МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЁТ

по лабораторной работе 4

по дисциплине

Защита информации

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Симанов В.С.

(подпись) (фамилия, и.,о.)

21-ИС\_\_\_\_\_\_\_\_

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2024

Цель: шифрование методом гаммирования

Задания:

Задание № 4

Реализовать возможность введения значений *p*, *q* и закрытого ключа *Kс,* вычислить открытый ключ *Ko* при помощи расширенного алгоритма Евклида и выполнить шифрование по алгоритму *RSA* открытым ключом (*Ko*) своей фамилии. Для представления букв в числовой форме использовать следующее соответствие: ‘A’ – 2, ‘Б’ – 3, ‘В’ – 4, ..., ‘Ё’ – 8, ..., ‘Я’ – 34.

«Шифрования данных **RSA**»:

алгоритм. Пусть абонент А хочет передать зашифрованное сообщение абоненту Б. В этом случае абонент Б должен подготовить пару (открытый ключ; закрытый ключ) и отправить свой открытый ключ пользователю А.

Первым этапом является генерация открытого и закрытого ключей. Для этого вначале выбираются два больших простых числа Р и Q. Затем вычисляется произведение N:

N = PQ.

После этого определяется вспомогательное число f:

f = (Р - l)(Q - 1).

Затем случайным образом выбирается число d < f и взаимно простое с f.

Далее необходимо найти число е, такое, что

еd mod f = 1.

Числа d и N будут открытым ключом пользователя, а значение е – закрытым ключом.

Таким образом, на этом этапе у пользователя должна быть информация, указанная в следующей таблице:

|  |  |  |
| --- | --- | --- |
|  | Открытый ключ | Закрытый ключ |
| Пользователь системы | N, d | e |

Так как пользователь Б хочет получить зашифрованное сообщение от пользователя А, значит пользователь Б должен отправить свой открытый ключ (d, N) пользователю А. Числа Р и Q больше не нужны, однако их нельзя никому сообщать; лучше всего их вообще забыть.

На этом этап подготовки ключей закончен и можно использовать основной протокол RSA для шифрования данных.

**Второй этап – шифрование данных**. Если абонент А хочет передать некоторые данные абоненту Б, он должен представить свое сообщение в цифровом виде и разбить его на блоки m1, m2, m3, ... , где mi < N. Зашифрованное сообщение будет состоять из блоков сi.

Абонент А шифрует каждый блок своего сообщения по формуле

ci = mid mod N

используя *открытые параметры* пользователя Б, и пересылает зашифрованное сообщение С=(с1, с2, с3, ...) по открытой линии.

Абонент Б, получивший зашифрованное сообщение, расшифровывает все блоки полученного сообщения по формуле

mi = ce mod N

Все расшифрованные блоки будут точно такими же, как и исходящие от пользователя А.

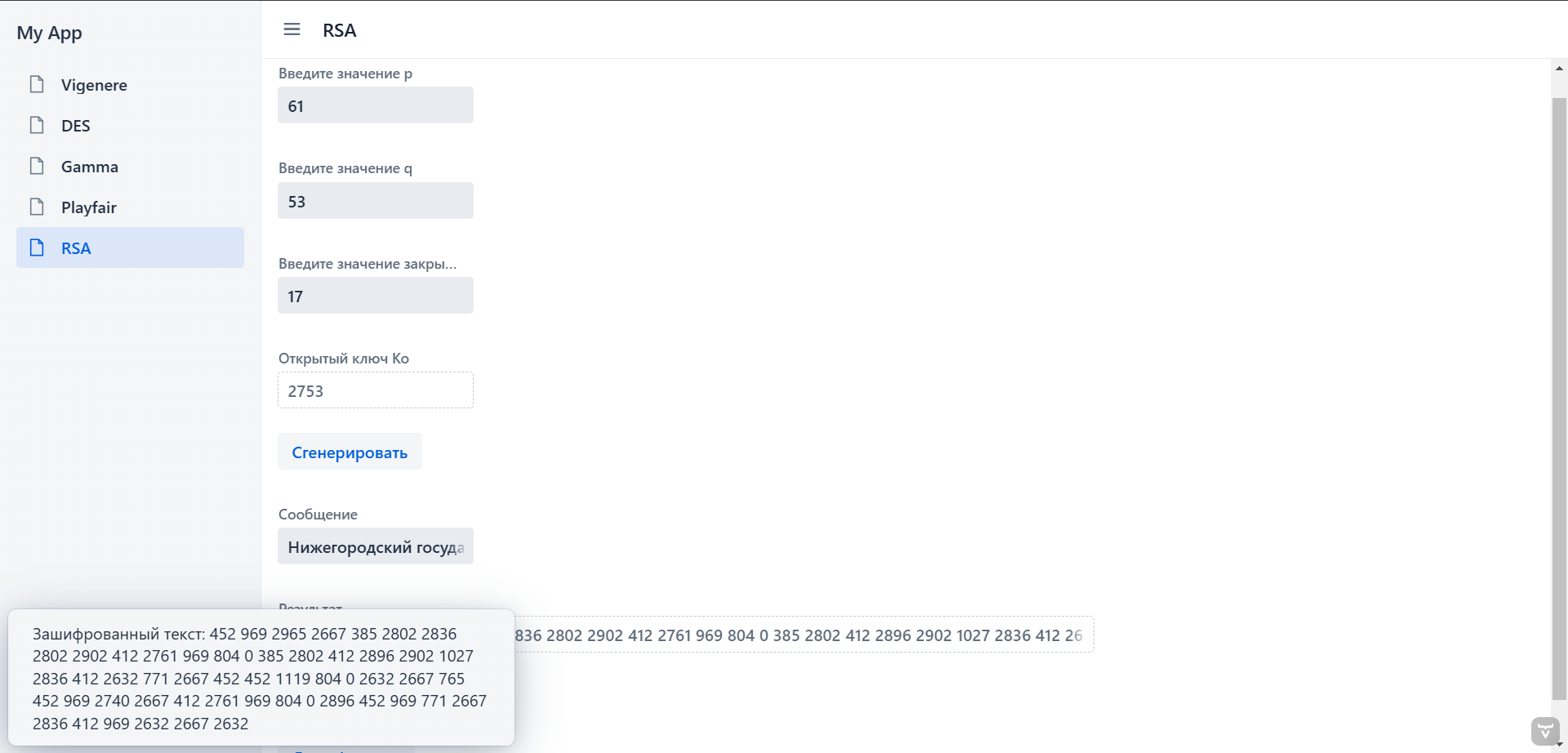
Злоумышленник, перехватывающий все сообщения и знающий всю открытую информацию, не сможет найти исходное сообщение при больших значениях Р и Q.

Примеры шифрования и расшифровывания:

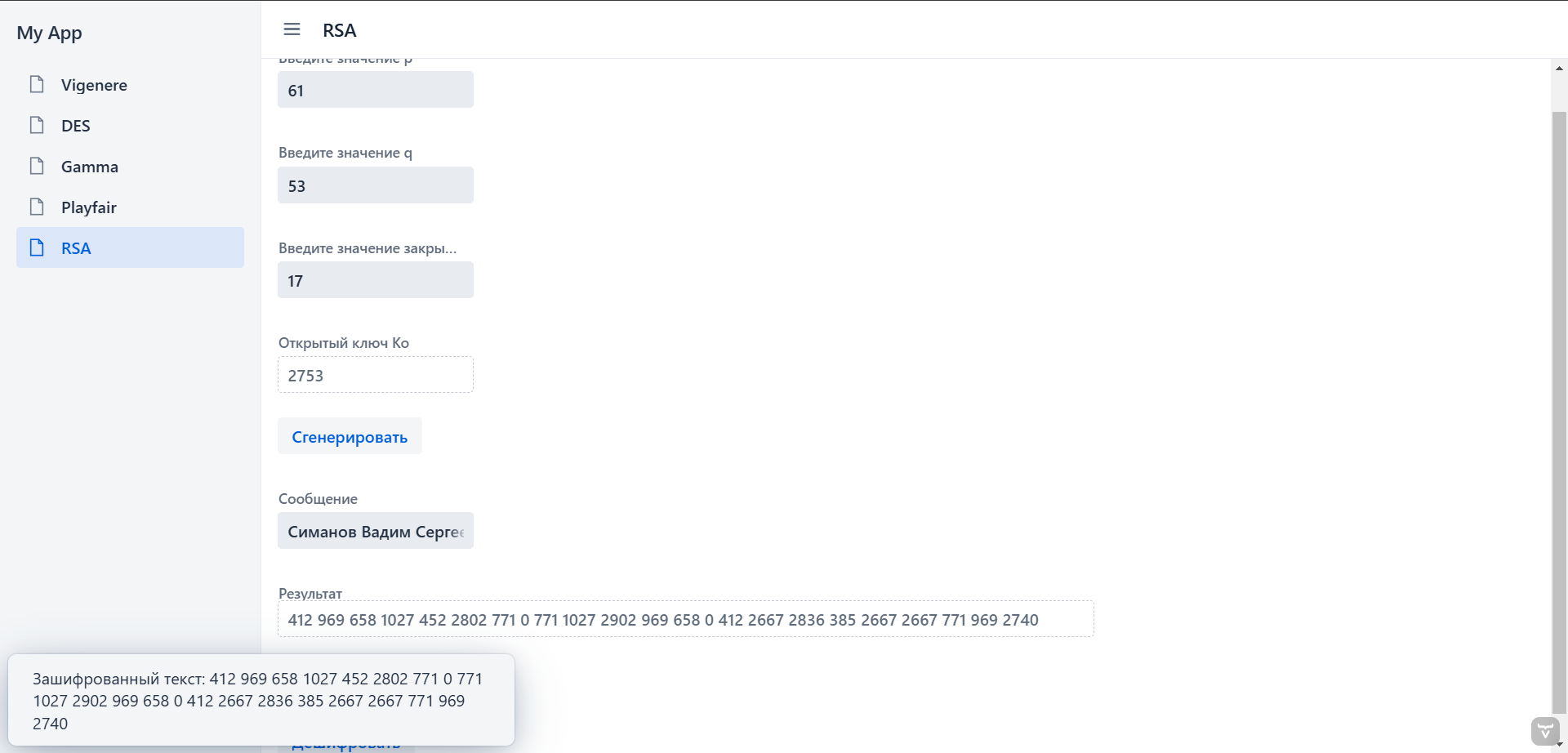
Нижегородский государственный технический университет

Шифрование:

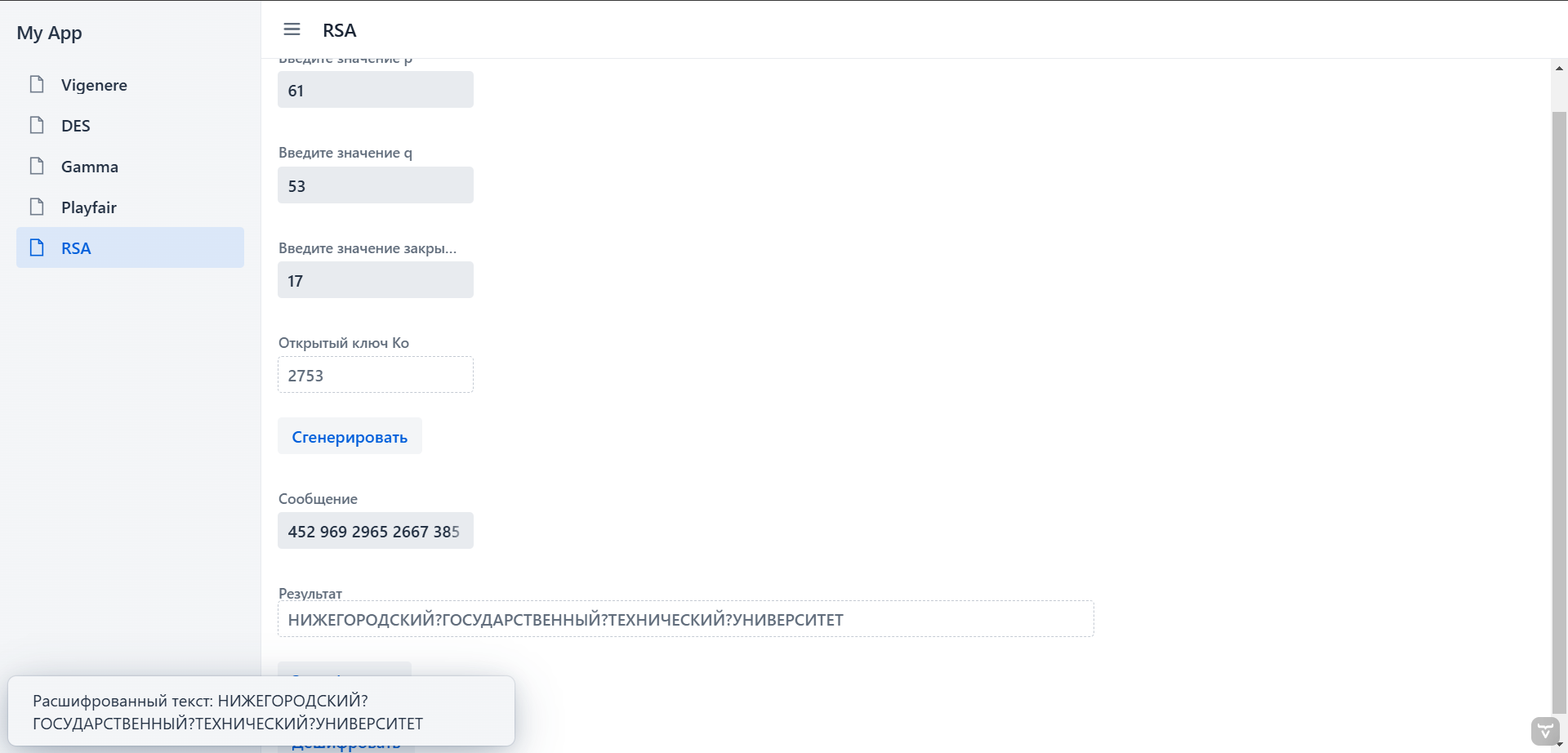
Нижегородский государственный технический университет:

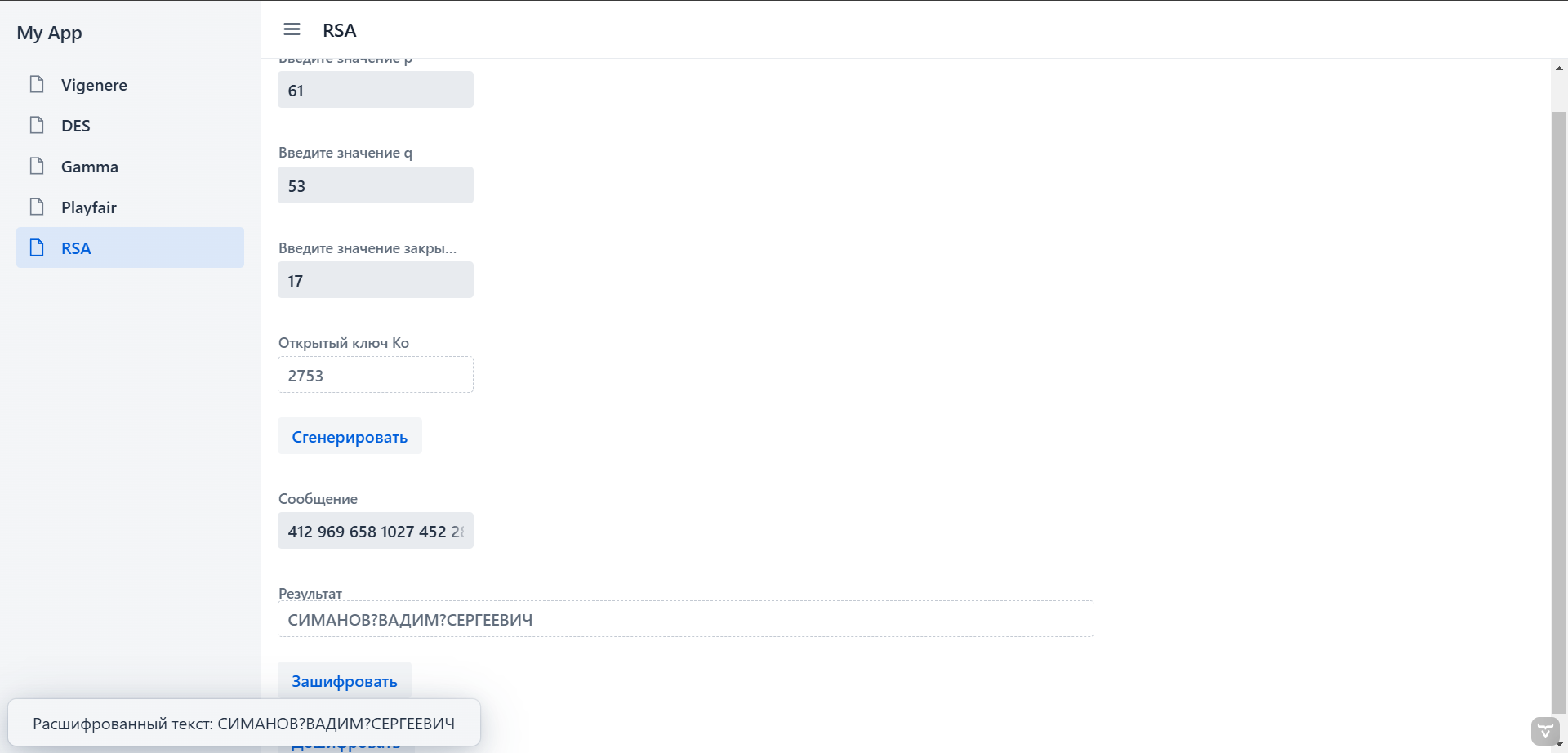


Симанов Вадим Сергеевич:



Расшифрование:





Текст программы с комментариями:

RSAView – отвечает за отображении страницы

package com.example.application.views.lb;  
  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Menu;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
  
import java.math.BigInteger;  
  
@PageTitle("RSA")  
@Menu(icon = "line-awesome/svg/file.svg", order = 0)  
@Route(value = "RSA")  
public class RSAView extends VerticalLayout {  
  
 private TextField pField = new TextField("Введите значение p");  
 private TextField qField = new TextField("Введите значение q");  
 private TextField privateKeyField = new TextField("Введите значение закрытого ключа Kc");  
 private TextField publicKeyField = new TextField("Открытый ключ Ko");  
 private TextField textField = new TextField("Сообщение");  
 private TextField result = new TextField("Результат");  
  
 private Button encryptButton = new Button("Зашифровать");  
 private Button decryptButton = new Button("Дешифровать");  
 private Button geterateButton = new Button("Сгенерировать");  
  
 BigInteger p;  
 BigInteger q;  
 BigInteger Kc;  
 String text;  
 BigInteger n;  
 BigInteger phi;  
 BigInteger Ko;  
  
 // Создаем объект класса RsaEncryption  
 private RSA rsa = new RSA();  
  
 public RSAView() {  
 add(pField, qField, privateKeyField,publicKeyField,geterateButton, textField, result, encryptButton, decryptButton);  
 publicKeyField.setReadOnly(true);  
 result.setReadOnly(true);  
 result.setWidth("800px");  
 result.setHeight("50px");  
 // Генерация открытого ключа  
 geterateButton.addClickListener(e-> {  
 try {  
 p = new BigInteger(pField.getValue());  
 q = new BigInteger(qField.getValue());  
 Kc = new BigInteger(privateKeyField.getValue());  
  
 n = p.multiply(q);  
 phi = (p.subtract(BigInteger.*ONE*)).multiply(q.subtract(BigInteger.*ONE*));  
 Ko = rsa.computePublicKey(phi, Kc);  
 publicKeyField.setValue(Ko.toString()); // Отображаем открытый ключ  
 } catch (Exception ex) {  
 throw new RuntimeException(ex);  
 }  
 });  
  
 encryptButton.addClickListener(e -> {  
 try {  
  
 // Шифрование текста  
 text = textField.getValue().toUpperCase();  
 String encryptedText = rsa.encryptText(text, Ko, n);  
 Notification.*show*("Зашифрованный текст: " + encryptedText);  
 result.setValue(encryptedText.toString()); // Заполняем поле зашифрованным текстом  
  
 } catch (Exception ex) {  
 Notification.*show*("Ошибка: " + ex.getMessage());  
 }  
 });  
  
 decryptButton.addClickListener(e -> {  
 try {  
 p = new BigInteger(pField.getValue());  
 q = new BigInteger(qField.getValue());  
 Kc = new BigInteger(privateKeyField.getValue());  
 String encryptedText = result.getValue();  
  
 // Вычисление n для дешифрования  
 n = p.multiply(q);  
  
 // Дешифрование текста  
 String decryptedText = rsa.decryptText(encryptedText, Kc, n);  
 result.setValue(decryptedText.toString());  
 Notification.*show*("Расшифрованный текст: " + decryptedText);  
 } catch (Exception ex) {  
 Notification.*show*("Ошибка: " + ex.getMessage());  
 }  
 });  
 }  
}

RSA – метод реализующий алгоритм шифрования данных

package com.example.application.views.lb;  
  
import java.math.BigInteger;  
import java.util.HashMap;  
import java.util.Map;  
  
public class RSA {  
 // Метод для вычисления открытого ключа Ko  
 public BigInteger computePublicKey(BigInteger phi, BigInteger Kc) throws Exception {  
 BigInteger[] gcdResult = extendedEuclideanAlgorithm(Kc, phi);  
 BigInteger Ko = gcdResult[1];  
 if (Ko.compareTo(BigInteger.*ZERO*) < 0) {  
 Ko = Ko.add(phi);  
 }  
 return Ko;  
 }  
  
 // Реализация расширенного алгоритма Евклида  
 private BigInteger[] extendedEuclideanAlgorithm(BigInteger a, BigInteger b) {  
 if (b.equals(BigInteger.*ZERO*)) {  
 return new BigInteger[]{a, BigInteger.*ONE*, BigInteger.*ZERO*};  
 } else {  
 BigInteger[] vals = extendedEuclideanAlgorithm(b, a.mod(b));  
 BigInteger d = vals[0];  
 BigInteger x = vals[2];  
 BigInteger y = vals[1].subtract((a.divide(b)).multiply(vals[2]));  
 return new BigInteger[]{d, x, y};  
 }  
 }  
  
 // Метод для преобразования текста в числа и шифрования по RSA  
 public String encryptText(String text, BigInteger Ko, BigInteger n) {  
 StringBuilder encryptedText = new StringBuilder();  
 Map<Character, Integer> charToNumberMap = getCharToNumberMap();  
  
 for (char ch : text.toCharArray()) {  
 int num = charToNumberMap.getOrDefault(ch, 0); // Преобразуем символ в число  
 BigInteger encryptedChar = BigInteger.*valueOf*(num).modPow(Ko, n);  
 encryptedText.append(encryptedChar.toString()).append(" ");  
 }  
  
 return encryptedText.toString().trim();  
 }  
  
 // Метод для расшифровки текста с использованием закрытого ключа Kc  
 public String decryptText(String encryptedText, BigInteger Kc, BigInteger n) {  
 StringBuilder decryptedText = new StringBuilder();  
 Map<Integer, Character> numberToCharMap = getNumberToCharMap();  
  
 // Разделение зашифрованного текста по пробелам, чтобы получить зашифрованные числа  
 String[] encryptedNumbers = encryptedText.split(" ");  
 for (String numStr : encryptedNumbers) {  
 BigInteger encryptedNum = new BigInteger(numStr);  
 BigInteger decryptedNum = encryptedNum.modPow(Kc, n); // Расшифровка числа  
  
 // Преобразование расшифрованного числа в символ  
 char decryptedChar = numberToCharMap.getOrDefault(decryptedNum.intValue(), '?');  
 decryptedText.append(decryptedChar);  
 }  
  
 return decryptedText.toString();  
 }  
  
 // Метод для создания соответствия 'А'-2, 'Б'-3 ... 'Я'-34  
 private Map<Character, Integer> getCharToNumberMap() {  
 Map<Character, Integer> map = new HashMap<>();  
 char startChar = 'А';  
 for (int i = 2; i <= 34; i++) {  
 map.put(startChar++, i);  
 }  
 return map;  
 }  
  
 // Обратное соответствие для расшифровки: 2-'А', 3-'Б' ... 34-'Я'  
 private Map<Integer, Character> getNumberToCharMap() {  
 Map<Integer, Character> map = new HashMap<>();  
 char startChar = 'А';  
 for (int i = 2; i <= 34; i++) {  
 map.put(i, startChar++);  
 }  
 return map;  
 }  
}

Application – запуск программы

package com.example.application;  
  
import com.vaadin.flow.component.page.AppShellConfigurator;  
import com.vaadin.flow.theme.Theme;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
@Theme(value = "my-app")  
public class Application implements AppShellConfigurator {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Application.class, args);  
 }  
  
}

Вывод: мною был реализован алгоритм шифрования данных RSA