**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине: «Основы защиты информации»

на тему: **«**Асимметричные алгоритмы шифрования**»**

Выполнила: студент гр. ИТП-31

Расшивалов Н.И.

Принял: доцент

Кравченко О.А.

Гомель 2021

**Цель работы:** освоить навыки применения асимметричных методов шифрования данных.

**Задание**

Разработать приложение, выполняющее шифрование и дешифрование текста с помощью алгоритма RSA. Открытый и закрытый ключ подобрать самостоятельно.

**Краткие теоретические сведения:**

**Алгоритм *RSA*:**

Первым этапом любого асимметричного алгоритма является создание пары ключей: открытого и закрытого и распространение открытого ключа «по всему миру». Для алгоритма *RSA* этап создания ключей состоит из следующих операций:

1. Выбираются два простых числа *p* и *q*

2. Вычисляется их произведение *n* = *p*\**q*

3. Выбирается произвольное число *e* (*e*<*n*), такое, что НОД(*e*,(*p*-1)(*q*-1)) = 1, то есть e должно быть взаимно простым с числом (*p*-1)(*q*-1).

4. Методом Евклида решается в целых числах уравнение *e*\**d*+(*p*-1)(*q*-1)\**y*=1. Здесь неизвестными являются переменные *d* и *y* – метод Евклида как раз и находит множество пар (*d,y*), каждая из которых является решением уравнения в целых числах.

5. Два числа (*e,n*) – публикуются как открытый ключ.

6. Число *d* хранится в строжайшем секрете – это и есть закрытый ключ, который позволит читать все послания, зашифрованные с помощью пары чисел (*e,n*).

Отправитель разбивает свое сообщение на блоки, равные *k*=[*log*2(*n*)] бит, где квадратные скобки обозначают взятие целой части от дробного числа.

Подобный блок может быть интерпретирован как число из диапазона (0;2k-1). Для каждого такого числа (назовем его *mi*) вычисляется выражение *ci=((mi)e)mod n*. Блоки *ci* и есть зашифрованное сообщение. Их можно передавать по открытому каналу, поскольку операция возведения в степень по модулю простого числа, является необратимой математической задачей. Обратная ей задача носит название «логарифмирование в конечном поле» и является на несколько порядков более сложной задачей. То есть даже если злоумышленник знает числа *e* и *n*, то по *ci* прочесть исходные сообщения *mi* он не может никак, кроме как полным перебором *mi*.

На приемной стороне процесс дешифрования осуществляется с помощью хранимого в секрете числа *d*.

Достаточно давно была доказана теорема Эйлера, частный случай которой утверждает, что если число n представимо в виде двух простых чисел *p* и *q*, то для любого *x* имеет место равенство *(x(p- 1)(q-1))mod n* = 1.

**Ход работы**

На рисунке 1 изображен результат выполнения шифрования с использованием *RSA*.

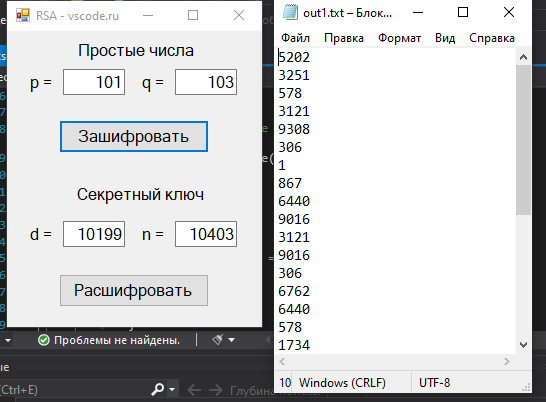


Рисунок 1 – Результат шифровки по *RSA*

На рисунке 2 изображен результат выполнения дешифрования с использованием *RSA*.

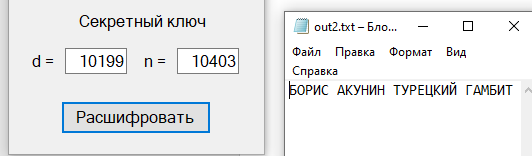


Рисунок 2 – Результат расшифровки по *RSA*

**Вывод**: в ходе выполнения данной лабораторной работы освоены освоить навыки применения асимметричных методов шифрования данных.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг программы**

**Класс Protect\_inf\_LR1:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Numerics;

namespace protect\_inf\_LR1

{

public partial class Form1 : Form

{

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' };

public Form1()

{

InitializeComponent();

}

//зашифровать

private void buttonEncrypt\_Click(object sender, EventArgs e)

{

if ((textBox\_p.Text.Length > 0) && (textBox\_q.Text.Length > 0))

{

long p = Convert.ToInt64(textBox\_p.Text);

long q = Convert.ToInt64(textBox\_q.Text);

if (IsTheNumberSimple(p) && IsTheNumberSimple(q))

{

string s = "";

StreamReader sr = new StreamReader("in.txt");

while (!sr.EndOfStream)

{

s += sr.ReadLine();

}

sr.Close();

s = s.ToUpper();

long n = p \* q;

long m = (p - 1) \* (q - 1);

long d = Calculate\_d(m);

long e\_ = Calculate\_e(d, m);

List<string> result = RSA\_Endoce(s, e\_, n);

StreamWriter sw = new StreamWriter("out1.txt");

foreach (string item in result)

sw.WriteLine(item);

sw.Close();

textBox\_d.Text = d.ToString();

textBox\_n.Text = n.ToString();

Process.Start("out1.txt");

}

else

MessageBox.Show("p или q - не простые числа!");

}

else

MessageBox.Show("Введите p и q!");

}

//расшифровать

private void buttonDecipher\_Click(object sender, EventArgs e)

{

if ((textBox\_d.Text.Length > 0) && (textBox\_n.Text.Length > 0))

{

long d = Convert.ToInt64(textBox\_d.Text);

long n = Convert.ToInt64(textBox\_n.Text);

List<string> input = new List<string>();

StreamReader sr = new StreamReader("out1.txt");

while (!sr.EndOfStream)

{

input.Add(sr.ReadLine());

}

sr.Close();

string result = RSA\_Dedoce(input, d, n);

StreamWriter sw = new StreamWriter("out2.txt");

sw.WriteLine(result);

sw.Close();

Process.Start("out2.txt");

}

else

MessageBox.Show("Введите секретный ключ!");

}

//проверка: простое ли число?

private bool IsTheNumberSimple(long n)

{

if (n < 2)

return false;

if (n == 2)

return true;

for (long i = 2; i < n; i++)

if (n % i == 0)

return false;

return true;

}

//зашифровать

private List<string> RSA\_Endoce(string s, long e, long n)

{

List<string> result = new List<string>();

BigInteger bi;

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

{

int index = Array.IndexOf(characters, s[i]);

bi = new BigInteger(index);

bi = BigInteger.Pow(bi, (int)e);

BigInteger n\_ = new BigInteger((int)n);

bi = bi % n\_;

result.Add(bi.ToString());

}

return result;

}

//расшифровать

private string RSA\_Dedoce(List<string> input, long d, long n)

{

string result = "";

BigInteger bi;

foreach (string item in input)

{

bi = new BigInteger(Convert.ToDouble(item));

bi = BigInteger.Pow(bi, (int)d);

BigInteger n\_ = new BigInteger((int)n);

bi = bi % n\_;

int index = Convert.ToInt32(bi.ToString());

result += characters[index].ToString();

}

return result;

}

//вычисление параметра d. d должно быть взаимно простым с m

private long Calculate\_d(long m)

{

long d = m - 1;

for (long i = 2; i <= m; i++)

if ((m % i == 0) && (d % i == 0)) //если имеют общие делители

{

d--;

i = 1;

}

return d;

}

//вычисление параметра e

private long Calculate\_e(long d, long m)

{

long e = 10;

while (true)

{

if ((e \* d) % m == 1)

break;

else

e++;

}

return e;

}

}

}