Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Пакеты и приложения |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | Н. А. Самарин |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | Студент КИ21-17/1Б, |  |  |  | Н. А. Самарин |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2024

**СОДЕРЖАНИЕ**

1 Задание............................................................................................................... 3

2 Исходный код разработанного алгоритма..................................................... 3

3 Результат........................................................................................................... 7

**1 Задание**

Изучить возможности классов BigInt в Scala и класса BigInteger в Java.  
Используя эти классы, реализовать на Scala алгоритм RSA и вспомогательные  
функции для чисел любой размерности и оформить их в отдельный пакет:

- def isPrime (n: BigInt) - возвращает true, если число n простое и false,  
если составное;

- def primes(n: BigInt, m: BigInt) - возвращает список (List) всех простых  
чисел от n до m включительно;

- def randomPrime(bits: Int) - возвращает случайное простое число BigInt с  
длиной в битах: bits ;

- def encrypt(s: String, e: BigInt, n: BigInt) - возвращает зашифрованную  
строку s открытым ключом (e, n); результатом должен быть массив  
приведенный к строке: 123456,234567,345678,...;

- def decrypt(s: String, d: BigInt, n: BigInt) - расшифровывает s секретным  
ключом (d, n); результатом должна быть исходная строка до ее шифрования.

Написать с использованием данного пакета утилиту (по выбору: либо для  
командной строки, либо GUI c использованием Spring), позволяющую решать  
следующие задачи:

- генерировать диапазон простых чисел и позволять их выбирать из  
списка;

- генерировать ключи RSA, основанные на простых числах любой  
размерности;

- разделять открытый и секретный ключи RSA (хранить их раздельно) ;

- вводить (или вставлять из буфера) сообщение для шифрования;

- шифровать сообщение открытым ключом;

- отображать зашифрованное сообщение в виде текста или сохранять  
зашифрованное сообщение в файле для возможности его передачи;

- прочитать зашифрованное сообщение из файла при его получении;

- расшифровывать полученное сообщение секретным ключом.

**2 Исходный код разработанного алгоритма**

Листинг 1 – Main.scala

import java.io.PrintWriter  
import scala.annotation.tailrec  
import scala.io.{Source, StdIn}  
object Main {  
 var rsa: Option[RSA] = None  
 private var isRun = true  
 def main(args: Array[String]): Unit = {  
 while (isRun) {  
 println("1. Создание RSA из P Q")  
 println("2. Создание RSA из существующих ключей")  
 println("3. Зашифровать сообщение")  
 println("4. Расшифровать сообщение")  
 println("5. Зашифровать в файл")  
 println("6. Расшифровать из файла")

Продолжение листинга 1

println("7. Выход")  
 var choice: Int = -1  
 try {  
 choice = scala.io.StdIn.readInt()  
 } catch {  
 case e: NumberFormatException =>  
 }  
 choice match {  
 case 1 => makeRsaFromPq()  
 case 2 => makeRsaFromExists()  
 case 3 =>  
 if (rsa.isDefined) encryptFromConsole()  
 else println("Сначала создайте RSA")  
 case 4 =>  
 if (rsa.isDefined) decryptFromConsole()  
 else println("Сначала создайте RSA")  
 case 5 =>  
 if (rsa.isDefined) encryptToFile()  
 else println("Сначала создайте RSA")  
 case 6 =>  
 if (rsa.isDefined) decryptFromFile()  
 else println("Сначала создайте RSA")  
 case 7 => isRun = false  
 case \_ => println("Некорректный ввод")  
 }  
 }  
 }  
 private def makeRsaFromPq(): Unit = {  
 val (p, q) = pickPQ()  
 val n = p \* q  
 val phi = (p - 1) \* (q - 1)  
 @tailrec  
 def findE(e: BigInt): BigInt = {  
 if (e.gcd(phi) == 1) e  
 else findE(Utils.randomPrime(phi.bitLength))  
 }  
 val e = findE(Utils.randomPrime(phi.bitLength))  
 val d = e.modInverse(phi)  
 val rsa = new RSA(n, e, d)  
 println(s"Создан RSAEncoder: $rsa")  
 this.rsa = Option.apply(rsa)  
 }  
 private def makeRsaFromExists(): Unit = {  
 val n = Utils.readBigInt("Введите значение для N:")  
 val e = Utils.readBigInt("Введите значение для E:")  
 val d = Utils.readBigInt("Введите значение для D:")  
 val rsa = new RSA(n, e, d)  
 println(s"Создан RSAEncoder: $rsa")  
 this.rsa = Option.apply(rsa)  
 }  
 private def encryptFromConsole(): Unit = {  
 println("Введите сообщение:")  
 val message = StdIn.readLine()  
 println(s"Зашифрованное сообщение:\n${rsa.get.encrypt(message)}")  
 }  
 private def decryptFromConsole(): Unit = {  
 println("Введите сообщение:")  
 val message = StdIn.readLine()

Продолжение листинга 1

try {  
 println(s"Расшифрованное сообщение:\n${rsa.get.decrypt(message)}")  
 } catch {  
 case e: NumberFormatException => System.err.println(s"Ошибка: некорректный  
 ввод ${e.getMessage}")  
 }  
 }  
 private def encryptToFile(): Unit = {  
 println("Введите имя файла:")  
 val filePath = StdIn.readLine()  
 println("Введите сообщение:")  
 val message = StdIn.readLine()  
 val enc = rsa.get.encrypt(message)  
 val writer = new PrintWriter(filePath)  
 try {  
 writer.write(enc)  
 println("Зашифрованное сообщение сохранено в файл")  
 } catch {  
 case e: Exception => System.err.println(e.getMessage)  
 } finally {  
 writer.close()  
 }  
 }  
 private def decryptFromFile(): Unit = {  
 println("Введите имя файла:")  
 val filePath = StdIn.readLine()  
 val source = Source.fromFile(filePath)  
 try {  
 println(s"Расшифрованное сообщение:\n${rsa.get.decrypt(source.mkString)}")  
 } catch {  
 case e: Exception => System.err.println(e.getMessage)  
 } finally {  
 source.close()  
 }  
 }  
 private def pickPQ(): (BigInt, BigInt) = {  
 val primes = pickPrimes()  
 primes.zipWithIndex.foreach {  
 case (prime, index) =>  
 println(s"$index. $prime")  
 }  
 val p = primes(chooseIndex("Выберете индекс числа для p", primes.size))  
 val q = primes(chooseIndex("Выберете индекс числа для q", primes.size))  
 println(s"p: $p\nq: $q")  
 (p, q)  
 }  
 @tailrec  
 private def pickPrimes(): List[BigInt] = {  
 val leftBound = Utils.readBigInt("Введите левую границу диапазона простых чи  
 сел:")  
 val rightBound = Utils.readBigInt("Введите правую границу диапазона простых   
 чисел:")  
 if (rightBound < leftBound || leftBound < 1 || rightBound < 1) {  
 System.err.println("Ошибка Некорректный ввод")  
 pickPrimes()  
 } else {  
 val primes = Utils.primes(leftBound, rightBound)  
 if (primes.isEmpty) {

Окончание листинга 1

println("В диапазоне нет простых чисел")  
 pickPrimes()  
 } else {  
 primes  
 }  
 }  
 }  
 @tailrec  
 private def chooseIndex(prompt: String, listSize: Int): Int = {  
 println(prompt)  
 val index = StdIn.readInt()  
 if (index < 0 || index >= listSize) {  
 System.err.println("Ошибка: введите корректный индекс.")  
 chooseIndex(prompt, listSize)  
 } else {  
 index  
 }  
 }  
}

Листинг 2 – RSA.scala

import scala.annotation.tailrec  
import scala.util.Random  
class RSA (val n: BigInt, val e: BigInt, val d: BigInt) {  
 override def toString: String = {  
 s"RSA(public\_key=($e, $n), private\_key=($d, $n))"  
 }  
 def publicKey: (BigInt, BigInt) = (e, n)  
 def privateKey: (BigInt, BigInt) = (d, n)  
 def encrypt(s: String): String = {  
 Utils.encrypt(s, e, n)  
 }  
 def decrypt(s: String): String = {  
 Utils.decrypt(s, d, n)  
 }  
}

Листинг 3 – Utils.scala

import scala.annotation.tailrec  
import scala.io.StdIn  
import scala.util.Random  
object Utils {  
 private val delimiter = ","  
 private val randomizer = new Random()  
 def isPrime(n: BigInt): Boolean = n.isProbablePrime(100)  
 def primes(n: BigInt, m: BigInt): List[BigInt] = (n to m).filter(isPrime).toLi  
 st  
 def randomPrime(bits: Int): BigInt = BigInt.probablePrime(bits, randomizer)  
 def encrypt(s: String, e: BigInt, n: BigInt): String = {  
 s.toCharArray.map(  
 BigInt(\_)  
 .modPow(e, n)  
 ).mkString(delimiter)  
 }  
 def decrypt(s: String, d: BigInt, n: BigInt): String = {  
 s.split(delimiter).map(  
 BigInt(\_)

Окончание листинга 3

.modPow(d, n)  
 .toChar  
 ).mkString("")  
 }  
 @tailrec  
 def readBigInt(prompt: String): BigInt = {  
 println(prompt)  
 try {  
 BigInt(StdIn.readLine())  
 } catch {  
 case \_: NumberFormatException =>  
 System.err.println("Ошибка: введите числовые значения.")  
 readBigInt(prompt)  
 }  
 }  
}

**3 Результат**

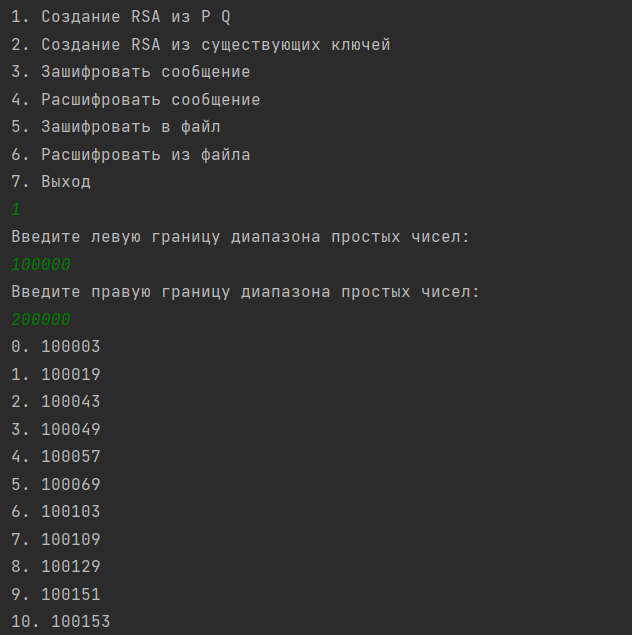


Рисунок 1 – Результат работы алгоритма 1

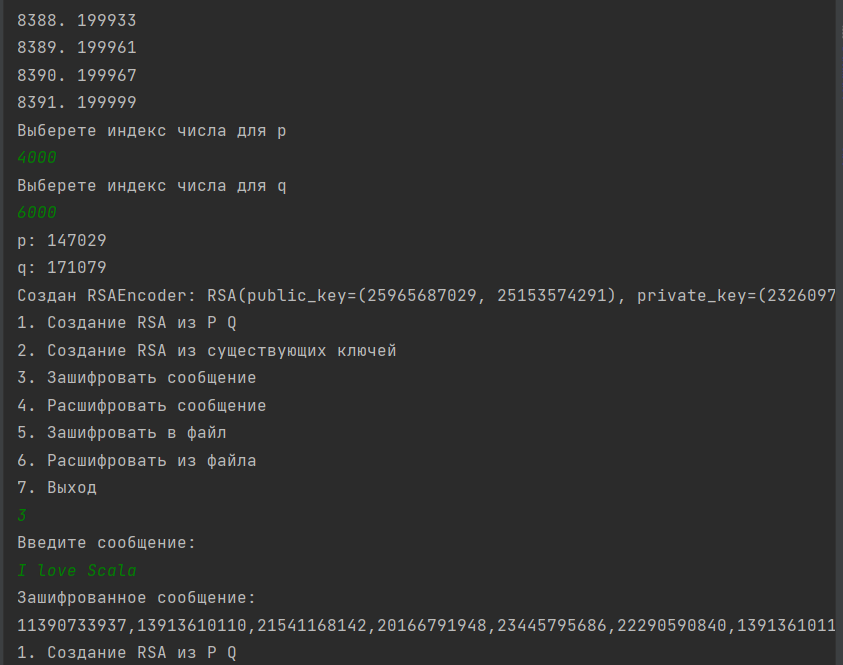


Рисунок 2 – Результат работы алгоритма 2

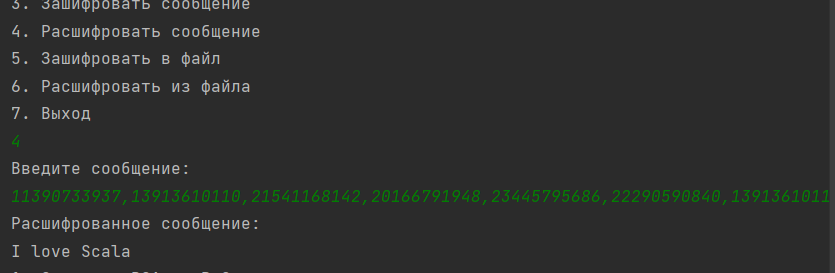


Рисунок 3 – Результат работы алгоритма 3

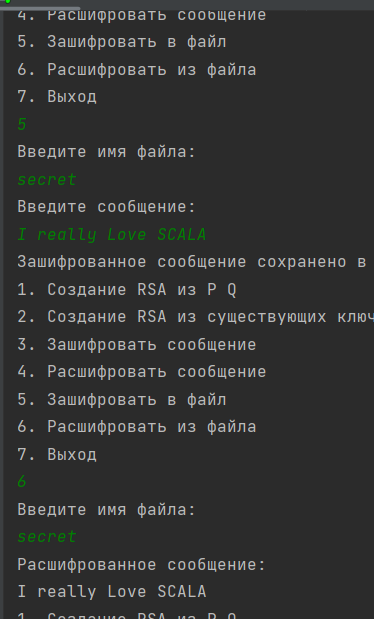


Рисунок 4 – Результат работы алгоритма 4