime o texto cifrado em Hexadecimal
49      System.out.print(prn.hexBytesToString(bMsgCifrada));
50      System.out.println("");
51      // Imprime cabecalho da mensagem
52      System.out.println("Mensagem Cifrada (String):");
53      // Imprime o texto cifrado em String
54      System.out.println(sMsgCifrada);
55      System.out.println("");
56      // Imprime texto
57      System.out.println(">>> Decifrando com o algoritmo Dummy...");
58      System.out.println("");
```

Criptografia em Java

Exemplo

```
59        // Gera a decifra Dummy da mensagem dada, segundo a chave Dummy simetrica gerada
60        cdummy.geraDecifra(bMsgCifrada, new File ("chave.dummy"));
61        // recebe o texto decifrado
62        bMsgDecifrada = cdummy.getTextoDecifrado();
63        // Converte o texto byte[] no equivalente String
64        sMsgDecifrada = (new String (bMsgDecifrada, "ISO-8859-1"));
65        // Imprime cabeÃ§alho da mensagem
66        System.out.println("Mensagem Decifrada (Hexadecimal):");
67        // Imprime o texto decifrado em Hexadecimal
68        System.out.print(prn.hexBytesToString(bMsgDecifrada));
69        System.out.println();
70        // Imprime cabecalho da mensagem
71        System.out.println("Mensagem Decifrada (String):");
72        // Imprime o texto decifrado em String
73        System.out.println(sMsgDecifrada);
74        System.out.println("");
75        /*
76         * Criptografia AES ----------------------------------------------------------------
77         */
78        // Imprime Texto
79        System.out.println(">>> Cifrando com o algoritmo AES...");
80        System.out.println("");
81        // Instancia um objeto da classe CryptoAES
82        CryptoAES   caes = new CryptoAES();
83        // Gera a Chave criptografica AES simetrica e o nome do arquivo onde será armazenada
84        caes.geraChave(new File ("chave.simetrica"));
85        // Gera a cifra AES da mensagem dada, com a chave simetrica dada
86        caes.geraCifra(bMsgClara, new File ("chave.simetrica"));
87        // Recebe o texto cifrado
88        bMsgCifrada = caes.getTextoCifrado();
```

Criptografia em Java

Exemplo

```
89      // Converte o texto byte[] no equivalente String
90      sMsgCifrada = (new String (bMsgCifrada, "ISO-8859-1"));
91      // Imprime cabecalho da mensagem
92      System.out.println("Mensagem Cifrada (Hexadecimal):");
93      // Imprime o texto cifrado em Hexadecimal
94      System.out.print(prn.hexBytesToString(bMsgCifrada));
95      System.out.println("");
96      // Imprime cabecalho da mensagem
97      System.out.println("Mensagem Cifrada (String):");
98      // Imprime o texto cifrado em String
99      System.out.println(sMsgCifrada);
100     System.out.println("");
101     // Imprime texto
102     System.out.println(">>> Decifrando com o algoritmo AES...");
103     System.out.println("");
104     // Gera a decifra AES da mensagem dada, segundo a chave simetrica gerada
105     caes.geraDecifra(bMsgCifrada, new File ("chave.simetrica"));
106     // recebe o texto decifrado
107     bMsgDecifrada = caes.getTextoDecifrado();
108     // Converte o texto byte[] no equivalente String
109     sMsgDecifrada = (new String (bMsgDecifrada, "ISO-8859-1"));
110     // Imprime cabecalho da mensagem
111     System.out.println("Mensagem Decifrada (Hexadecimal):");
112     // Imprime o texto decifrado em Hexadecimal
113     System.out.print(prn.hexBytesToString(bMsgDecifrada));
114     System.out.println();
115     // Imprime cabecalho da mensagem
116     System.out.println("Mensagem Decifrada (String):");
117     // Imprime o texto decifrado em String
118     System.out.println(sMsgDecifrada);
119     System.out.println("");
```

Criptografia em Java

Exemplo

```
120     /*
121      * Criptografia RSA ----------------------------------------------------------
122      */
123     // Imprime Texto
124     System.out.println(">>> Cifrando com o algoritmo RSA...");
125     System.out.println("");
126     // Instancia um objeto da classe CryptoRSA
127     CryptoRSA   crsa = new CryptoRSA();
128     // Gera as Chaves criptografica RSA publica e privada e os arquivos onde armazenar
129     crsa.geraParDeChaves(new File ("chave.publica"), new File ("chave.privada"));
130     // Gera a cifra RSA da mensagem dada, segundo a chave publica gerada
131     crsa.geraCifra(bMsgClara, new File ("chave.publica"));
132     // Recebe o texto cifrado
133     bMsgCifrada = crsa.getTextoCifrado();
134     // Converte o texto byte[] no equivalente String
135     sMsgCifrada = (new String (bMsgCifrada, "ISO-8859-1"));
136     // Imprime cabecalho da mensagem
137     System.out.println("Mensagem Cifrada (Hexadecimal):");
138     // Imprime o texto cifrado em Hexadecimal
139     System.out.print(prn.hexBytesToString(bMsgCifrada));
140     System.out.println("");
141     // Imprime cabecalho da mensagem
142     System.out.println("Mensagem Cifrada (String):");
143     // Imprime o texto cifrado em String
144     System.out.println(sMsgCifrada);
145     System.out.println("");
146     // Imprime texto
147     System.out.println(">>> Decifrando com o algoritmo RSA...");
148     System.out.println("");
149     // Gera a decifra RSA da mensagem dada, segundo a chave privada gerada
150     crsa.geraDecifra(bMsgCifrada, new File ("chave.privada"));
```

Criptografia em Java

Exemplo

```
151         // recebe o texto decifrado
152         bMsgDecifrada = crsa.getTextoDecifrado();
153         // Converte o texto byte[] no equivalente String
154         sMsgDecifrada = (new String (bMsgDecifrada, "ISO-8859-1"));
155         // Imprime cabecalho da mensagem
156         System.out.println("Mensagem Decifrada (Hexadecimal):");
157         // Imprime o texto decifrado em Hexadecimal
158         System.out.print(prn.hexBytesToString(bMsgDecifrada));
159         System.out.println();
160         // Imprime cabecalho da mensagem
161         System.out.println("Mensagem Decifrada (String):");
162         // Imprime o texto decifrado em String
163         System.out.println(sMsgDecifrada);
164         System.out.println("");
165     }
166 }
167
```

Exercício 1

- Altere o código fornecido para construir um aplicativo que, ao invés de receber o texto claro diretamente do código em Java (em ***hardcode***), possa fazê-lo através da leitura direta de um arquivo de texto (***.txt***), gerado pelo aplicativo "Bloco de Notas" do Windows, ou equivalente em uma IDE;

Exercício 2

- Altere o código do Exercício 1, anterior a este, para que, ao invés de apresentar os resultados em tela, apresente-os através da geração dos respectivos arquivos texto ***texto_cifrada.txt*** e ***texto_decifrado.txt***, para serem lidos através do aplicativo "Bloco de Notas" do Windows, ou equivalente em uma IDE; e

Exercício 3

- Altere o código da classe ***CryptoDummy.java***, dada em anexo, para que realize um algoritmo de criptografia diferente, de sua própria autoria;
- Teste-o e verifique seu funcionamento e grau de segurança, junto a um colega, desconhecedor do algoritmo elaborado.

Bibliografia Básica

- MILETTO, Evandro M.; BERTAGNOLLI, Silvia de Castro. Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, javascript e PHP (Tekne). Porto Alegre: Bookman, 2014. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788582601969
- WINDER, Russel; GRAHAM, Roberts. Desenvolvendo Software em Java, 3ª edição. Rio de Janeiro: LTC, 2009. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/978-85-216-1994-9
- DEITEL, Paul; DEITEL, Harvey. Java: how to program early objects. Hoboken, N. J: Pearson, c2018. 1234 p. ISBN 9780134743356.

Continua...

Bibliografia Básica (*continuação*)

- HORSTMANN, Cay S; CORNELL, Gary. Core Java. SCHAFRANSKI, Carlos (Trad.), FURMANKIEWICZ, Edson (Trad.). 8. ed. São Paulo: Pearson, 2010. v. 1. 383 p. ISBN 9788576053576.
- LIANG, Y. Daniel. Introduction to Java: programming and data structures comprehensive version. 11. ed. New York: Pearson, c2015. 1210 p. ISBN 9780134670942.
- TURINI, Rodrigo. Desbravando Java e orientação a objetos: um guia para o iniciante da linguagem. São Paulo: Casa do Código, [2017]. 222 p. (Caelum).

Bibliografia Complementar

- HORSTMANN, Cay. Conceitos de Computação com Java. Porto Alegre: Bookman, 2009. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788577804078
- MACHADO, Rodrigo P.; FRANCO, Márcia H. I.; BERTAGNOLLI, Silvia de Castro. Desenvolvimento de software III: programação de sistemas web orientada a objetos em java (Tekne). Porto Alegre: Bookman, 2016. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788582603710
- BARRY, Paul. Use a cabeça! Python. Rio de Janeiro: Alta Books, 2012. 458 p.
  ISBN 9788576087434.

Continua...

Bibliografia Complementar (continuação)

- LECHETA, Ricardo R. Web Services RESTful: aprenda a criar Web Services RESTfulem Java na nuvem do Google. São Paulo: Novatec, c2015. 431 p.
  ISBN 9788575224540.
- SILVA, Maurício Samy. JQuery: a biblioteca do programador. 3. ed. rev. e ampl. São Paulo: Novatec, 2014. 544 p.
  ISBN 9788575223871.
- SUMMERFIELD, Mark. Programação em Python 3: uma introdução completa à linguagem Phython. Rio de Janeiro: Alta Books, 2012. 506 p.
  ISBN 9788576083849.

Continua...

Bibliografia Complementar (continuação)

- YING, Bai. Practical database programming with Java. New Jersey: John Wiley & Sons, c2011. 918 p.
- ZAKAS, Nicholas C. The principles of object-oriented JavaScript. San Francisco, CA: No Starch Press, c2014. 97 p. ISBN 9781593275402.
- CALVETTI, Robson. Programação Orientada a Objetos com Java. Material de aula, São Paulo, 2020.

Aula 17 – L1/1 e L2/1

FIM

Engenharia da Computação – 3ª série

Criptografia em Java
(L1/1 – L2/1)

2023

Aula 17 – L1/2 e L2/2

Horário

Terça-feira: 2 aulas/semana

- L1/2 (09h30min-11h10min): *Prof. Calvetti*;
- L2/2 (11h20min-13h00min): *Prof. Calvetti*;

Exercícios

- Terminar, entregar e apresentar ao professor para avaliação, os exercícios propostos na aula de teoria, deste material.

Bibliografia (apoio)

- LOPES, ANITA. GARCIA, GUTO. Introdução à Programação: 500 algoritmos resolvidos. Rio de Janeiro: Elsevier, 2002.
- DEITEL, P. DEITEL, H. Java: como programar. 8 Ed. São Paulo: Prentice-Hall (Pearson), 2010;
- BARNES, David J.; KÖLLING, Michael. Programação orientada a objetos com Java: uma introdução prática usando o BlueJ . 4. ed. São Paulo: Pearson Prentice Hall, 2009.

Aula 17 – L1/2 e L2/2

FIM