Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 5

По дисциплине СМЗКС

Тема: «Атаки на алгоритм RSA»

Выполнил:

Студент 4 курса

Группы ИИ-16

Журавлёв В.А.

Проверил:

Хацкевич М.В.

Брест, 2022

**Цель работы:** приобретение практических навыков кодирования/декодирования двоичных данных при использовании кода Хемминга.

**Задание:**

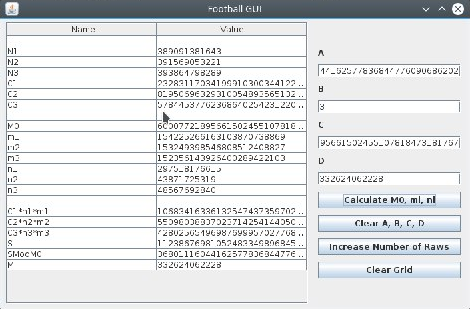
АТАКА НА АЛГОРИТМ ШИФРОВАНИЯ RSA,  
ОСНОВАННЫЙ НА КИТАЙСКОЙ ТЕОРЕМЕ ОБ ОСТАТКАХ



**Код программы:**

|  |
| --- |
| Import org.nevec.rjm.BigDecimalMath;  import java.math.BigDecimal;  import java.math.BigInteger;  import java.math.MathContext;  public class Main1 implements MainInterface{  public static final BigInteger N1 = new BigInteger("363542076673");  public static final BigInteger N2 = new BigInteger("728740902979");  public static final BigInteger N3 = new BigInteger("522993716719");  public static final BigInteger C1 = new BigInteger("246562834516");  public static final BigInteger C2 = new BigInteger("291375746601");  public static final BigInteger C3 = new BigInteger("222724269731");  private static final int e = 3;  private static BigInteger M0;  private static BigInteger m1;  private static BigInteger m2;  private static BigInteger m3;  private static BigInteger n1;  private static BigInteger n2;  private static BigInteger n3;  private static BigInteger C1n1m1;  private static BigInteger C2n2m2;  private static BigInteger C3n3m3;  private static BigInteger S;  private static BigInteger SModM0;  private static BigInteger M;  public static void main(String[] args) {  M0 = N1.multiply(N2).multiply(N3);  System.out.println("M0 = " + M0);  m1 = N2.multiply(N3);  m2 = N1.multiply(N3);  m3 = N1.multiply(N2);  System.out.println("m1 = " + m1);  System.out.println("m2 = " + m2);  System.out.println("m3 = " + m3);  n1 = m1.modInverse(N1);  n2 = m2.modInverse(N2);  n3 = m3.modInverse(N3);  System.out.println("n1 = " + n1);  System.out.println("n2 = " + n2);  System.out.println("n3 = " + n3);  C1n1m1 = C1.multiply(n1).multiply(m1);  C2n2m2 = C2.multiply(n2).multiply(m2);  C3n3m3 = C3.multiply(n3).multiply(m3);  S = C1n1m1.add(C2n2m2).add(C3n3m3);  System.out.println("S = " + S);  SModM0 = S.mod(M0);  System.out.println("SModM0 = " + SModM0);  BigDecimal Md = new BigDecimal(SModM0);  BigDecimal ed = BigDecimal.ONE.divide(BigDecimal.valueOf(e), MathContext.DECIMAL128);  System.out.println("ed = " + ed);  M = BigDecimalMath.pow(Md, ed).toBigInteger();  System.out.println("M = " + M);  }  @Override  public void start() {  M0 = N1.multiply(N2).multiply(N3);  System.out.println("M0 = " + M0);  m1 = N2.multiply(N3);  m2 = N1.multiply(N3);  m3 = N1.multiply(N2);  System.out.println("m1 = " + m1);  System.out.println("m2 = " + m2);  System.out.println("m3 = " + m3);  n1 = m1.modInverse(N1);  n2 = m2.modInverse(N2);  n3 = m3.modInverse(N3);  System.out.println("n1 = " + n1);  System.out.println("n2 = " + n2);  System.out.println("n3 = " + n3);  C1n1m1 = C1.multiply(n1).multiply(m1);  C2n2m2 = C2.multiply(n2).multiply(m2);  C3n3m3 = C3.multiply(n3).multiply(m3);  S = C1n1m1.add(C2n2m2).add(C3n3m3);  System.out.println("S = " + S);  SModM0 = S.mod(M0);  System.out.println("SModM0 = " + SModM0);  BigDecimal Md = new BigDecimal(SModM0);  BigDecimal ed = BigDecimal.ONE.divide(BigDecimal.valueOf(e), MathContext.DECIMAL128);  System.out.println("ed = " + ed);  M = BigDecimalMath.pow(Md, ed).toBigInteger();  System.out.println("M = " + M);  }  @Override  public BigInteger getN1() {  return N1;  }  @Override  public BigInteger getN2() {  return N2;  }  @Override  public BigInteger getN3() {  return N3;  }  @Override  public BigInteger getC1() {  return C1;  }  @Override  public BigInteger getC2() {  return C2;  }  @Override  public BigInteger getC3() {  return C3;  }  @Override  public BigInteger getM0() {  return M0;  }  @Override  public BigInteger getm1() {  return m1;  }  @Override  public BigInteger getm2() {  return m2;  }  @Override  public BigInteger getm3() {  return m3;  }  @Override  public BigInteger getn1() {  return n1;  }  @Override  public BigInteger getn2() {  return n2;  }  @Override  public BigInteger getn3() {  return n3;  }  @Override  public BigInteger getC1n1m1() {  return C1n1m1;  }  @Override  public BigInteger getC2n2m2() {  return C2n2m2;  }  @Override  public BigInteger getC3n3m3() {  return C3n3m3;  }  @Override  public BigInteger getS() {  return S;  }  @Override  public BigInteger getSModM0() {  return SModM0;  }  @Override  public BigInteger getM() {  return M;  }  @Override  public int getE() {  return e;  }  }  import java.awt.BorderLayout; |
| import java.awt.GridBagLayout; | import java.awt.Insets; |
| import java.awt.event.ActionEvent; | import java.awt.event.ActionListener; |
| import java.util.\*; |  |
| import javax.swing.\*; | import javax.swing.table.DefaultTableModel; |
|  | public class FootballGUI implements Runnable { |
|  | private DefaultTableModel model; |
| private JFrame frame; | private JTable table; |
| private Map<String, String> data; | private JTextField textFieldA = new JTextField(); |
| private JTextField textFieldB = new JTextField(); | private JTextField textFieldC = new JTextField(); |
| private JTextField textFieldD = new JTextField(); |  |
| static MainInterface main = new Main2(); |  |
| // Launch the application. | public static void main(String[] args) { |
| SwingUtilities.invokeLater(new FootballGUI()); | } |
|  | public FootballGUI() { |
| String[] columnNames = {" Name", " Value "}; | this.model = new DefaultTableModel(); |
|  | for (String s : columnNames) { |
| model.addColumn(s); | } |
|  | this.data = new LinkedHashMap<String, String>() {{ |
| put(" ", ""); | put("N1", main.getN1().toString()); |
| put("N2", main.getN2().toString()); | put("N3", main.getN3().toString()); |
| put("C1", main.getC1().toString()); | put("C2", main.getC2().toString()); |
| put("C3", main.getC3().toString()); | put(" ", ""); |
| }}; | } |
|  | // Create the application. |
| @Override | public void run() { |
| frame = new JFrame("Football GUI"); | frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE); |
|  | frame.add(createTablePanel(), BorderLayout.CENTER); |
| frame.add(createButtonPanel(), BorderLayout.AFTER\_LINE\_ENDS); |  |
| frame.pack(); | frame.setLocationByPlatform(true); |
| frame.setVisible(true); | } |
|  | private JPanel createTablePanel() { |
| JPanel panel = new JPanel(); |  |
| table = new JTable(model); | JScrollPane scrollPane = new JScrollPane(table); |
| panel.add(scrollPane); |  |
| return panel; | } |
|  | private JPanel createButtonPanel() { |
| JPanel panel = new JPanel(new GridBagLayout()); | panel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5)); |
|  | GridBagConstraints gbc = new GridBagConstraints(); |
| gbc.anchor = GridBagConstraints.CENTER; | gbc.fill = GridBagConstraints.HORIZONTAL; |
| gbc.gridx = 0; | gbc.gridy = 0; |
| gbc.insets = new Insets(5, 5, 5, 5); | gbc.weightx = 1.0d; |
|  | JLabel labelA = new JLabel("A"); |
| panel.add(labelA, gbc); |  |
| gbc.gridy++; | textFieldA = new JTextField(16); |
| panel.add(textFieldA, gbc); |  |
| gbc.gridy++; | JLabel labelB = new JLabel("B"); |
| panel.add(labelB, gbc); |  |
| gbc.gridy++; | textFieldB = new JTextField(16); |
| panel.add(textFieldB, gbc); |  |
| gbc.gridy++; | JLabel labelc = new JLabel("C"); |
| panel.add(labelc, gbc); |  |
| gbc.gridy++; | textFieldC = new JTextField(16); |
| panel.add(textFieldC, gbc); |  |
| gbc.gridy++; | JLabel labelD = new JLabel("D"); |
| panel.add(labelD, gbc); |  |
| gbc.gridy++; | textFieldD = new JTextField(16); |
| panel.add(textFieldD, gbc); |  |
| gbc.gridy++; | JButton displayTeams = new JButton("Calculate M0, mi, ni"); |
| displayTeams.addActionListener(new CalculateListener()); | panel.add(displayTeams, gbc); |
|  | gbc.gridy++; |
| JButton clearTextFields = new JButton("Clear A, B, C, D"); | clearTextFields.addActionListener(new ClearTextFieldsListener()); |
| panel.add(clearTextFields, gbc); |  |
| gbc.gridy++; | JButton increaseNumberOfRaws = new JButton("Increase Number of Raws"); |
| increaseNumberOfRaws.addActionListener(new IncreaseNumberOfRaws()); | panel.add(increaseNumberOfRaws, gbc); |
|  | gbc.gridy++; |
| JButton clearGrid = new JButton("Clear Grid"); | clearGrid.addActionListener(new ClearGrid()); |
| panel.add(clearGrid, gbc); |  |
| return panel; | } |
|  | public class CalculateListener implements ActionListener { |
|  | @Override |
| public void actionPerformed(ActionEvent event) { |  |
| main.start(); |  |
| int count = model.getRowCount(); | for (int i = 0; i < count; i++) { |
| model.removeRow(0); | } |
|  | data.put("M0", String.valueOf(main.getM0())); |
| data.put("m1", String.valueOf(main.getm1())); | data.put("m2", String.valueOf(main.getm2())); |
| data.put("m3", String.valueOf(main.getm3())); | data.put("n1", main.getn1().toString()); |
| data.put("n2", main.getn2().toString()); | data.put("n3", main.getn3().toString()); |
| data.put(" ", " "); | data.put("C1\*n1\*m1", main.getC1n1m1().toString()); |
| data.put("C2\*n2\*m2", main.getC2n2m2().toString()); | data.put("C3\*n3\*m3", main.getC3n3m3().toString()); |
| data.put("S", main.getS().toString()); | data.put("SModM0", main.getSModM0().toString()); |
| data.put("M", main.getM().toString()); |  |
| textFieldA.setText(main.getSModM0().toString()); | textFieldB.setText(String.valueOf(main.getE())); |
| textFieldC.setText(main.getM0().toString()); | textFieldD.setText(main.getM().toString()); |
|  | for (Map.Entry<String, String> entry : data.entrySet()) { |
| model.addRow(new Object[]{entry.getKey(), entry.getValue()}); | } |
| } | } |
|  | public class ClearGrid implements ActionListener { |
|  | @Override |
| public void actionPerformed(ActionEvent event) { | model.setNumRows(0); |
| } | } |
|  | public class IncreaseNumberOfRaws implements ActionListener { |
|  | @Override |
| public void actionPerformed(ActionEvent event) { | model.addRow(new Object[]{"", ""}); |
| } | } |
|  | public class ClearTextFieldsListener implements ActionListener { |
|  | @Override |
| public void actionPerformed(ActionEvent event) { |  |
| textFieldA.setText(""); | textFieldB.setText(""); |
| textFieldC.setText(""); | textFieldD.setText(""); |
|  | } |
| } |  |
| } |  |
| import java.awt.BorderLayout; | import java.awt.GridBagConstraints; |
| import java.awt.GridBagLayout; | import java.awt.Insets; |
| import java.awt.event.ActionEvent; | import java.awt.event.ActionListener; |
| import java.util.\*; |  |
| import javax.swing.\*; | import javax.swing.table.DefaultTableModel; |
|  | public class FootballGUI implements Runnable { |
|  | private DefaultTableModel model; |
| private JFrame frame; | private JTable table; |
| private Map<String, String> data; | private JTextField textFieldA = new JTextField(); |
| private JTextField textFieldB = new JTextField(); | private JTextField textFieldC = new JTextField(); |
| private JTextField textFieldD = new JTextField(); |  |
| static MainInterface main = new Main2(); |  |
| // Launch the application. | public static void main(String[] args) { |
| SwingUtilities.invokeLater(new FootballGUI()); | } |
|  | public FootballGUI() { |
| String[] columnNames = {" Name", " Value "}; | this.model = new DefaultTableModel(); |
|  | for (String s : columnNames) { |
| model.addColumn(s); | } |
|  | this.data = new LinkedHashMap<String, String>() {{ |
| put(" ", ""); | put("N1", main.getN1().toString()); |
| put("N2", main.getN2().toString()); | put("N3", main.getN3().toString()); |
| put("C1", main.getC1().toString()); | put("C2", main.getC2().toString()); |
| put("C3", main.getC3().toString()); | put(" ", ""); |
| }}; | } |
|  | // Create the application. |
| @Override | public void run() { |
| frame = new JFrame("Football GUI"); | frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE); |
|  | frame.add(createTablePanel(), BorderLayout.CENTER); |
| frame.add(createButtonPanel(), BorderLayout.AFTER\_LINE\_ENDS); |  |
| frame.pack(); | frame.setLocationByPlatform(true); |
| frame.setVisible(true); | } |
|  | private JPanel createTablePanel() { |
| JPanel panel = new JPanel(); |  |
| table = new JTable(model); | JScrollPane scrollPane = new JScrollPane(table); |
| panel.add(scrollPane); |  |
| return panel; | } |
|  | private JPanel createButtonPanel() { |
| JPanel panel = new JPanel(new GridBagLayout()); | panel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5)); |
|  | GridBagConstraints gbc = new GridBagConstraints(); |
| gbc.anchor = GridBagConstraints.CENTER; | gbc.fill = GridBagConstraints.HORIZONTAL; |

**Результаты работы программы:**

****

**Вывод:** приобрел практические навыки кодирования/декодирования двоичных данных при использовании кода Хемминга.