ASSIGNMENT 03

COMSATS University
Islamabad
Sahiwal Campus

Usama Sarwar
FA17-BS(CS)-090-B

Mr. Khalid Mehmood
Information Security

December 12, 2020

Question No # 01

Codes of Different Ciphers

Additive Cipher:

The simplest monoalphabetic cipher is the additive cipher. This cipher is sometimes called a shift cipher and sometimes a Caesar cipher, but the term additive cipher better reveals its mathematical nature.

Code:

```
package additive.cipher;
import java.util.*;
/**
*
*/
public class AdditiveCipher {
 /**
 * @param args the command line arguments
 */
 public static void main(String[] args) {
 // TODO code application logic here
 Scanner sc = new Scanner(System.in);
 System.out.println(" Input the plaintext message here: ");
 String plaintext = sc.nextLine();
 System.out.println(" Enter the value by which each character in the plaintext message gets
shifted :
");
 int shift = sc.nextInt();
 String ciphertext = "";
 char alphabet;
 for(int i=0; i < plaintext.length();i++)
 {
 // Shift one character at a time
 alphabet = plaintext.charAt(i);

 // if alphabet lies between a and z
 if(alphabet >= 'a' && alphabet <= 'z')
 {
 // shift alphabet
 alphabet = (char) (alphabet + shift);
 // if shift alphabet greater than 'z'
 if(alphabet > 'z') {
 // reshift to starting position
 alphabet = (char) (alphabet+'a'-'z'-1);
 }
 ciphertext = ciphertext + alphabet;
 }
 // if alphabet lies between 'A'and 'Z'
 else if(alphabet >= 'A' && alphabet <= 'Z') {
alphabet = (char) (alphabet + shift);
if(alphabet > 'Z') {
```

```
 //reshift to starting position
 alphabet = (char) (alphabet+'A'-'Z'-1);
 }
 ciphertext = ciphertext + alphabet;
 }
 else {
 ciphertext = ciphertext + alphabet;
 }
 }
System.out.println(" ciphertext : " + ciphertext);
 System.out.println("The decrypted cipher is : "+ plaintext);
 }
}
```

Output

Multiple Cipher:

The multiplicative cipher is similar to additive cipher except the fact that the key bit is multiplied to the plain-text symbol during encryption. Likewise, the cipher-text is multiplied by the multiplicative inverse of key for decryption to obtain back the plain-text. The key space of multiplicative cipher is 12.

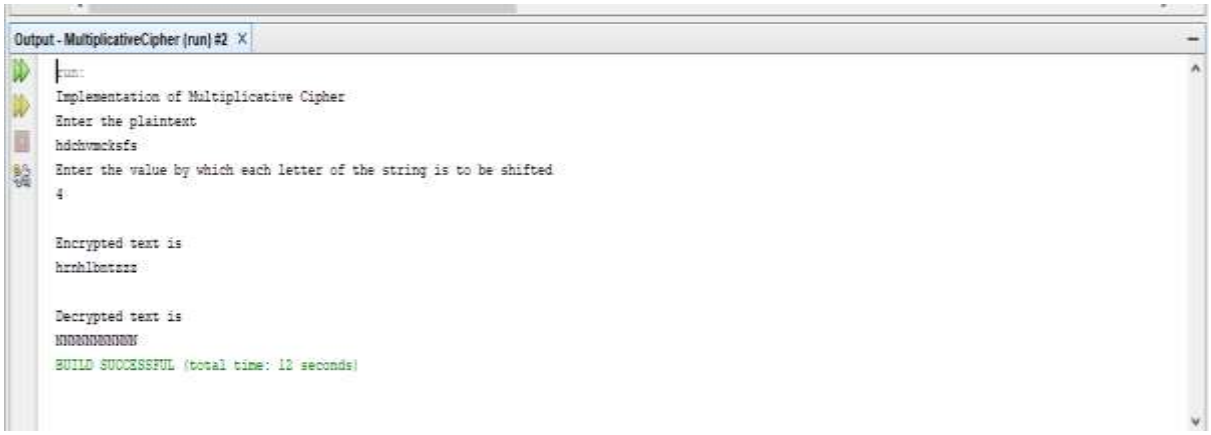
Code:

```
package multiplicativecipher;
import java.util.*;
/**
*
*/
public class MultiplicativeCipher {
 /**
 * @param args the command line arguments
       */

     public static void main(String[] args) {

        // TODO code application

        logic here Scanner sc=new

        Scanner(System.in);

   int
```

```
shift,i,n;
String str;
String
str1="";
String
str2="";
System.out.println("Implementation of Multiplicative
Cipher"); System.out.println("Enter the plaintext");
str=sc.nextLine()
;
str=str.toLowerC
ase();
n=str.length();
char
ch1[]=str.toCharArray(
); char ch3;
char ch4;
System.out.println("Enter the value by which each letter of the string is to
be shifted"); shift=sc.nextInt();

System.out.println();
System.out.println("Encrypted
text is");

for(i=0;i<n;i++)
{
if(Character.isLetter(ch1[i]))
{
ch3=(char)(((int)ch1[i]*shift-
{
str1=str1+ch1[i];
```

```
}

}
System.out.println(str1);
inverse int q=0,flag=0;

for(i=0;i<26;i++)
   {
      if(((i*26)+1)%shift==0)
      {
         q=((i*26)+1)/

         shift; break;

      }
   }

System.out.println();

System.out.println("Decrypted

text is"); char

ch2[]=str1.toCharArray();

for(i=0;i<str1.length();i++)

{
if(Character.isLetter(ch2[i]))
{


ch4=(char)(((int)ch2[i]*q-

97)%26+97); str2=str2+ch4;

}

else if(ch2[i]==' ')
{
str2=str2+ch2[i];
}
}
System.out.println(str2);
}
}
```

Output:



Affine Cipher:

The affine cipher is a type of monoalphabetic substitution cipher, where each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter.

Code:

```
package

affinecipher;

import

java.util.*;

/**
 *
*/
public class Affinecipher {
   // Key values of a

   and b static int a =

   17;

   static int b = 20;


   static String encryptMessage(char[] msg)
   {
      /// Cipher Text initially

      empty String cipher =

      "";

      for (int i = 0; i < msg.length; i++)
      {
         // Avoid space to be encrypted
         /* applying encryption formula ( a x + b ) mod m
         {here x is msg[i] and m is 26} and added

         'A' to bring it in range of ascii alphabet[
```

```
            65-90 | A-Z ] */ if (msg[i] != ' ')

            {
               cipher = cipher
                     + (char) ((((a * (msg[i] - 'A')) + b) % 26) + 'A');
            } else // else simply append space character
            {
               cipher += msg[i];
            }
         }
         return cipher;
      }

static String decryptCipher(String cipher)
      {

         String msg

         = ""; int

         a_inv = 0;

         int flag =

         0;


         //Find a^-1 (the multiplicative inverse of a
         //in the group of integers

         modulo m.) for (int i = 0; i <

         26; i++)

         {
            flag = (a * i) % 26

// Check if (a*i)%26 == 1,

            // then i will be the multiplicative
            inverse of a if (flag == 1)
            {

               a_inv = i;

            }

         }

         for (int i = 0; i < cipher.length(); i++)

         {
```

```
        /*Applying decryption formula a^-1 ( x - b ) mod m

        {here x is cipher[i] and m is 26} and added 'A'

        to bring it in range of ASCII alphabet[ 65-90 |
        A-Z ] */ if (cipher.charAt(i) != ' ')
        {

            msg = msg + (char) (((a_inv *
                  ((cipher.charAt(i) + 'A' - b)) % 26))
                  + 'A');
        }

        else //else simply append space character

        {

            msg += cipher.charAt(i);
        }

    }


    return msg;

}



/**

 * @param args the command line arguments

 */

public static void main(String[] args) {
        // TODO code application logic here


    Scanner S= new
          Scanner(System.in);
    System.out.println("Implementation of Affine-Cipher is : ");



    System.out.println("Enter the text here :
    "); String msg = S.nextLine();
    System.out.println("Actual String :" +
    msg);
    // Calling encryption function

    String cipherText =
    encryptMessage(msg.toCharArray());
    System.out.println("Encrypted Message is : " +
    cipherText);
```

```
        // Calling Decryption function
        System.out.println("Decrypted String :"
        + msg);

}
```

Output

Playfair Cipher:

The Playfair cipher or Playfair square or Wheatstone-Playfair cipher is a manual symmetric encryption technique and was the first literal diagram substitution cipher.

Code:

```
package

plafaircipher;

import java.util.*;

import

java.util.Scanner;



/**

 *

 * @author LAPTOP POINT

 */

public class Plafaircipher {

 private String KeyWord    = new
```

```
String(); private String Key          =
new String(); private char
matrix_arr[][] = new char[5][5];

public void setKey(String k)
{
   String K_adjust = new String();
   boolean flag = false;
   K_adjust = K_adjust +
   k.charAt(0); for (int i = 1; i <
   k.length(); i++)
   {
      for (int j = 0; j < K_adjust.length(); j++)
      {
         if (k.charAt(i) == K_adjust.charAt(j))
         {
            flag = true;
         }
      }
      if (flag == false)
         K_adjust = K_adjust +
      k.charAt(i); flag = false;
   }
   KeyWord = K_adjust;
}

public void KeyGen()
{
   boolean flag =
   true; char
   current;
```

```
        Key = KeyWord;
        for (int i = 0; i < 26; i++)
        {
            current = (char) (i
            + 97); if (current
            == 'j')
                continue;

            for (int j = 0; j < KeyWord.length(); j++)

if (current == KeyWord.charAt(j))
                {
                    flag =
                    false;
                    break;
                }
            }
            if (flag)
                Key = Key +
            current; flag =
            true;
        }
        System.out.println(
        Key); matrix();
    }

    private void matrix()
    {
        int counter = 0;
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 5; j++)
```

```
            {
               matrix_arr[i][j]                    =
               Key.charAt(counter);
               System.out.print(matrix_arr[i][j]
               + " "); counter++;
            }
            System.out.println();
         }
   }

      private String format(String old_text)
      {
         int i =
         0; int
         len =
         0;
         String text = new
         String(); len =
         old_text.length();
         for (int tmp = 0; tmp < len; tmp++)
         {
            if (old_text.charAt(tmp) == 'j')
            {
               text = text + 'i';
            }
            else
               text = text + old_text.charAt(tmp);
         }
         len = text.length();
         for (i = 0; i < len; i = i + 2)
         {
            if (text.charAt(i + 1) == text.charAt(i))
```

```
        {
            text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);
        }
    }
    return text;


}

private String[] Divid2Pairs(String new_string)
{
    String Original =
    format(new_string); int size =
    Original.length();
    if (size % 2 != 0)
    {
        size++;
        Original = Original + 'x';
    }
    String x[] = new
    String[size / 2]; int
    counter = 0;
    for (int i = 0; i < size / 2; i++)
    {
        x[i] = Original.substring(counter,
        counter + 2); counter = counter + 2;
    }
    return x;
}

public int[] GetDiminsions(char letter)
{
```

```
    int[] key = new
    int[2]; if (letter
    == 'j')
       letter = 'i';
    for (int i = 0; i < 5; i++)
    {
       for (int j = 0; j < 5; j++)
       {
          if (matrix_arr[i][j] == letter)
          {
             key[0] = i;
             key[1
             ] j;
             break;


    }
    return key;
}

public String encryptMessage(String Source)
{
    String src_arr[] =
    Divid2Pairs(Source); String Code =
    new String();
    char
    one;
    char
    two;
    int part1[] = new
    int[2]; int part2[]
    = new int[2];
```

```
for (int i = 0; i < src_arr.length; i++)
{
   one =
   src_arr[i].charAt(0);
   two =
   src_arr[i].charAt(1);
   part1 =
   GetDiminsions(one);
   part2 =
   GetDiminsions(two);
   if (part1[0] ==
   part2[0])
   {
      if (part1[1]
         < 4)
         part1[1]
         ++;
      else
         part1[1]
      = 0; if
      (part2[1] <
      4)
         part2[1
      ]++; else
         part2[1] = 0;
   }
   else if (part1[1] == part2[1])
   {
      if (part1[0]
         < 4)
         part1[0]
```

```
            ++;
GetDiminsions(two); if
(part1[0] == part2[0])
        {
          if
            (part1
            [1] <
            4)
            part1[
            1]++;
          else
            part1[
          1] = 0;
          if
          (part2[1
          ] < 4)
            part
          2[1]+
          +; else
            part2[1] = 0;
        }
        else if (part1[1] == part2[1])
        {
          if
            (part1
            [0] <
            4)
            part1[
            0]++;
          else
```

```
            part1[
          0] = 0;
          if
          (part2[0
          ] < 4)
             part
          2[0]+
          +; else
             part2[0] = 0;
       }
       else
       {
          int  temp  =
          part1[1];
          part1[1]   =
          part2[1];
          part2[1]   =
          temp;
       }
       Code = Code + matrix_arr[part1[0]][part1[1]]
            + matrix_arr[part2[0]][part2[1]];
    }
    return Code;
}


/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
```

```
        // TODO code application
        logic here Plafaircipher x =
        new Plafaircipher(); Scanner
        sc = new
        Scanner(System.in);

        System.out.println("Implementation of Playfair Cipher:");

        System.out.println("Enter a
        keyword:"); String keyword =
        sc.next(); x.setKey(keyword);
        x.Ke
        yGen
        ();
        Syste
        m.ou
        t
              .println("Enter word to encrypt: (Make sure length of message is
        even)"); String key_input = sc.next();
        if (key_input.length() % 2 == 0)
        {
           System.out.println("Encryption: " + x.encryptMessage(key_input));
        }
        else
        {
           System.out.println("Message length should be even");
        }

        sc.close();

}
```

Output

Autokey Cipher:

An autokey cipher is a cipher that incorporates the message into the key. The key is generated from the message in some automated fashion, sometimes by selecting certain letters from the text or, more commonly, by adding a short primer key to the front of the message.

Code:

```
package
autokey;
import
java.lang.*
; import
java.util.*;


/**
 *
public class Autokey {
    private static final String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";


    /**
     * @param args the command line arguments
     */
```

```
public static void main(String[] args) {
   // TODO code application logic here
        String msg = "INFORMATION SECURITY";
   String key = "K";

   // This if statement is all about java regular expression
   // [] for range
   // // Extra \ is used to escape one \
   // \\d acts as delimiter
   // ? once or not at all
   // . Any character (may or may not match line
   terminators) if (key.matches("[-
   +]?\\d*\\.?\\d+"))
      key = "" +
   alphabet.charAt(Integer.parseInt(key));
   String enc = autoEncryption(msg, key);
   System.out.println("Implemenation of Autokey-
   Cipher : "); System.out.println("Plaintext is : " +
   msg); System.out.println("Encrypted is : " + enc);
   System.out.println("Decrypted is : " +
   autoDecryption(enc, key));
}

public static String autoEncryption(String msg, String key)
{
   int len = msg.length();

   // generating the keystream
   String newKey = key.concat(msg);
   newKey = newKey.substring(0, newKey.length() -
   key.length()); String encryptMsg = "";
```

```
// applying encryption algorithm for (int x = 0; x <

        len; x++) {
            int first = alphabet.indexOf(msg.charAt(x));
            int second =
            alphabet.indexOf(newKey.charAt(x));
            int total = (first + second) % 26;

            encryptMsg += alphabet.charAt(total);
        }
        return encryptMsg;
    }

    public static String autoDecryption(String msg, String key)
    {
        String
        currentKey =
        key;
        String
        decryptMsg =
        "";

        // applying decryption algorithm
        for (int x = 0; x < msg.length(); x++) {
            int get1 = alphabet.indexOf(msg.charAt(x));
            int get2 =
            alphabet.indexOf(currentKey.charAt(x))
            ; int total = (get1 - get2) % 26;
            total = (total < 0) ? total + 26 :
            total; decryptMsg +=
            alphabet.charAt(total);
            currentKey +=
```

```
            alphabet.charAt(total);
        }
        return decryptMsg;
    }

}
```

Output