Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ

Кафедра интеллектуальных информационных технологий

Лабораторная работа № 5

“Вариант 3”

Отчет

Выполнил:

студент гр. 221703

Веркович Е.В.

Проверил: Крищенович В.А.

Минск 2024

**Задание**

Разработать программное обеспечение, реализующее функции генерации секретного и открытого ключей, шифрования и цифровой подписи для алгоритма RSA. Обмен входными и выходными данными должен осуществляться через файлы: – открытого ключа; – секретного ключа; – исходного сообщения; – зашифрованного сообщения.

Листинг когда:

public class RSA {  
  
 private static final int *BIT\_LENGTH* = 1024;  
 private static final SecureRandom *random* = new SecureRandom();  
 private BigInteger p, q, n, phi, e, d;  
  
 *// Конструктор для генерации ключей* public RSA() {  
 *// Генерация двух больших простых чисел p и q* p = BigInteger.*probablePrime*(*BIT\_LENGTH*, *random*);  
 q = BigInteger.*probablePrime*(*BIT\_LENGTH*, *random*);  
 n = p.multiply(q);  
  
 *// Вычисление функции Эйлера φ(n)* phi = (p.subtract(BigInteger.*ONE*)).multiply(q.subtract(BigInteger.*ONE*));  
  
 *// Выбор e: 65537 — стандартное значение, используемое в RSA* e = BigInteger.*valueOf*(65537);  
  
 *// Проверка, что e и φ(n) взаимно просты* while (!gcd(e,phi).equals(BigInteger.*ONE*)) {  
 e = e.add(BigInteger.*TWO*);  
 }  
  
 *// Вычисление закрытого ключа d, обратного к e по модулю φ(n)* d = modInverse(e,phi);  
 }  
  
 *// Шифрование сообщения M -> C (M^e mod n)* public BigInteger encrypt(BigInteger message) {  
 return message.modPow(e, n);  
 }  
  
 *// Расшифрование сообщения C -> M (C^d mod n)* public BigInteger decrypt(BigInteger ciphertext) {  
 return ciphertext.modPow(d, n);  
 }  
  
 *// Подпись сообщения: s = m^d mod n* public BigInteger sign(BigInteger message) {  
 return message.modPow(d, n);  
 }  
  
 *// Проверка подписи: m\* = s^e mod n* public boolean verify(BigInteger message, BigInteger signature) {  
 BigInteger decryptedMessage = signature.modPow(e, n);  
 return decryptedMessage.equals(message);  
 }  
  
 *// Сохранение ключей в файлы* public void saveKeys() throws IOException {  
 Files.*write*(Paths.*get*("public\_key.txt"), (e.toString() + "\n" + n.toString()).getBytes());  
 Files.*write*(Paths.*get*("private\_key.txt"), (d.toString() + "\n" + n.toString()).getBytes());  
 }

private BigInteger gcd(BigInteger a, BigInteger b) {  
 while (!b.equals(BigInteger.*ZERO*)) {  
 BigInteger temp = b;  
 b = a.mod(b);  
 a = temp;  
 }  
 return a;  
 }  
  
 private BigInteger modInverse(BigInteger a, BigInteger m) {  
 BigInteger m0 = m;  
 BigInteger y = BigInteger.*ZERO*, x = BigInteger.*ONE*;  
  
 if (m.equals(BigInteger.*ONE*))  
 return BigInteger.*ZERO*;  
  
 while (a.compareTo(BigInteger.*ONE*) > 0) {  
 *// q - это частное* BigInteger q = a.divide(m);  
 BigInteger t = m;  
  
 *// m теперь остаток от предыдущего деления* m = a.mod(m);  
 a = t;  
 t = y;  
  
 *// Обновляем x и y* y = x.subtract(q.multiply(y));  
 x = t;  
 }  
  
 *// Убедимся, что x положительно* if (x.compareTo(BigInteger.*ZERO*) < 0)  
 x = x.add(m0);  
  
 return x;  
 }  
  
 *// Загрузка ключей из файлов* public static RSA loadKeys() throws IOException {  
 RSA rsa = new RSA();  
 String[] publicKey = Files.*readAllLines*(Paths.*get*("public\_key.txt")).toArray(new String[0]);  
 rsa.e = new BigInteger(publicKey[0]);  
 rsa.n = new BigInteger(publicKey[1]);  
  
 String[] privateKey = Files.*readAllLines*(Paths.*get*("private\_key.txt")).toArray(new String[0]);  
 rsa.d = new BigInteger(privateKey[0]);  
 rsa.n = new BigInteger(privateKey[1]);  
 return rsa;  
 }  
}

**Тестирование программы:**







