Федеральное государственное автономное

образовательное учреждение

высшего профессионального образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

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

институт

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

кафедра

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

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

тема

\_\_\_\_\_\_\_\_\_\_\_\_Алгоритм шифрования RSA\_\_\_\_\_\_\_\_\_\_

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

Преподаватель \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы, фамилия

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Оглавление

[Задание 3](#_Toc416208793)

[Описание алгоритма 4](#_Toc416208794)

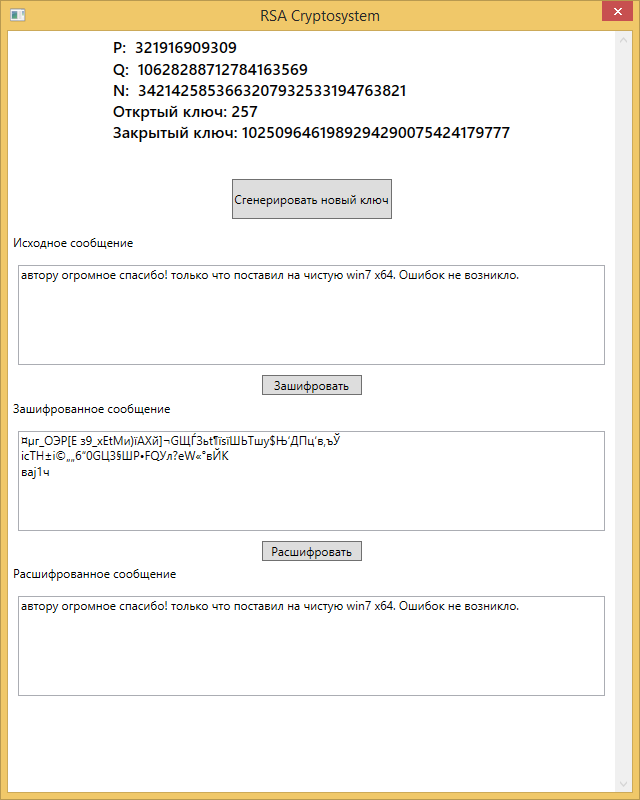
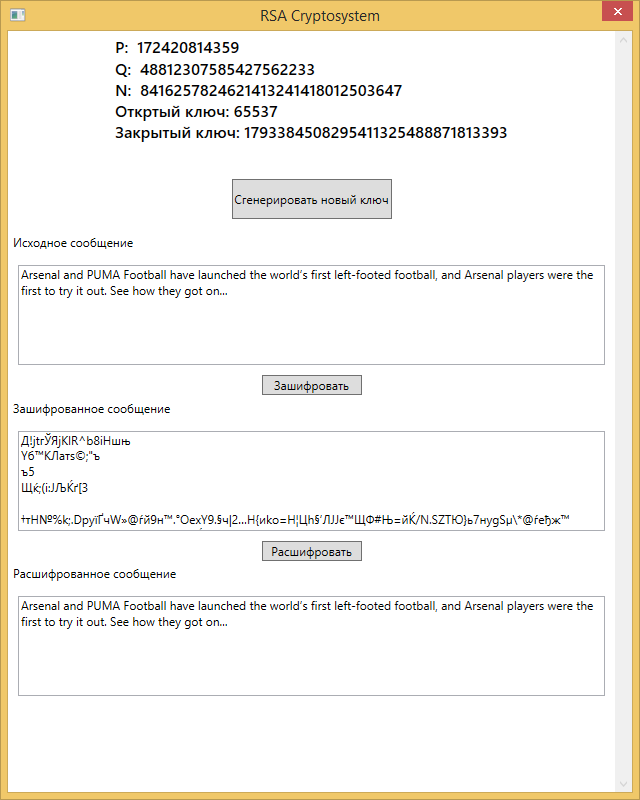
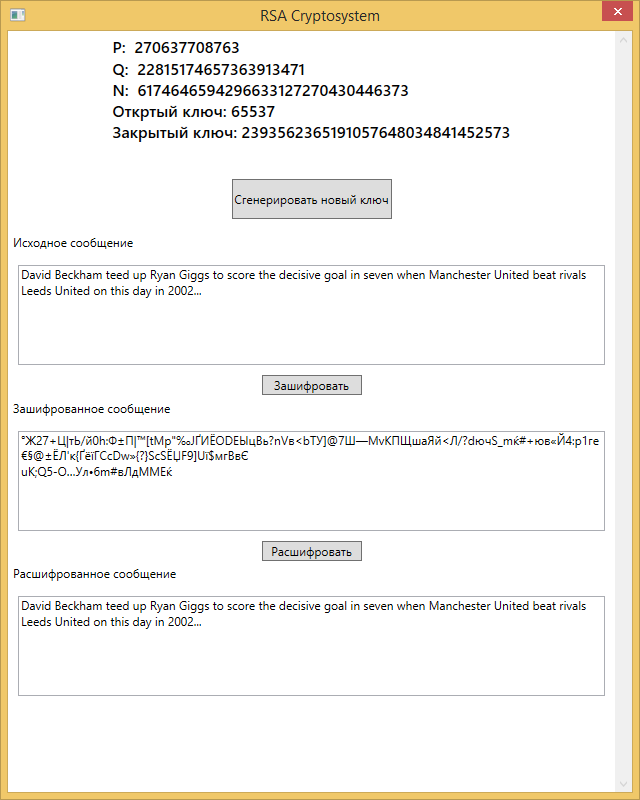
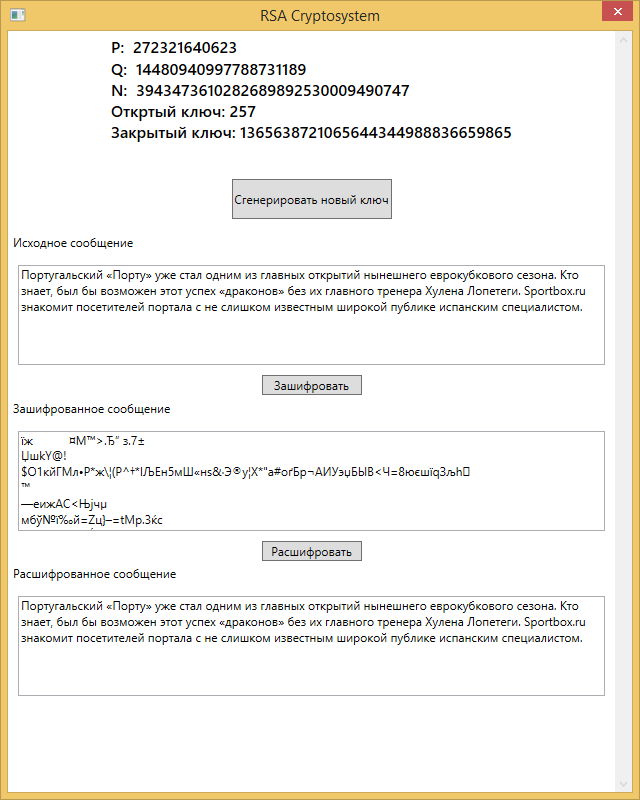
[Блок-схема алгоритма 5](#_Toc416208795)

[Листинг программы, реализующей алгоритмы 6](#_Toc416208796)

[Класс RSACryptography 6](#_Toc416208797)

[Класс MessageManager 8](#_Toc416208798)

[Примеры работы программы 10](#_Toc416208799)

[**** 11](#_Toc416208800)

[Заключение 15](#_Toc416208801)

# Задание

1. Составить в виде блок-схемы алгоритм шифрования/дешифрования RSA, со следующими особенностями:

* объём исходного текста – любой (в разумных пределах);
* исходный текст может состоять из русских и английских букв, цифр, а также знаков препинания;
* исходный текст находится в кодировке ASCII;
* N состоит из 31 десятичного знака.
* Числа P и Q выбираются случайным образом, так, что , где P и Q – простые числа.
* исходный текст разбивается на K блоков, где K выбирается исходя из значения модуля N

1. Убедиться в правильности составления алгоритмов, а затем на языке C# составить программу, которая реализует данный алгоритм.
2. На ряде контрольных примеров (не менее 10) открытого текста проверить правильность работы алгоритмов шифрования и дешифрования (в качестве контрольного примера понимается текстовый файл в кодировке ASCII).
3. Оценить криптостойкость моего варианта алгоритма RSA, а также сделать оценку производительности, разработанной программы.

# Описание алгоритма

1. **Генерация открытого и закрытого ключей**
   1. Выбираются два различных случайных простых числа заданного размера.
   2. Вычисляется их произведение
   3. Вычисляется значение *функции Эйлера*
   4. Случайным образом выбирается число и взаимно простое с
   5. Вычисляется ***e***, такое что - обратный элемент в кольце целых чисел
2. **Шифрование**
   1. Сообщение ***M*** разбивается на на ***K***, блоков : так что размер каждого блока **.**
   2. Каждый из блоков шифруется по формуле
   3. Зашифрованное сообщениепередается.
3. **Расшифровка**
   1. Для каждого блока из полученного  вычисляется

* 1. Блоки объединяются в сообщение.
  2. - исходное сообщение

# Блок-схема алгоритма



# Листинг программы, реализующей алгоритмы

Программа разработана с использованием технологий ООП и инкапсуляцией алгоритмов шифратора в объекте класса RSACryptography. А класс MessageManager занимается конвертацией строк в массив байт и обратно, а такаже разбиением сообщений на блоки.

## Класс RSACryptography

public class RSACryptography

{

private int[] fermatNumbers = new int[] { 17, 257, 65537 };//Числа ферма

private BigInteger N, P, Q;//P,Q -простые числа. N - их произведение

private BigInteger public\_key, private\_key;//откртый и закрытый ключи

#region Свойства

public BigInteger Module

{//Возвращает значения модуля(N)

get

{

return N;

}

}

public BigInteger PrivateKey

{//значение закрытого ключа

get

{

return private\_key;

}

}

public BigInteger PublicKey

{//значение откртыго ключа

get

{

return public\_key;

}

}

#endregion

public RSACryptography(BigInteger \_P,BigInteger \_Q)

{//Инициализация класса

this.P = \_P;

this.Q = \_Q;

this.N = P \* Q;

BigInteger eulierFunction = (P - 1) \* (Q - 1);

this.public\_key = fermatNumbers[new Random().Next(0,fermatNumbers.Length)];

this.private\_key = calcPrivateKey(eulierFunction);

}

public byte[] Encrypt(byte[] message)

{//Шифруем сообщение открытым ключом

BigInteger val = new BigInteger(message);

val = BigInteger.ModPow(val, public\_key, N);

return val.ToByteArray();

}

public long ModuleLength

{//Длина модуля в байтах

get

{

long l = N.ToByteArray().Length;

return l;

}

}

public byte[] Decrypt(byte[] message)

{//Расшифруем cообщение открытм ключом

BigInteger val = new BigInteger(message);

val = BigInteger.ModPow(val, private\_key, N);

return val.ToByteArray();

}

private BigInteger calcPrivateKey(BigInteger E)

{//Вычисляем закрытый ключ, находя обратный по модулю элемент кольца

BigInteger x,y;

BigInteger g = GCD(public\_key,E ,out x,out y);

return (x % E + E) % E;

}

private BigInteger GCD(BigInteger a, BigInteger b,out BigInteger x,out BigInteger y)

{//Расширеный Алгоритм Евклида

if (a.IsZero)//Находит НОД чисел A и B, и коэфициенты x,y уравнения Ax +By = НОД(A,B)

{

x = 0;

y = 1;

return b;

}

BigInteger x1, y1;

BigInteger d = GCD(b % a, a, out x1, out y1);

x = y1 - (b / a) \* x1;

y = x1;

return d;

}

}

## Класс MessageManager

public class MessageManager

{

//Класс работающий с кодировками и разбиением текста

public static String PartitionToString(List<byte[]> p)

{//Переобразует разбиение в строку

String result = String.Empty;

foreach (byte[] block in p)

{

result += GetString(block);

}

return result;

}

public static List<byte[]> partitionOfString(String s, long length)

{//Разбиваем строку на блоки(байт-массивы)

List<byte[]> result = new List<byte[]>();

byte[] bytes = GetBytes(s);

List<byte> block = new List<byte>();

int count = 0;

foreach (byte x in bytes)

{

count++;

if (count < length)

{

block.Add(x);

}

else

{

result.Add(block.ToArray());

block.Clear();

block.Add(x);

count = 1;

}

}

if (block.Count > 0) result.Add(block.ToArray());

return result;

}

public static byte[] GetBytes(string str)

{

var utf8bytes = Encoding.Unicode.GetBytes(str);

var win1251Bytes = Encoding.Convert(

Encoding.Unicode, Encoding.GetEncoding("windows-1251"), utf8bytes);

//byte[] bytes = Encoding.ASCII.GetBytes(str);

return win1251Bytes;

}

public static string GetString(byte[] bytes)

{

var unicodeBytes = Encoding.Convert(

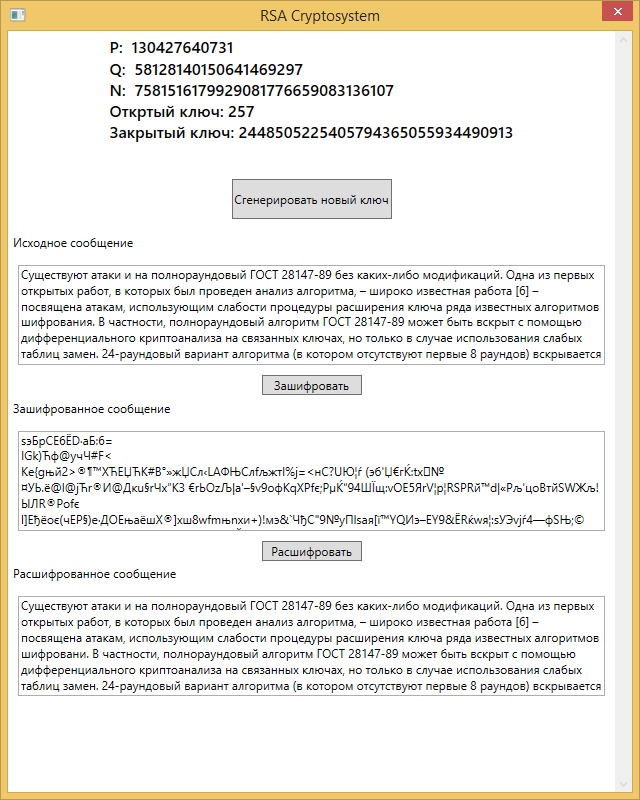
Encoding.GetEncoding("windows-1251"),Encoding.Unicode, bytes);

return Encoding.Unicode.GetString(unicodeBytes);

}

}

# Примеры работы программы

****

# RSA_screen_2RSA_screen_3RSA_screen_4RSA_screen_5

# Заключение

1. В ходе работы были изучены следующие дополнительные навыки:

* Обобщеный алгоритм Евклида
* Вероятностный тест на простоту Миллера-Рабина
* Работ с большими числами в .NET
* Реализация оконного интерфейса с помощью технологии WPF
* Базовые знания теории чисел такие как:
  + Кольцо целых чисел
  + Функция Эйлера
  + Теорема Ферма
  + НОД
  + Сравнение по модулю

1. Алгоритм RSA является криптостойким и это свойство зависит напрямую от длины модуля N. Чем длиннее число, тем дольше происходит процесс его факторизации.