МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе № 7

по дисциплине «Методы и средства защиты информации»

на тему «RSA с программной реализацией»

Выполнил студент группы ФИб-4302-51-00 \_\_\_\_\_\_\_\_\_\_\_ / Д.А. Савин /

Проверил доцент кафедры ПМиИ \_\_\_\_\_\_\_\_\_\_\_ / Д.Ю. Ляпунов /

Киров 2021

**Оглавление**

[Введение 3](#_Toc92898937)

[Описание алгоритма RSA 4](#_Toc92898938)

[Программная реализация 4](#_Toc92898939)

Введение

**Цель работы:** ознакомиться с ассиметричным алгоритмом шифрования RSA, написать его реализацию.

**Задачи:**

1. Поиск литературы, содержащей необходимую информацию по теме;
2. Рассмотрение теории по теме алгоритма;
3. Шифрование и дешифрование некоторого текста с помощью реализованной программы;
4. Формулировка соответствующих выводов;
5. Составление вопросов для итогового теста с целью закрепления материала по данной теме.

Описание алгоритма RSA

RSA – криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел. Создан в 1977 году группой учёных. Метод получил своё название по первым буквам фамилий этих учёных: R – Ривест, S – Шамир, A – Адлеман. [1]

Работа данного шифрования заключается в генерации ключей (открытого и закрытого). Пользователь создаёт открытый ключ, основанный на двух больших простых числах (типа long) и вспомогательных значениях. Сложность расшифровки сообщения прямо пропорциональна длине входного ключа.

RSA является относительно медленным алгоритмом, в следствие чего не так широко используется пользователями для шифрования своих личных данных. Однако, чаще всего этот метод используют для передачи в зашифрованном виде общих ключей для симметричного ключа шифрования, который, в свою очередь, может выполнять операции массового шифрования и дешифрования гораздо более высокой скорости.

Программная реализация

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Numerics;  using System.Text;  using System.Threading.Tasks;  namespace RSA  {  internal class Program  {  static char[] characters = new char[] { '#', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ы', 'Ъ', 'Э', 'Ю', 'Я', ' ', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0' };  static void Main(string[] args)  {  try  {  Console.Write("Введите p: ");  long p = long.Parse(Console.ReadLine());  Console.Write("Введите q: ");  long q = long.Parse(Console.ReadLine());  List<string> Result = new List<string>();  long d, n;  if (MillerRabinTest(p) && MillerRabinTest(q))  {  Console.Write("Введите текст: ");  string Text = Console.ReadLine();  Text = Text.ToUpper();  n = p \* q;  long Euler = (p - 1) \* (q - 1);  d = GetPrivateKey(Euler);  long e = GetPublicKey(d, Euler);  Result = Cipher(Text, e, n);  for (int i = 0; i < Result.Count; i++)  {  Console.Write("{0} ", Result[i]);  }  Console.WriteLine();  Console.WriteLine(d.ToString());  Console.WriteLine(n.ToString());  }  else throw new ArgumentException();  long d1 = d;  long n1 = n;  List<string> Decypher = new List<string>();  for (int i = 0; i < Result.Count; i++)  {  Decypher.Add(Result[i]);  }  string rez = DeCypher(Decypher, d1, n1);  Console.WriteLine(rez);  }  catch (ArgumentException)  {  Console.WriteLine("Введены не простые числа");  }  Console.ReadKey();  }  static int NodMod(int a, int b)  {  if (b == 0) return a;  else if (a == 0) return b;  else return NodMod(b, a % b);  }  static long GetPrivateKey(long m)  {  long d = m - 1;  for (long i = 2; i <= m; i++)  {  if (m % i == 0 && d % i == 0)  {  d--;  i = 1;  }  }  return d;  }  static long GetPublicKey(long d, long m)  {  long e = 10;  while (true)  {  if ((e \* d) % m == 1) break;  else e++;  }  return e;  }  static List<string> Cipher(string s, long e, long n)  {  List<string> Result = new List<string>();  long big;  for (int i = 0; i < s.Length; i++)  {  int index = Array.IndexOf(characters, s[i]);  big = index;  big = ModPow(big, (int)e, (int)n);  //big %= (int)n;  Result.Add(big.ToString());  }  return Result;  }  static string DeCypher(List<string> Input, long d, long n)  {  string s = "";  long big;  foreach (string item in Input)  {  big = (long)Convert.ToDouble(item);  big = ModPow(big, (int)d, (int)n);  int index = Convert.ToInt32(big.ToString());  s += characters[index].ToString();  }  return s;  }  static bool MillerRabinTest(long digit)  {  if (digit == 2 || digit == 3) return true;  else if (digit < 2 || digit % 2 == 0) return false;  long t = digit - 1;  int count = 0;  while (t % 2 == 0)  {  t /= 2;  count++;  }  Console.Write("Введите количество проверок:");  int k = int.Parse(Console.ReadLine());  for (int i = 0; i < k; i++)  {  long a;  do  {  a = RandomGeneration(digit);  } while (a < 2 || a >= digit - 2);  long x = ModPow(a, t, digit);  if (x == 1 || x == digit - 1) continue;  for (int r = 1; r < count; r++)  {  x = (long)Math.Pow(x, 2) % digit;  if (x == 1) return false;  if (x == digit - 1) break;  }  if (x != digit - 1) return false;  }  return true;  }  static long RandomGeneration(long digit)  {  Random Rand = new Random();  digit = digit - 2;  long newDigit = 0;  List<byte> randomDigit = new List<byte>();  for (int i = 0; i < Size(digit); i++)  {  randomDigit.Add((byte)Rand.Next(0, 10));  newDigit = newDigit \* 10 + randomDigit[i];  }  return newDigit;  }  static long ModPow(long a, long m, long p)  {  long Result = 1;  while (m != 0)  {  if (m % 2 == 0)  {  m /= 2;  a \*= a;  a %= p;  }  else  {  m--;  Result \*= a;  Result %= p;  }  }  return Result % p;  }  static int Size(long digit)  {  int Size = 0;  while (digit > 0)  {  digit /= 10;  Size++;  }  return Size;  }    }  } |

Изображение выглядит как текст

Автоматически созданное описание