Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ



Звіт

з лабораторної роботи № 8

з дисципліни «Захист інформації в комп’ютерних системах»

на тему: «Шифрування даних за допомогою AES»

Виконав: ст. гр. КІ-302

Радевич-Винницький Я.А.

Перевірив:

Муляревич О.В.

**Мета роботи:** ознайомитись з одним із сучасних симетричних алгоритмів блочного шифрування AES, навчитися його застосовувати для різних типів даних.

**Варіант:** 24

**Завдання:**

Створити програму для шифрування файлів за допомогою AES. Налаштувати алгоритм згідно варіанту:

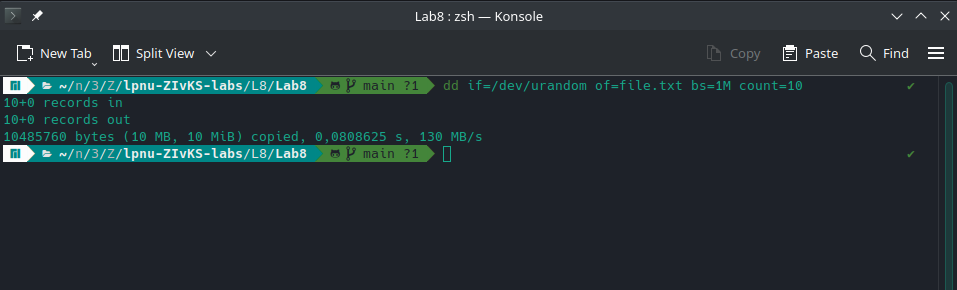
|  |  |  |  |
| --- | --- | --- | --- |
| Варіант | Режим | Довжина ключа | Ключ |
| 24 | ECB | 128 | День народження+прізвище+ім’я |

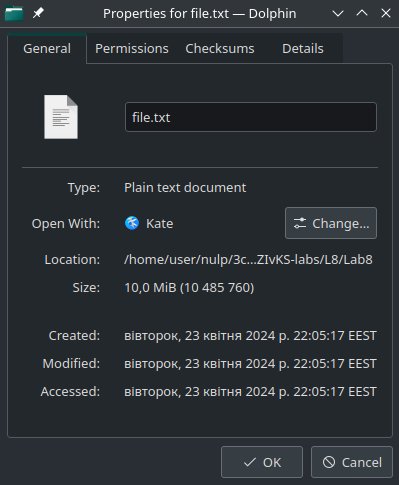
**Виконання завдання:**

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

Демонстрація роботи програми:

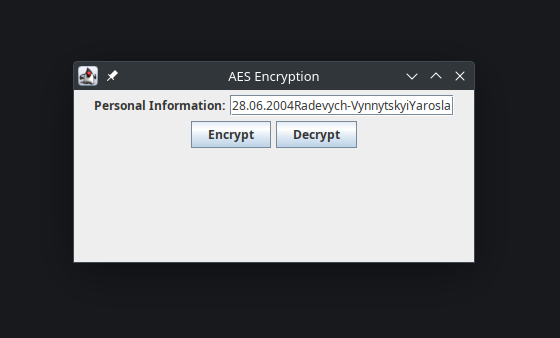
1. Згенеровано текстовий файл з розміром 10 Мб, заповнений випадковими числами:

*Рис. 1 – генерація файлу*

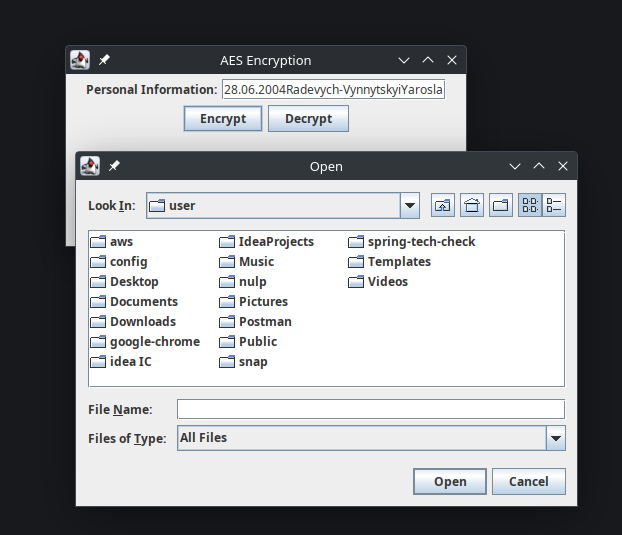


*Рис. 2 – властивості згенерованого файлу*

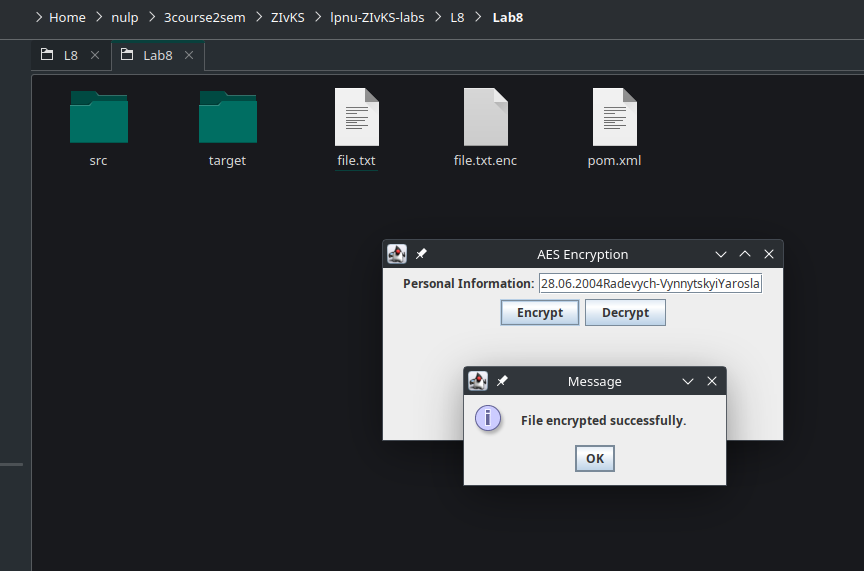
2. Запущено програму. Перше вікно містить текстове поле для введення персональної інформації, на основі якої генерується ключ.

*Рис. 3 – вікно програми*

Після того, як дані введені та натиснута кнопка «Encrypt» відкривається «File Chooser», за допомогою якого можна вибрати файл для криптування.

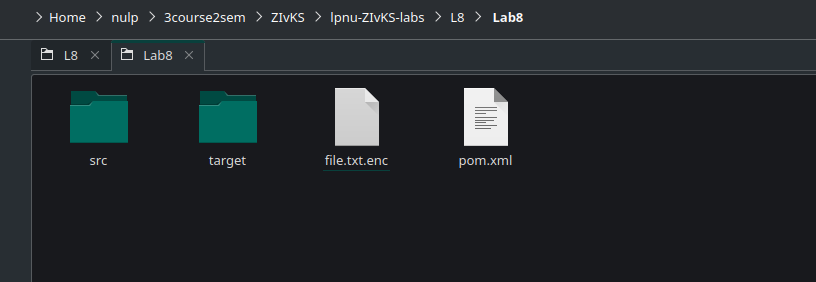
*Рис. 4 – вікно програми після активації кнопки «Encrypt»*

Після того, як було вибрано файл, програма криптує його і повідомляє про успіх. У директорії створюється зашифрована версія файлу.

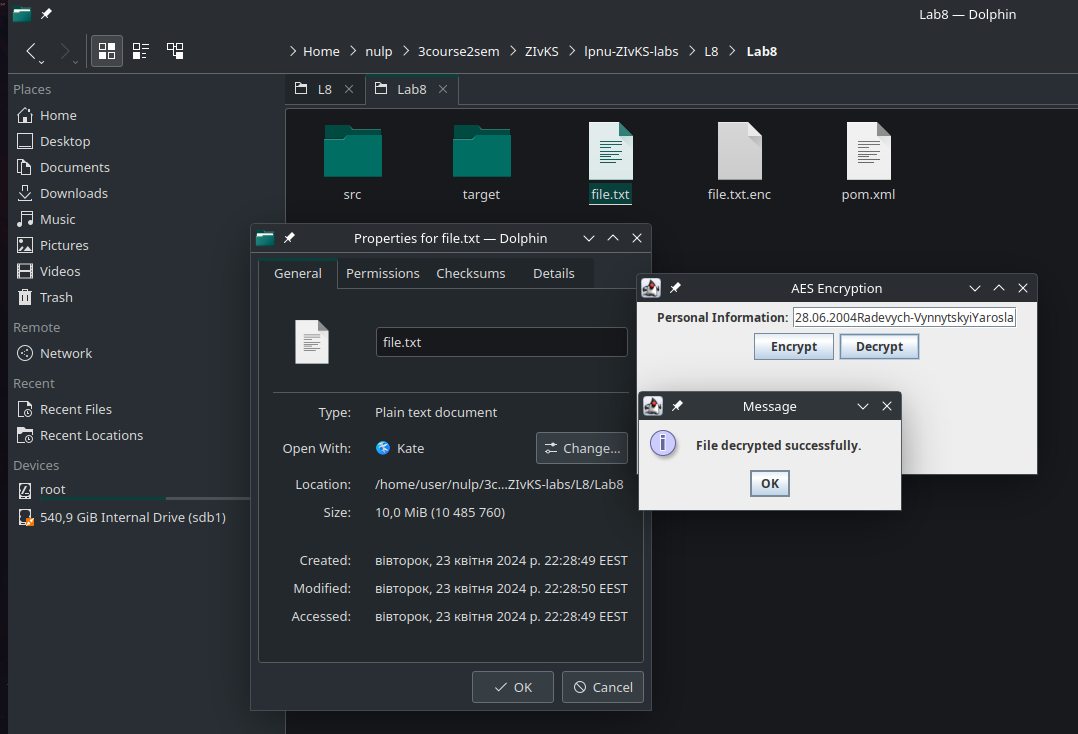


*Рис. 5 – директорія проєкту і програма після криптування*

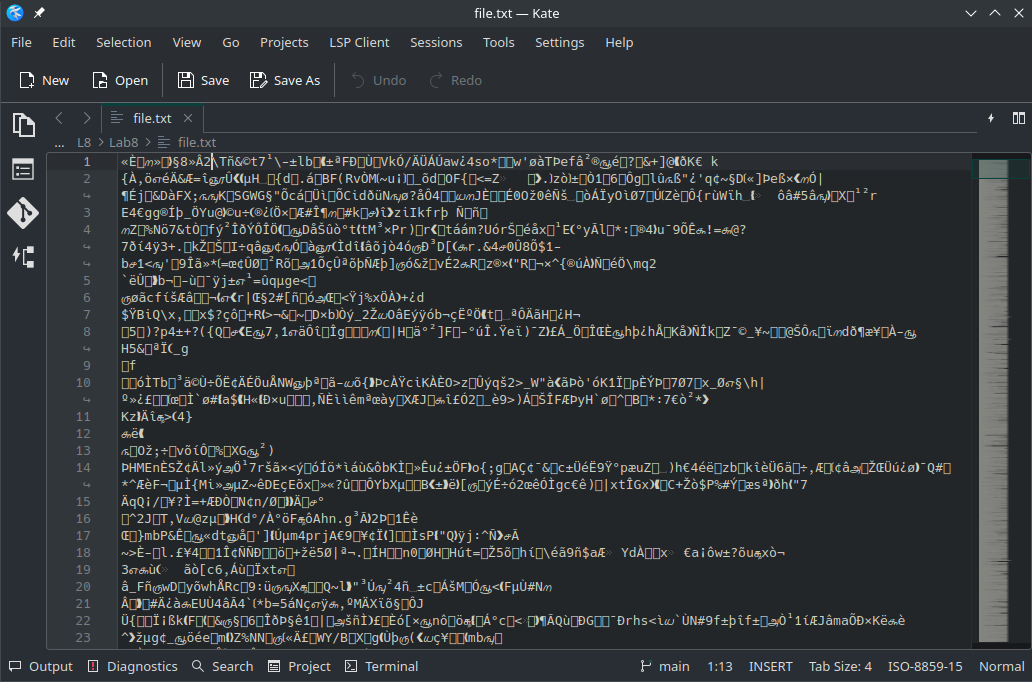
3. Протестовано дешифрування. Для цього було видалено першопочатковий відкритий файл з директорії.

*Рис. 5 – директорія проєкту після видалення відкритого файлу*

Після цього натиснуто кнопку «Decrypt», вибрано закриптований файл і розшифровано його:

*Рис. 6 – директорія проєкту після декриптування*

Вміст файлу такий самий як і на початку демонстрації:

Рис. 7 – вміст файлу file.txt

**Висновок:** у ході виконання лабораторної роботи було досліджено сучасний алгоритм блочного шифрування AES та розроблено на основі нього Java-програму для криптування/декриптування файлів.

**Додаток**

Код файлу *AESEncryption.java*:

Лістинг 1

|  |
| --- |
| package com.application.encrypt;  import javax.crypto.Cipher;  import javax.crypto.spec.SecretKeySpec;  import javax.swing.JButton;  import javax.swing.JFileChooser;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JOptionPane;  import javax.swing.JTextField;  import javax.swing.SwingUtilities;  import java.awt.FlowLayout;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.io.File;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.security.MessageDigest;  import java.security.NoSuchAlgorithmException;  public class AESEncryption extends JFrame {  private JButton encryptButton;  private JButton decryptButton;  private JTextField personalInfoField;  private JFileChooser fileChooser;  public AESEncryption() {  super("AES Encryption");  setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  setSize(400, 200);  setLayout(new FlowLayout());  personalInfoField = new JTextField(20);  add(new JLabel("Personal Information:"));  add(personalInfoField);  encryptButton = new JButton("Encrypt");  encryptButton.addActionListener(new EncryptButtonListener());  add(encryptButton);  decryptButton = new JButton("Decrypt");  decryptButton.addActionListener(new DecryptButtonListener());  add(decryptButton);  fileChooser = new JFileChooser();  fileChooser.setFileSelectionMode(JFileChooser.FILES\_ONLY);  }  private class EncryptButtonListener implements ActionListener {  public void actionPerformed(ActionEvent e) {  int returnVal = fileChooser.showOpenDialog(AESEncryption.this);  if (returnVal == JFileChooser.APPROVE\_OPTION) {  File file = fileChooser.getSelectedFile();  String personalInfo = personalInfoField.getText();  try {  SecretKeySpec secretKey = generateKeyFromPersonalInfo(personalInfo);  encryptFile(file.getAbsolutePath(), secretKey);  JOptionPane.showMessageDialog(AESEncryption.this,  "File encrypted successfully.");  } catch (Exception ex) {  ex.printStackTrace();  JOptionPane.showMessageDialog(AESEncryption.this,  "Error encrypting file: " + ex.getMessage(),  "Error", JOptionPane.ERROR\_MESSAGE);  }  }  }  }  private class DecryptButtonListener implements ActionListener {  public void actionPerformed(ActionEvent e) {  int returnVal = fileChooser.showOpenDialog(AESEncryption.this);  if (returnVal == JFileChooser.APPROVE\_OPTION) {  File file = fileChooser.getSelectedFile();  String personalInfo = personalInfoField.getText();  try {  SecretKeySpec secretKey = generateKeyFromPersonalInfo(personalInfo);  decryptFile(file.getAbsolutePath(), secretKey);  JOptionPane.showMessageDialog(AESEncryption.this,  "File decrypted successfully.");  } catch (Exception ex) {  ex.printStackTrace();  JOptionPane.showMessageDialog(AESEncryption.this,  "Error decrypting file: "  + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);  }  }  }  }  private SecretKeySpec generateKeyFromPersonalInfo(String personalInfo) throws NoSuchAlgorithmException {  MessageDigest md = MessageDigest.getInstance("SHA-256");  byte[] keyBytes = md.digest(personalInfo.getBytes());  byte[] truncatedKeyBytes = new byte[16];  System.arraycopy(keyBytes, 0, truncatedKeyBytes, 0, Math.min(keyBytes.length,  truncatedKeyBytes.length));  return new SecretKeySpec(truncatedKeyBytes, "AES");  }  private void encryptFile(String filePath, SecretKeySpec secretKey) throws Exception {  Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");  cipher.init(Cipher.ENCRYPT\_MODE, secretKey);  try (FileInputStream fis = new FileInputStream(filePath);  FileOutputStream fos = new FileOutputStream(filePath + ".enc")) {  byte[] inputBytes = new byte[4096];  int bytesRead;  while ((bytesRead = fis.read(inputBytes)) != -1) {  byte[] outputBytes = cipher.update(inputBytes, 0, bytesRead);  if (outputBytes != null) {  fos.write(outputBytes);  }  }  byte[] outputBytes = cipher.doFinal();  if (outputBytes != null) {  fos.write(outputBytes);  }  }  }  private void decryptFile(String encryptedFilePath, SecretKeySpec secretKey) throws Exception {  Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");  cipher.init(Cipher.DECRYPT\_MODE, secretKey);  try (FileInputStream fis = new FileInputStream(encryptedFilePath);  FileOutputStream fos = new FileOutputStream(encryptedFilePath.replace(".enc",  ""))) {  byte[] inputBytes = new byte[4096];  int bytesRead;  while ((bytesRead = fis.read(inputBytes)) != -1) {  byte[] outputBytes = cipher.update(inputBytes, 0, bytesRead);  if (outputBytes != null) {  fos.write(outputBytes);  }  }  byte[] outputBytes = cipher.doFinal();  if (outputBytes != null) {  fos.write(outputBytes);  }  }  }  public static void main(String[] args) {  SwingUtilities.invokeLater(new Runnable() {  public void run() {  new AESEncryption().setVisible(true);  }  });  }  } |