Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Отчет**

«Исследование асимметричных шифров»

Выполнил:

студент 3 курса 4 группы

специальности ПОИТ

Чернявский А. Л.

Минск 2020

Цель: изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров.

**Задачи**:

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров.

2. Разработать приложение для реализации указанных преподавателем методов генерации ключевой информации и ее использования для асимметричного зашифрования/расшифрования.

3. Выполнить анализ криптостойкости асимметричных шифров.

4. Оценить скорость зашифрования/расшифрования реализованных шифров.

5. Результаты выполнения лабораторной работы оформить в виде описания

разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Практическое задание**:

11. Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться доступными библиотеками либо программными кодами.

В основе вычислений – кодировочные таблицы Base64 и ASCII.

Приложение должно реализовывать следующие операции:

• генерация сверхвозрастающей последовательности (тайного ключа);

старший член последовательности – 100-битное число; в простейшем

случае принимается z = 6 (для кодировки Base64) и z = 8 (для кодировки

ASCII);

• вычисление нормальной последовательности (открытого ключа);

• зашифрование сообщения, состоящего из собственных фамилии, имени и отчества;

• расшифрование сообщения;

• оценка времени выполнения операций зашифрования и расшифрования.

2. Проанализировать время выполнения операций зашифрования/расшифрования при увеличении числа членов ключевой последовательности: при использовании разных таблиц колировки.

3. Результаты оформить в виде отчета по установленным правилам

**Результат выполнения работы приложения:**

|  |
| --- |
| Вектор :  1  14934  135756810  1625181830713  18931192898950332  298151957482936109333  3991238402747480480701782  79319428375482881164066363344  n = 265252859812191058636308479999999  a = 75838465738475849374657348573930  e (Открытый ключ):  75838465738475849374657348573930  207188800154705242731942483074889  22513036897380226428699624777540  191348826807245437042532760975459  87758055018529545897554211268899  116582848056986360774353980055872  98018603051970588211732991176788  143939256326619781825194335236474  Исходная строка:Charniauski Aliaksei  Результат зашифрования:  449146659533295612768869809488151  0  317459892070615015058196319121328  0  373641093378705250985836443088903  0  519069266911301494414907860004676  0  532061343179571964044283290353988  0  461399148397234796883390654357802  0  373641093378705250985836443088903  0  681572768242937048802723184120234  0  663008523237921276240102195241150  0  559417751449205385095123645534590  0  461399148397234796883390654357802  0  22513036897380226428699624777540  0  351128056481325024557136818311363  0  434042740127601375832550299177200  0  461399148397234796883390654357802  0  373641093378705250985836443088903  0  559417751449205385095123645534590  0  663008523237921276240102195241150  0  490223941435691611760190423144775  0  461399148397234796883390654357802  0  0  Время зашифрования: 770004  Расшифрование:  Charniauski Aliaksei  Время расшифрования: 40015 |

**Листинги**

using System;

using System.Linq;

using System.Text;

using System.Numerics;

namespace Lab9

{

class Program

{

static void Main(string[] args)

{

BigInteger[] mass = Vector100();

Console.WriteLine();

BigInteger W = 0; // Summary

for (int i = 0; i < 8; i++)

{

W += mass[i];

}

//Console.WriteLine("w = " + W);

string Qs = "265252859812191058636308479999999";

BigInteger Q = BigInteger.Parse(Qs);

Console.WriteLine("n = " + Q); // > W

string Rs = "75838465738475849374657348573930";

BigInteger R = BigInteger.Parse(Rs);

Console.WriteLine("a = " + R); // < Q

BigInteger[] B = new BigInteger[8];

Console.WriteLine();

Console.WriteLine("e (Открытый ключ):");

for (int i = 0; i < 8; i++) // Public Key

{

B[i] = (mass[i] \* R) % Q; // e[i] = (d[i] \* a) mod n

Console.WriteLine(B[i]);

}

Console.WriteLine();

string[] set = toBin();

BigInteger[] crypt = new BigInteger[set.Length];

Console.WriteLine();

Console.WriteLine("Результат зашифрования:");

long ellapledTicks = DateTime.Now.Ticks;

for (int i = 0; i < set.Length; i++) // calculate by block

{

char[] b = set[i].ToCharArray();

for (int j = 0; j < b.Length; j++)

{

crypt[i] += BigInteger.Parse(b[j].ToString()) \* B[j];

}

Console.WriteLine(crypt[i]);

}

ellapledTicks = DateTime.Now.Ticks - ellapledTicks;

Console.WriteLine();

Console.WriteLine("Время зашифрования: " + ellapledTicks);

string modRs = "198053286779263733527432885858298";

BigInteger modR = BigInteger.Parse(modRs);

Console.WriteLine();

//Console.WriteLine("mod r = " + modR);

BigInteger[] startEncrypt = new BigInteger[crypt.Length];

long ellapledTicks2 = DateTime.Now.Ticks;

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

{

startEncrypt[i] = (crypt[i] \* modR) % Q;

}

string[] encrypt = new string[startEncrypt.Length];

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

{

string currentWord = "";

BigInteger toNull = startEncrypt[i];

for (int j = mass.Length - 1; j >= 0; j--)

{

if (toNull >= mass[j])

{

toNull -= mass[j];

currentWord += "1";

}

else

{

currentWord += "0";

}

//if(j==0 && toNull != 0)

//{

// j = mass.Length - 1;

//}

}

encrypt[i] = new string(currentWord.ToCharArray().Reverse().ToArray());

}

ellapledTicks2 = DateTime.Now.Ticks - ellapledTicks2;

string bicaryEncrypt = "";

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

{

bicaryEncrypt += encrypt[i];

}

var stringArray = Enumerable.Range(0, bicaryEncrypt.Length / 8).Select(i => Convert.ToByte(bicaryEncrypt.Substring(i \* 8, 8), 2)).ToArray();

var result = Encoding.Unicode.GetString(stringArray);

result = result.Remove(result.Length - 1);

Console.WriteLine();

Console.WriteLine("Расшифрование:");

Console.WriteLine(result);

Console.WriteLine();

Console.WriteLine("Время расшифрования: " + ellapledTicks2);

Console.ReadLine();

}

static BigInteger[] Vector100()

{

Random rnd = new Random();

BigInteger[] mass = new BigInteger[8];

mass[0] = rnd.Next(1, 9);

Console.WriteLine("Вектор :");

Console.WriteLine(mass[0]);

for (int i = 1; i < 8; i++)//100 bit = 30 symbls

{

for (int j = 0; j < mass.Length; j++)

{

BigInteger coef = rnd.Next(75, 150);

mass[i] += mass[j] \* coef;

}

Console.WriteLine(mass[i]);

}

return mass;

}

static string[] toBin()

{

string str = "Charniauski Aliaksei";

Console.WriteLine("Исходная строка:" + str);

StringBuilder sb = new StringBuilder();

foreach (byte b in Encoding.Unicode.GetBytes(str))

sb.Append(Convert.ToString(b, 2).PadLeft(8, '0')).Append(' ');

string binaryStr = sb.ToString();

string[] set = binaryStr.Split(' ');

Console.WriteLine();

//Console.WriteLine("Бинарный формат:");

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

//{

// Console.WriteLine(set[i]);

//}

return set;

}

}

}