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](#_Toc417410804)

[Описание алгоритма 4](#_Toc417410805)

[**1.** **Генерация открытого и закрытого ключей** 4](#_Toc417410806)

[**2.** **Шифрование** 4](#_Toc417410807)

[**3.** **Расшифровка** 4](#_Toc417410808)

[Блок-схема алгоритма 5](#_Toc417410809)

[Листинг программы, реализующей алгоритмы 6](#_Toc417410810)

[Класс RsaCrypt 6](#_Toc417410811)

[Класс TextManager 8](#_Toc417410812)

[Класс MainWindow 9](#_Toc417410813)

[Графический интерфейс 12](#_Toc417410814)

[Примеры работы программы 13](#_Toc417410815)

[Тестирование производительности 16](#_Toc417410816)

[Заключение 24](#_Toc417410817)

# 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 49 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 RsaCrypt

using System;

using System.Collections.Generic;

using System.Numerics;

using System.Security.Cryptography;

namespace RSA\_Project

{

public class RsaCrypt

{

private static readonly Random Rnd = new Random((int) DateTime.Now.Ticks);

private static readonly RandomNumberGenerator Rng = RandomNumberGenerator.Create();

private readonly int[] \_fermat = {17, 257, 65537}; //Числа ферма

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

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

private BigInteger \_priKey; //открытый и закрытый ключи

private BigInteger \_pubKey; //открытый и закрытый ключи

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

public RsaCrypt(BigInteger p, BigInteger q)

{

//Инициализация класса

\_p = p;

\_q = q;

\_n = \_p\*\_q;

BigInteger eulierFunction = (\_p - 1)\*(\_q - 1);

\_pubKey = \_fermat[Rnd.Next(0, \_fermat.Length)];

\_priKey = BuildPrivateKey(eulierFunction);

}

public RsaCrypt(int n)

{

GenerateKeys(n);

}

public long ModuleLength

{

//Длина модуля в байтах

get

{

long l = \_n.ToByteArray().Length;

return l;

}

}

public List<byte[]> EncryptMessage(IEnumerable<byte[]> p)

{

//Зашифровать строку

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

foreach (var block in p)

{

result.Add(Encrypt(block));

}

return result;

}

public List<byte[]> DecryptMessage(IEnumerable<byte[]> p)

{

//Разшифровать строку

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

foreach (var block in p)

{

result.Add(Decrypt(block));

}

return result;

}

private byte[] Encrypt(byte[] message)

{

//Шифруем сообщение открытым ключом

var val = new BigInteger(message);

val = BigInteger.ModPow(val, \_pubKey, \_n);

return val.ToByteArray();

}

private byte[] Decrypt(byte[] message)

{

//Расшифруем cообщение открытм ключом

var val = new BigInteger(message);

val = BigInteger.ModPow(val, \_priKey, \_n);

return val.ToByteArray();

}

private BigInteger BuildPrivateKey(BigInteger E)

{

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

BigInteger x, y;

BigInteger g = Gcd(\_pubKey, E, out x, out y);

return (x%E + E)%E;

}

private static 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;

}

#region Свойства

public BigInteger Module

{

//Возвращает значения модуля(N)

get { return \_n; }

}

public BigInteger PriKey

{

//значение закрытого ключа

get { return \_priKey; }

}

public BigInteger PubKey

{

//значение откртыго ключа

get { return \_pubKey; }

}

#endregion

#region Методы для генерации случайного ключа

public static BigInteger Random(int bytes)

{

if (bytes == 0) return new BigInteger(0);

var buffer = new byte[bytes];

Rng.GetBytes(buffer);

buffer[bytes - 1] = (byte) ((buffer[bytes - 1] & 127) | 64); // старший байт

return new BigInteger(buffer);

}

public static int NumberOfTests(BigInteger x)

{

return 2\*x.ToByteArray().Length + 100; // Чем больше тестов тем меньше вероятность ошибиться

}

public static bool IsPrimary(BigInteger x)

{

if (x < 2) return false; // отбрасываем отрицательные и единицу

int len = x.ToByteArray().Length;

int tests = NumberOfTests(x); // Чем больше тестов тем меньше вероятность ошибиться

BigInteger y = x - 1;

for (int i = 0; i < tests; i++)

{

BigInteger a = (Random((int) (len\*Rnd.NextDouble()))%y) + 1; // берём ненулевое

// проверяем выполнение малой теоремы Ферма

// если простое то

if (!(BigInteger.ModPow(a, y, x) - 1).IsZero) return false;

}

// признаём число простым

// хотя можем продолжать ошибаться

return true;

}

public static BigInteger GeneratePrimary(int bytes)

{

BigInteger x = Random(bytes) | 1; // Простые являются нечётными

while (!IsPrimary(x)) x += 2; // Движемся вперёд пока не встретим простое

return x;

}

public void GenerateKeys(int n)

{

var bits = (int) Math.Ceiling(n/Math.Log10(2));

int bits1 = bits/2;

int bits2 = bits - bits1;

int bytes1 = (bits1 + 7)/8;

int bytes2 = (bits2 + 7)/8;

\_p = GeneratePrimary(bytes1);

\_q = GeneratePrimary(bytes2);

\_n = \_p\*\_q;

BigInteger eulierFunction = (\_p - 1)\*(\_q - 1);

\_pubKey = \_fermat[Rnd.Next(0, \_fermat.Length)];

\_priKey = BuildPrivateKey(eulierFunction);

}

#endregion

}

}

## Класс TextManager

using System;

using System.Collections.Generic;

using System.Text;

namespace RSA\_Project

{

public static class TextManager

{

//Класс работающий с кодировками и разбиением текста

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

{

//Переобразует разбиение в строку

String result = String.Empty;

foreach (var block in p)

{

result += GetString(block);

}

return result;

}

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

{

//Разбиваем строку на блоки(байт-массивы)

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

byte[] bytes = GetBytes(s);

var 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)

{

byte[] utf8bytes = Encoding.Unicode.GetBytes(str);

byte[] win1251Bytes = Encoding.Convert(

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

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

return win1251Bytes;

}

public static string GetString(byte[] bytes)

{

byte[] unicodeBytes = Encoding.Convert(

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

return Encoding.Unicode.GetString(unicodeBytes);

}

}

}

## Класс MainWindow

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

namespace RSA\_Project

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private const int Len = 49; //Длина ключа

private static List<byte[]> \_partition; //Разбиение строки на байт массивы

private readonly RsaCrypt \_crypt; // Класс шифровальщика с открытым ключом

public MainWindow()

{

InitializeComponent();

\_crypt = new RsaCrypt(Len);

UpdateInfo();

}

private void UpdateInfo()

{

infoRSALabel.Content = String.Format("\nN: {0} \nОткртый ключ: {1} \nЗакрытый ключ: {2}\n",

\_crypt.Module, \_crypt.PubKey, \_crypt.PriKey);

}

#region События формы

private void generate\_Click(object sender, RoutedEventArgs e)

{

\_crypt.GenerateKeys(Len);

UpdateInfo();

}

private void encryptBtn\_Click(object sender, RoutedEventArgs e)

{

\_partition =

\_crypt.EncryptMessage(TextManager.PartitionOfString(messageBox.Text,

\_crypt.ModuleLength - 1));

encryptedBox.Text = TextManager.PartitionToString(\_partition);

}

private void decryptBtn\_Click(object sender, RoutedEventArgs e)

{

decryptedBox.Text = TextManager.PartitionToString(\_crypt.DecryptMessage(\_partition));

}

private void messageBox\_TextChanged(object sender, TextChangedEventArgs e)

{

encryptedBox.Text = "";

decryptedBox.Text = "";

}

#endregion

}

}

using System;

using System.Collections.Generic;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using RSA\_Project;

namespace ZI\_Lab\_Lab\_3\_9.UnitTest

{

[TestClass]

public class UnitTest1

{

private const int Len = 49; //Длина ключа

private readonly RsaCrypt \_crypt = new RsaCrypt(Len);

// Класс шифровальщика с открытым ключом

private readonly string[] \_texts =

{

"Так в 1923 году В. И. Ленин озаглавил свою замечательную статью о мерах, которые необходимо было принять для укрепления и улучшения советского государственного аппарата. Слова эти оказались столь многозначительными и вескими, были так удачно найдены, что скоро из простого заглавия превратились в настоящее крылатое слово со значением: качество может быть важнее количества.",

"В огромных старорусских семьях было принято, чтобы их члены чередовались понедельно на домашних работах: эту неделю Федор колет дрова, Иван таскает воду, следующую - наоборот. Так же поочередно мололи муку на домашнем жернове: очень нелегкая работа. Болтунам же, отлынивавшим от всякого труда, говорили с насмешкой: \"Мели, Емеля, твоя неделя!\" - играя на том, что выражение \"языком молоть\" комически сопоставлялось тут именно с представлением о самой тяжелой работе на жернове.",

"Выражение это, так сказать, царского происхождения. Автором его был царь Иван IV, прозванный в народе Грозным за массовые казни и убийства. Для усиления своей самодержавной власти, что невозможно было без ослабления князей, бояр и духовенства, Иван Грозный ввел опричнину, наводившую ужас на все государство Российское.Не мог примириться с разгулом опричников и митрополит Московский Филипп.",

"Что это за выражение? При чем тут решето? Оказывается, это сокращение старинной русской поговорки, которая гласила: \"чудеса в решете - дыр много, а выскочить некуда\". До нас дошла только первая часть фразы, которая как бы вобрала в себя смысл всего высказывания.",

"В греческой и римской мифологии Цербер - чудовищный трехглавый пес со змеиным хвостом, охранявший вход в подземное царство Аида, который был укрощен Гераклом. Отсюда слово \"цербер\" употребляется в значении: злой, свирепый надсмотрщик.",

"Когда великий титан Прометей похитил с Олимпа и передал людям огонь богов, отец богов Зевс страшно покарал смельчака, но было поздно. Обладая божественным пламенем, люди перестали подчиняться небожителям, научились разным наукам, вышли из своего жалкого состояния. Еще немного - и они завоевали бы себе полное счастье...",

"По древнегреческому мифу, Юпитеру (греч. Зевс) приглянулась дочь финикийского царя Европа. Юпитер превратился в быка и похитил ее. Quod licet Jovi, non licet bovi - пословица говорит о нескромной или безосновательной претензии."

};

private string ConvertStringToHex(string asciiString)

{

string hex = "";

foreach (char c in asciiString)

{

int tmp = c;

hex += String.Format("{0:x2}", Convert.ToUInt32(tmp.ToString()));

}

return hex;

}

[TestMethod]

public void TestMethod1()

{

for (int i = 0; i < \_texts.Length; i++)

{

string text = \_texts[i];

Console.WriteLine(@"Тест #: " + i);

\_crypt.GenerateKeys(Len);

Console.WriteLine(@"N: {0}", \_crypt.Module);

Console.WriteLine(@"Открытый ключ: {0}", \_crypt.PubKey);

Console.WriteLine(@"Закрытый ключ: {0}", \_crypt.PriKey);

Console.WriteLine(@"Исходный текст: " + text);

List<byte[]> partition =

\_crypt.EncryptMessage(TextManager.PartitionOfString(text, \_crypt.ModuleLength - 1));

string cipher = TextManager.PartitionToString(partition);

Console.WriteLine(@"Шифрованный текст: " + ConvertStringToHex(cipher));

string plain = TextManager.PartitionToString(\_crypt.DecryptMessage(partition));

Console.WriteLine(@"Расшифрованный текст: " + plain);

//Assert.IsTrue(String.CompareOrdinal(text.Trim(), cipher.Trim()) != 0);

//Assert.IsTrue(String.CompareOrdinal(text.Trim(), plain.Trim()) == 0);

Console.WriteLine();

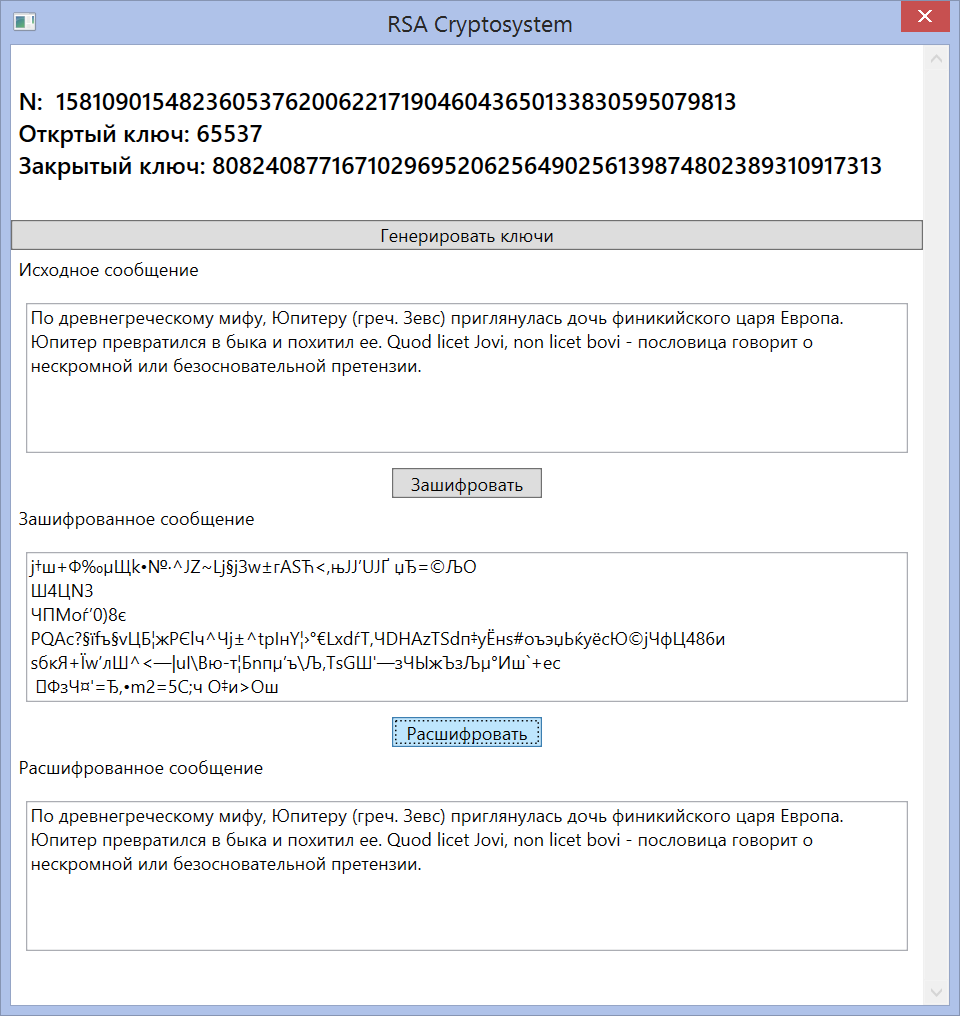
}

}

}

}

## Графический интерфейс



# Примеры работы программы

Тест #: 0

N: 10247412618064917916064552210343464902652677646493611

Открытый ключ: 65537

Закрытый ключ: 8009265247648493071312000052812410727596140142712481

Исходный текст: Так в 1923 году В. И. Ленин озаглавил свою замечательную статью о мерах, которые необходимо было принять для укрепления и улучшения советского государственного аппарата. Слова эти оказались столь многозначительными и вескими, были так удачно найдены, что скоро из простого заглавия превратились в настоящее крылатое слово со значением: качество может быть важнее количества.

Шифрованный текст: 

Расшифрованный текст: Так в 1923 году В. И. Ленин озаглавил свою замечательную статью о мерах, которые необходимо было принять для укрепления и улучшения советского государственного аппарата. Слова эти оказались столь многозначительными и вескими, были так удачно найдены, что скоро из простого заглавия превратились в настоящее крылатое слово со значением: качество может быть важнее количества.

Тест #: 1

N: 17335771606492691098688204605085327950040429731364039

Открытый ключ: 65537

Закрытый ключ: 16910425302218155381662850226168058413358948724752113

Исходный текст: В огромных старорусских семьях было принято, чтобы их члены чередовались понедельно на домашних работах: эту неделю Федор колет дрова, Иван таскает воду, следующую - наоборот. Так же поочередно мололи муку на домашнем жернове: очень нелегкая работа. Болтунам же, отлынивавшим от всякого труда, говорили с насмешкой: "Мели, Емеля, твоя неделя!" - играя на том, что выражение "языком молоть" комически сопоставлялось тут именно с представлением о самой тяжелой работе на жернове.

Шифрованный текст: 

Расшифрованный текст: В огромных старорусских семьях было принято, чтобы их члены чередовались понедельно на домашних работах: эту неделю Федор колет дрова, Иван таскает воду, следующую - наоборот. Так же поочередно мололи муку на домашнем жернове: очень нелегкая работа. Болтунам же, отлынивавшим от всякого труда, говорили с насмешкой: "Мели, Емеля, твоя неделя!" - играя на том, что выражение "языком молоть" комически сопоставлялось тут именно с представлением о самой тяжелой работе на жернове.

Тест #: 2

N: 9767889172004967161011862018590318956813019513684801

Открытый ключ: 17

Закрытый ключ: 6894980592003506231302490695515069471509896017282353

Исходный текст: Выражение это, так сказать, царского происхождения. Автором его был царь Иван IV, прозванный в народе Грозным за массовые казни и убийства. Для усиления своей самодержавной власти, что невозможно было без ослабления князей, бояр и духовенства, Иван Грозный ввел опричнину, наводившую ужас на все государство Российское.Не мог примириться с разгулом опричников и митрополит Московский Филипп.

Шифрованный текст: 

Расшифрованный текст: Выражение это, так сказать, царского происхождения. Автором его был царь Иван IV, прозванный в народе Грозным за массовые казни и убийства. Для усиления своей самодержавной власти, что невозможно было без ослабления князей, бояр и духовенства, Иван Грозный ввел опричнину, наводившую ужас на все государство Российское.Не мог примириться с разгулом опричников и митрополит Московский Филипп.

Тест #: 3

N: 14719416524974645300051781891026538997570117508175273

Открытый ключ: 17

Закрытый ключ: 8658480308808614882383400969568598285368158395124153

Исходный текст: Что это за выражение? При чем тут решето? Оказывается, это сокращение старинной русской поговорки, которая гласила: "чудеса в решете - дыр много, а выскочить некуда". До нас дошла только первая часть фразы, которая как бы вобрала в себя смысл всего высказывания.

Шифрованный текст: 

Расшифрованный текст: Что это за выражение? При чем тут решето? Оказывается, это сокращение старинной русской поговорки, которая гласила: "чудеса в решете - дыр много, а выскочить некуда". До нас дошла только первая часть фразы, которая как бы вобрала в себя смысл всего высказывания.

Тест #: 4

N: 14151804955194623896921457829352897201049796417238379

Открытый ключ: 17

Закрытый ключ: 3329836460045793858099166491912094576288933068999665

Исходный текст: В греческой и римской мифологии Цербер - чудовищный трехглавый пес со змеиным хвостом, охранявший вход в подземное царство Аида, который был укрощен Гераклом. Отсюда слово "цербер" употребляется в значении: злой, свирепый надсмотрщик.

Шифрованный текст: 

Расшифрованный текст: В греческой и римской мифологии Цербер - чудовищный трехглавый пес со змеиным хвостом, охранявший вход в подземное царство Аида, который был укрощен Гераклом. Отсюда слово "цербер" употребляется в значении: злой, свирепый надсмотрщик.

Тест #: 5

N: 22134860724932263986886968534914355202365601656172789

Открытый ключ: 257

Закрытый ключ: 4478648862632209055712538320364019757452864257724657

Исходный текст: Когда великий титан Прометей похитил с Олимпа и передал людям огонь богов, отец богов Зевс страшно покарал смельчака, но было поздно. Обладая божественным пламенем, люди перестали подчиняться небожителям, научились разным наукам, вышли из своего жалкого состояния. Еще немного - и они завоевали бы себе полное счастье...

Шифрованный текст: 

Расшифрованный текст: Когда великий титан Прометей похитил с Олимпа и передал людм огонь богов, отец богов Зевс страшно покарал смельчака, но было поздно. Обладая божественным пламенем, люди перестали подчиняться небожителям, научились разным наукам, вышли из своего жалкого состония. Еще немного - и они завоевали бы себе полное счастье...

Тест #: 6

N: 19443902127891253669714988528652058510525031485576927

Открытый ключ: 17

Закрытый ключ: 13725107384393826119798815234811007880187663607336209

Исходный текст: По древнегреческому мифу, Юпитеру (греч. Зевс) приглянулась дочь финикийского царя Европа. Юпитер превратился в быка и похитил ее. Quod licet Jovi, non licet bovi - пословица говорит о нескромной или безосновательной претензии.

Шифрованный текст: 

Расшифрованный текст: По древнегреческому мифу, Юпитеру (греч. Зевс) приглянулась дочь финикийского царя Европа. Юпитер превратился в быка и похитил ее. Quod licet Jovi, non licet bovi - пословица говорит о нескромной или безосновательной претензии.

# Тестирование производительности

Windows 8.1 Профессиональная

Процессор: Intel Core Duo E8600 @ 3.33GHz 2.67GHz

Установленная память: 8,00 ГБ

Тип системы: 64-разрядная операционная система, процессор x64

Время измеряется в миллисекундах (1s == 1000ms)

N – количество десятичных разрядов

TestMethod1 – измерение среднего времени генерации ключа

TestMethod2 – измерение среднего времени генерации одного простого числа

TestMethod3 – измерение среднего времени одной итерации проверки числа на простоту

Для построения графиков использовалась программа Gnuplot http://www.gnuplot.info

using System;

using System.Numerics;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using RSA\_Project;

namespace ZI\_Lab\_Lab\_3\_9.UnitTest

{

[TestClass]

public class UnitTest2

{

[TestMethod]

public void TestMethod1()

{

const int count = 20;

Console.WriteLine(@"# N GenerateKeys");

for (int n = 20; n < 50; n++)

{

var rsa = new RsaCrypt(n);

DateTime t = DateTime.Now;

for (int i = 0; i < count; i++)

rsa.GenerateKeys(n);

var ts = new TimeSpan(DateTime.Now.Ticks - t.Ticks);

Console.WriteLine(@"{0} {1}", n, ts.TotalMilliseconds/count);

}

}

[TestMethod]

public void TestMethod2()

{

const int count = 20;

Console.WriteLine(@"# N GeneratePrimary");

for (int n = 20; n < 50; n++)

{

var bits = (int) Math.Ceiling(n/Math.Log10(2));

int bytes = (bits + 7)/8;

DateTime t = DateTime.Now;

for (int i = 0; i < count; i++)

RsaCrypt.GeneratePrimary(bytes);

var ts = new TimeSpan(DateTime.Now.Ticks - t.Ticks);

Console.WriteLine(@"{0} {1}", n, ts.TotalMilliseconds/count);

}

}

[TestMethod]

public void TestMethod3()

{

const int count = 20;

Console.WriteLine(@"# N IsPrimary");

for (int n = 20; n < 50; n++)

{

var bits = (int) Math.Ceiling(n/Math.Log10(2));

int bytes = (bits + 7)/8;

DateTime t = DateTime.Now;

int total = 0;

for (int i = 0; i < count; i++)

{

BigInteger x = RsaCrypt.Random(bytes) | 1; // Простые являются нечётными

total += RsaCrypt.NumberOfTests(x);

RsaCrypt.IsPrimary(x);

}

var ts = new TimeSpan(DateTime.Now.Ticks - t.Ticks);

Console.WriteLine(@"{0} {1}", n, ts.TotalMilliseconds / count);

}

}

}

}

**# N GenerateKeys**

**20 4.25026**

**21 4.15027**

**22 4.75027**

**23 5.30031**

**24 4.650265**

**25 6.30035**

**26 6.700385**

**27 5.6503**

**28 6.50036**

**29 6.150355**

**30 7.600445**

**31 8.900525**

**32 8.3505**

**33 8.750485**

**34 8.4005**

**35 9.25053**

**36 9.050505**

**37 9.050535**

**38 8.400485**

**39 14.35085**

**40 13.850805**

**41 14.800885**

**42 14.20084**

**43 15.850935**

**44 17.451005**

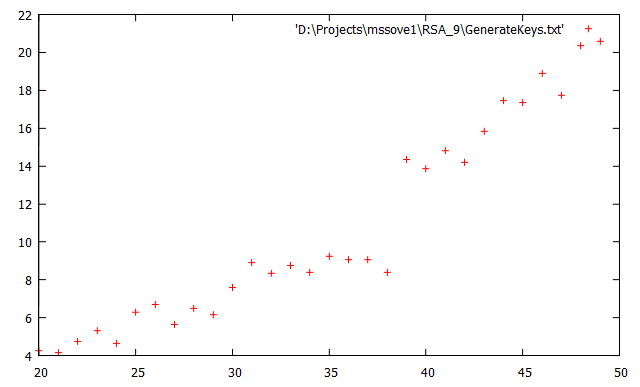
**45 17.35098**

**46 18.90112**

**47 17.75102**

**48 20.351195**

**49 20.601175**



**# N GeneratePrimary**

**20 7.300415**

**21 7.85046**

**22 11.20067**

**23 9.60053**

**24 9.15052**

**25 10.05057**

**26 10.05063**

**27 12.25071**

**28 12.850725**

**29 16.80095**

**30 16.80097**

**31 20.7012**

**32 20.35119**

**33 20.4012**

**34 24.05138**

**35 21.55124**

**36 23.101365**

**37 26.1515**

**38 23.601375**

**39 41.152365**

**40 37.35219**

**41 38.90222**

**42 39.452295**

**43 36.90212**

**44 40.752375**

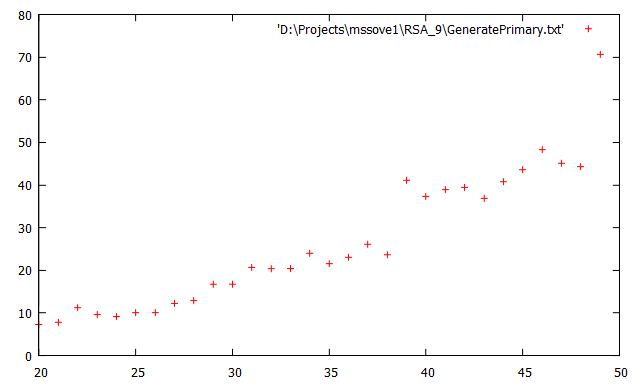
**45 43.552515**

**46 48.302775**

**47 45.15261**

**48 44.302585**

**49 70.7041**



**# N IsPrimary**

**20 0.299985**

**21 0.05**

**22 1.15009**

**23 0.050005**

**24 0.049985**

**25 0.45004**

**26 0.500025**

**27 0.049985**

**28 1.0501**

**29 0.09999**

**30 0.150005**

**31 0.149995**

**32 1.6501**

**33 0.85007**

**34 0.149995**

**35 0.90006**

**36 0.14999**

**37 0.15001**

**38 0.150035**

**39 0.199995**

**40 0.20001**

**41 1.55008**

**42 1.550095**

**43 0.25001**

**44 0.200025**

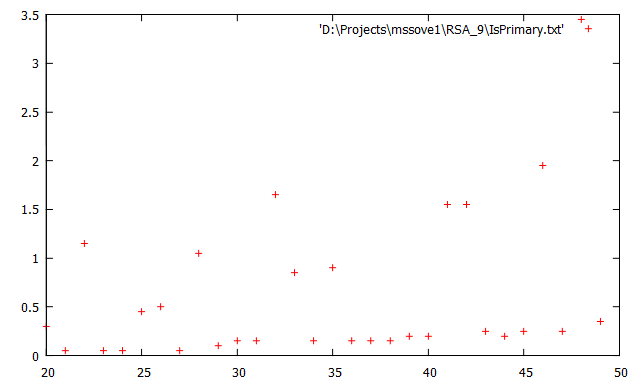
**45 0.25**

**46 1.95013**

**47 0.25001**

**48 3.450175**

**49 0.35005**



# Заключение

1. В ходе работы были изучены следующие дополнительные навыки:

* Обобщённый 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.