Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт

з лабораторної роботи № 6-7

з дисципліни «Захист інформації в комп’ютерних системах»

на тему: «Сучасні комп’ютеризовані методи шифрування та дешифрування текстових повідомлень»

Виконав: ст. гр. КІ-302

Радевич-Винницький Я.А.

Перевірив:

Муляревич О.В.

**Мета роботи:** дослідження статистичних властивостей відкритого тексту та шифрованого тексту, вивчення простих методів шифрування та дешифрування інформації та їх властивостей для сучасних шифрів, які використовуються із застосуванням комп’ютерної техніки.

**Завдання:**

Створити програму, що реалізовує шифрування і дешифрування текстів шифром Хілла.

**Виконання завдання:**

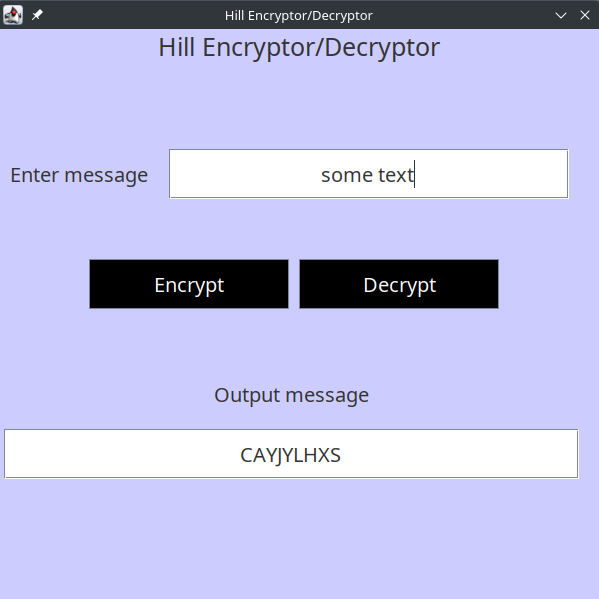
Для виконання завдання було вибрано мову Java та бібліотеку Swing для створення графічного інтерфейсу. Програмний код наведено в додатку.

Демонстрація роботи програми:



*Рис. 1 – вікно програми*

Вікно програми пропонує користувачеві ввести відкритий або закритий текст та зашифрувати або розшифрувати повідомлення.



*Рис. 2 – шифрування*



*Рис. 3 – дешифрування*

**Висновок:** у ході виконання лабораторної роботи було вивчено сучасні методи шифрування та дешифрування текстових повідомлень. За допомогою мови програмування Java та набору інструментів з бібліотеки Swing було створено програму, що реалізує алгоритм шифрування/розшифрування Хілла.

**Додаток**

Код файлу *HillCipher.java*:

Лістинг 1

|  |
| --- |
| package com.application.encryptor;  public class HillCipher {  private static final int KEY\_SIZE = 3;  private static final int[][] KEY =  {{5,6,3},  {5,3,2},  {7,5,3}};  public static String encrypt(String message) {  message = message.toUpperCase();  // check if det = 0  validateDeterminant(KEY, KEY\_SIZE);  int[][] messageVector = new int[KEY\_SIZE][1];  String cipherText = "";  int[][] cipherMatrix = new int[KEY\_SIZE][1];  int j = 0;  while (j < message.length()) {  for (int i = 0; i < KEY\_SIZE; i++) {  if (j >= message.length()) {  messageVector[i][0] = 23;  } else {  messageVector[i][0] = (message.charAt(j)) % 65;  }  j++;  }  int x, i;  for (i = 0; i < KEY\_SIZE; i++) {  cipherMatrix[i][0] = 0;  for (x = 0; x < KEY\_SIZE; x++) {  cipherMatrix[i][0] += KEY[i][x] \* messageVector[x][0];  }  cipherMatrix[i][0] = cipherMatrix[i][0] % 26;  }  for (i = 0; i < KEY\_SIZE; i++) {  cipherText += (char) (cipherMatrix[i][0] + 65);  }  }  return cipherText;  }  // Following function decrypts a message  public static String decrypt(String message) {  message = message.toUpperCase();  validateDeterminant(KEY, KEY\_SIZE);  // Calculate the inverse of the key matrix  int[][] inverseKeyMatrix = calculateInverse(KEY, KEY\_SIZE);  // solving for the required plaintext message  int[][] messageVector = new int[KEY\_SIZE][1];  StringBuilder plainText = new StringBuilder();  int[][] plainMatrix = new int[KEY\_SIZE][1];  int j = 0;  while (j < message.length()) {  for (int i = 0; i < KEY\_SIZE; i++) {  if (j >= message.length()) {  messageVector[i][0] = 23;  } else {  messageVector[i][0] = (message.charAt(j)) % 65;  }  j++;  }  int x, i;  for (i = 0; i < KEY\_SIZE; i++) {  plainMatrix[i][0] = 0;  for (x = 0; x < KEY\_SIZE; x++) {  plainMatrix[i][0] += inverseKeyMatrix[i][x] \* messageVector[x][0];  }  plainMatrix[i][0] = plainMatrix[i][0] % 26;  }  for (i = 0; i < KEY\_SIZE; i++) {  plainText.append((char) (plainMatrix[i][0] + 65));  }  }  return plainText.toString();  }  // Determinant calculator  private static int determinant(int[][] a, int n) {  int det = 0, sign = 1, p = 0, q = 0;  if (n == 1) {  det = a[0][0];  } else {  int[][] b = new int[n - 1][n - 1];  for (int x = 0; x < n; x++) {  p = 0;  q = 0;  for (int i = 1; i < n; i++) {  for (int j = 0; j < n; j++) {  if (j != x) {  b[p][q++] = a[i][j];  if (q % (n - 1) == 0) {  p++;  q = 0;  }  }  }  }  det = det + a[0][x] \* determinant(b, n - 1) \* sign;  sign = -sign;  }  }  return det;  }  // Function to calculate the inverse of the key matrix  private static int[][] calculateInverse(int[][] keyMatrix, int n) {  int det = determinant(keyMatrix, n);  int[][] adjugate = new int[n][n];  int[][] inverse = new int[n][n];  // Calculate adjugate matrix  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  int[][] minor = new int[n - 1][n - 1];  int p = 0, q = 0;  for (int x = 0; x < n; x++) {  if (x == i) continue;  q = 0;  for (int y = 0; y < n; y++) {  if (y == j) continue;  minor[p][q++] = keyMatrix[x][y];  }  p++;  }  adjugate[j][i] = (int) Math.pow(-1, i + j) \* determinant(minor, n - 1);  }  }  // Calculate inverse matrix  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  inverse[i][j] = (adjugate[i][j] % 26 + 26) % 26;  }  }  return inverse;  }  // Function to validate determinant  private static void validateDeterminant(int[][] keyMatrix, int n) {  int det = determinant(keyMatrix, n);  if (det == 0) {  System.exit(0);  }  }  } |

Код файлу *Frame.java*:

Лістинг 2

|  |
| --- |
| package com.application.ui;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JTextField;  import java.awt.Color;  import java.awt.Dimension;  import java.awt.Font;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import com.application.encryptor.HillCipher;  public class Frame extends JFrame implements ActionListener {  private static final String FRAME\_TITLE = "Hill Encryptor/Decryptor";  private static final int DIMENSION = 600;  private HillCipher hillCipher;  private JFrame frame;  private JLabel headLabel;  private JLabel inputMessageLabel;  private JLabel outputMessageLabel;  private JTextField inputTextField;  private JTextField outputTextField;  private JButton encryptionButton;  private JButton decryptionButton;  public Frame() {  headLabel = new JLabel();  adjustHeadLabelSettings(headLabel);  inputMessageLabel = new JLabel();  adjustInputMessageLabelSettings(inputMessageLabel);  inputTextField = new JTextField();  adjustInputTextFieldSettings(inputTextField);  decryptionButton = new JButton();  adjustDecryptButtonSettings(decryptionButton);  encryptionButton = new JButton();  adjustEncryptButtonSettings(encryptionButton);  outputMessageLabel = new JLabel();  adjustOutputMessageLabelSettings(outputMessageLabel);  outputTextField = new JTextField();  adjustOutputTextFieldSettings(outputTextField);  frame = new JFrame();  adjustFrameSettings(frame);  frame.add(headLabel);  frame.add(inputMessageLabel);  frame.add(inputTextField);  frame.add(encryptionButton);  frame.add(decryptionButton);  frame.add(outputMessageLabel);  frame.add(outputTextField);  }  private void adjustFrameSettings(JFrame frame) {  frame.setTitle(FRAME\_TITLE);  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frame.setResizable(false);  frame.setSize(DIMENSION,DIMENSION);  frame.getContentPane().setBackground(new Color(204,204,255));  frame.setLayout(null);  frame.setVisible(true);  }  private void adjustHeadLabelSettings(JLabel headLabel) {  headLabel.setText(FRAME\_TITLE);  headLabel.setFont(new Font("Century Gothic", Font.PLAIN, 25));  headLabel.setVerticalAlignment(JLabel.TOP);  headLabel.setHorizontalAlignment(JLabel.CENTER);  headLabel.setBounds(0,0, DIMENSION,50);  }  private void adjustInputMessageLabelSettings(JLabel inputMessageLabel) {  inputMessageLabel.setText("Enter message");  inputMessageLabel.setFont(new Font("Century Gothic", Font.PLAIN, 20));  inputMessageLabel.setHorizontalAlignment(JLabel.CENTER);  inputMessageLabel.setBounds(5, 120, 150, 50);  }  private void adjustInputTextFieldSettings(JTextField inputTextField) {  inputTextField.setPreferredSize(new Dimension(250,40));  inputTextField.setBounds(170, 120, 400, 50);  inputTextField.setFont(new Font("Century Gothic", Font.PLAIN, 20));  inputTextField.setHorizontalAlignment(JLabel.CENTER);  }  private void adjustEncryptButtonSettings(JButton encryptionButton) {  encryptionButton.setBounds(90, 230, 200, 50);  encryptionButton.setText("Encrypt");  encryptionButton.setFont(new Font("Century Gothic", Font.PLAIN, 20));  encryptionButton.setForeground(Color.WHITE);  encryptionButton.setFocusable(false);  encryptionButton.setBackground(Color.black);  encryptionButton.addActionListener(this);  }  private void adjustDecryptButtonSettings(JButton decryptionButton) {  decryptionButton.setBounds(300, 230, 200, 50);  decryptionButton.setText("Decrypt");  decryptionButton.setFont(new Font("Century Gothic", Font.PLAIN, 20));  decryptionButton.setForeground(Color.WHITE);  decryptionButton.setFocusable(false);  decryptionButton.setBackground(Color.black);  decryptionButton.addActionListener(this);  }  private void adjustOutputMessageLabelSettings(JLabel outputMessageLabel) {  outputMessageLabel.setText("Output message");  outputMessageLabel.setFont(new Font("Century Gothic", Font.PLAIN, 20));  outputMessageLabel.setHorizontalAlignment(JLabel.CENTER);  outputMessageLabel.setBounds(5, 340, 575, 50);  }  private void adjustOutputTextFieldSettings(JTextField outputTextField) {  outputTextField.setPreferredSize(new Dimension(575,50));  outputTextField.setBounds(5, 400, 575, 50);  outputTextField.setFont(new Font("Century Gothic", Font.PLAIN, 20));  outputTextField.setHorizontalAlignment(JLabel.CENTER);  }  @Override  public void actionPerformed(ActionEvent e) {  if (e.getSource().equals(encryptionButton)) {  String encryptedMessage = HillCipher.encrypt(inputTextField.getText());  outputTextField.setText(encryptedMessage);  }  if (e.getSource().equals(decryptionButton)) {  String plainText = HillCipher.decrypt(inputTextField.getText());  outputTextField.setText(plainText);  }  }  } |

Код файлу *Main.java*:

Лістинг 3

|  |
| --- |
| package com.application;  import com.application.ui.Frame;  public class Main{  public static void main(String[] args) {  Frame frame = new Frame();  }  } |