Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**АСИММЕТРИЧНЫЕ АЛГОРИТМЫ ШИФРОВАНИЯ**

**Отчет по лабораторной работе №3**

**По дисциплине**

**«Информационная безопасность»**

Студент гр. 431-3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.П. Андреев

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Проверил: старший преподаватель кафедры АСУ.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Я.В.Яблонский

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Томск 2024

# Цель работы

Познакомиться и научиться работать с асимметричными алгоритмами шифрования.

# Задание на лабораторную работу

Вариант 9. Алгоритм Эль-Гамаля. Параметры: P = 9437, g = 29. Пользуясь алгоритмом, напишите программу, которая позволит зашифровать произвольный открытый текст, предварительно закодировав его согласно прилагаемым таблицам 2.1 – 2.3 и расшифровать его. Зашифрованный текст должен сохраняться в файле для пересылки своему другу. При написании программы реализуйте алгоритм быстрого возведения в степень и алгоритмы Евклида.

Таблица 2.1 – Кодировка русского алфавита

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| А | Б | В | Г | Д | Е | Ж | З | И | Й |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|  |  |  |  |  |  |  |  |  |  |
| К | Л | М | Н | О | П | Р | С | Т | У |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|  |  |  |  |  |  |  |  |  |  |
| Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
|  |  |  |  |  |  |  |  |  |  |
| Ю | Я |  |  |  |  |  |  |  |  |
| 40 | 41 |  |  |  |  |  |  |  |  |

Таблица 2.2 – Кодировка латинского алфавита

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J |
| 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 |
|  |  |  |  |  |  |  |  |  |  |
| K | L | M | N | O | P | Q | R | S | T |
| 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 |
|  |  |  |  |  |  |  |  |  |  |
| U | V | W | X | Y | Z |  |  |  |  |
| 62 | 63 | 64 | 65 | 66 | 67 |  |  |  |  |

Таблица 2.3 – Дополнительные символы

|  |  |  |
| --- | --- | --- |
| Пробел | Запятая | Точка |
| 68 | 69 | 70 |

# Описание алгоритма шифрования

Алгоритм Эль-Гамаля — это криптографический алгоритм, основанный на сложности задачи дискретного логарифма. Он используется для обеспечения конфиденциальности и цифровой подписи. Генерация ключа начинается с выбора параметров. Выбирается большое простое число p и генератор группы g. Генерируется случайное число x (секретный ключ) в диапазоне [1, p−2]. Вычисляется . Открытый ключ состоит из пары (p, g, y). Далее происходит шифрование. Генерируем случайное число k в диапазоне [1, p−2]. Шифрованное сообщение состоит из пары (c1, c2). Вычисляется и вычисляется .

Для расшифровки исходного сообщения восстанавливаем полученную пару как . .

# Листинг программы

using System;

using System.Numerics;

using System.IO;

namespace ИБ\_2

{

class Program

{

static int p = 9437; // Простое число

static int g = 29; // Генератор

static int[] Keys=new int[2]; //Ключи

static char[] ABC = new char[] { 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р',

'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь', 'Э', 'Ю', 'Я',

'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q',

'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',

' ', ',', '.'};

static int[] ABC123 = new int[] { 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,

27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,

42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,

59, 60, 61, 62, 63, 64, 65, 66, 67,

68, 69, 70};

static int[] Coding(char[] \_text)

{

int[] \_EncodText =new int[\_text.Length];

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

{

for (int j = 0; j < 61; j++)

{

if (\_text[i] == ABC[j])

{

\_EncodText[i] = \_EncodText[i] \* 100 + ABC123[j];

break;

}

if (\_text[i].ToString() == ABC[j].ToString().ToLower())

{

\_EncodText[i] = \_EncodText[i] \* 100 + ABC123[j];

break;

}

}

}

return \_EncodText;

}

static char Encoding(int \_text)

{

char letter=' ';

for (int i = 0; i < 61; i++)

{

if (\_text == ABC123[i])

{

letter = ABC[i];

break;

}

}

return letter;

}

static void CreatKeys()

{

Random rnd = new Random();

Keys[0] = rnd.Next(1, (int)p - 1); //Секретный ключ

Keys[1] = FastExponentiation(g, Keys[0], p); // Открытый ключ

}

static int[] Encrypt(int m)

{

int[] \_Blocks = new int[2];

// Генерируем случайное число k

Random rnd = new Random();

int k = rnd.Next(1, (int)p - 1);

// Вычисляем c1 и c2

\_Blocks[0] = FastExponentiation(g, k, p);

\_Blocks[1] = (m \* FastExponentiation(Keys[1], k, p)) % p;

return \_Blocks;

}

static int Decrypt(int c1, int c2)

{

// Вычисляем s

int s = FastExponentiation(c1, Keys[0], p);

// Находим обратное значение s

int sInverse = ModularInverse(s, p);

// Расшифровываем сообщение

return (c2 \* sInverse) % p;

}

static int FastExponentiation(int baseValue, int exponent, int modulus)

{

int result = 1;

baseValue = baseValue % modulus;

while (exponent > 0)

{

if ((exponent & 1) == 1) // Если exponent нечётный

result = (result \* baseValue) % modulus;

exponent >>= 1; // Делим exponent на 2

baseValue = (baseValue \* baseValue) % modulus; // Удваиваем основание

}

return result;

}

static int ModularInverse(int a, int m)

{

int m0 = m, t, q;

int x0 = 0, x1 = 1;

if (m == 1)

return 0;

while (a > 1)

{

// q — целая часть a / m

q = a / m;

t = m;

// m — остаток, теперь a — старое значение m

m = a % m;

a = t;

t = x0;

x0 = x1 - q \* x0;

x1 = t;

}

if (x1 < 0)

x1 += m0;

return x1;

}

static void Main(string[] args)

{

Console.WriteLine("Программа начала работу");

Console.WriteLine("Кодирование входного сообщения");

//Кодирование входного сообщения

char[] text;

using (FileStream InFile = new FileStream("IN.txt", FileMode.Open))

using (BinaryReader reader = new BinaryReader(InFile))

{

text = reader.ReadChars(reader.PeekChar());

}

int[] EncodText = Coding(text);

//Кодирование входного сообщения

Console.WriteLine("Генерация ключа");

//Генерация ключей

CreatKeys();

//Генерация ключей

Console.WriteLine("Шифрование сообщения");

//Шифрование сообщения

int[] Block1 = new int[EncodText.Length];

int[] Block2 = new int[EncodText.Length];

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

{

int[] Blocks = Encrypt(EncodText[i]);

Block1[i] = Blocks[0];

Block2[i] = Blocks[1];

}

//Шифрование сообщения

Console.WriteLine("Запись шифра в файл");

//Запись шифра в файл

using (FileStream EncryptionFile = new FileStream("ENCRYPTION.txt", FileMode.Create))

using (StreamWriter encryptedWriter = new StreamWriter(EncryptionFile))

{

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

{

string encrypted;

encrypted = Block1[i].ToString() + Block2[i].ToString();

encryptedWriter.Write(encrypted);

}

}

//Запись шифра в файл

Console.WriteLine("Расшифровка сообщения");

//Расшифровка сообщения

string decryptedText = "";

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

{

char decryptedMessage = Encoding(Decrypt(Block1[i], Block2[i]));

decryptedText += decryptedMessage;

}

//Расшифровка сообщения

Console.WriteLine("Запись расшифровоного сообщения");

//Запись расшифровоного сообщения

using (FileStream DecryptionFile = new FileStream("OUT.txt", FileMode.Create))

using (StreamWriter decryptedWriter = new StreamWriter(DecryptionFile))

{

decryptedWriter.Write(decryptedText);

}

//Запись расшифровоного сообщения

Console.ReadKey();

}

}

}

# Примеры работы программы

В первом примере входные файл IN.txt будет иметь такое содержание (рисунок 5.1).

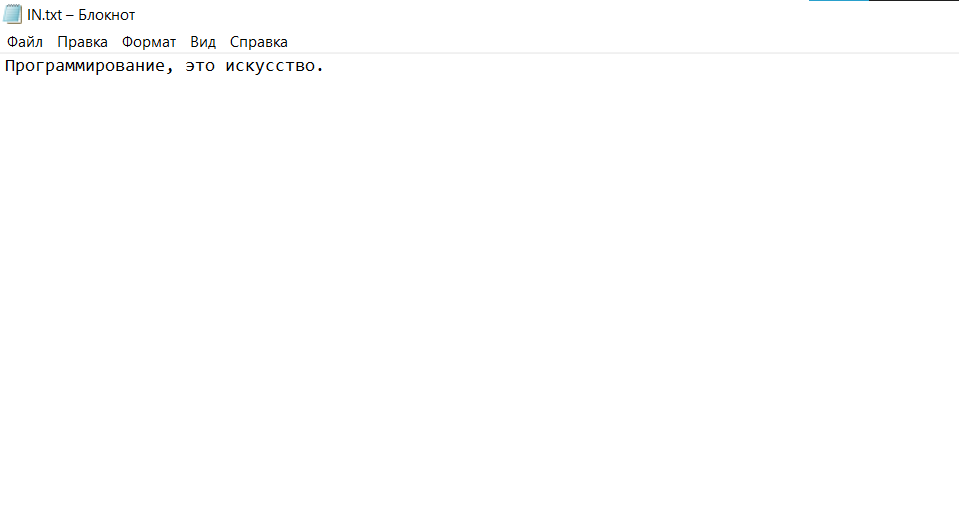


Рисунок 5.1 – Входной файл IN.txt

При запуске программы на экране будет выведена информация об этапах выполнения шифрования и расшифровывания входного файла (рисунок 5.2).

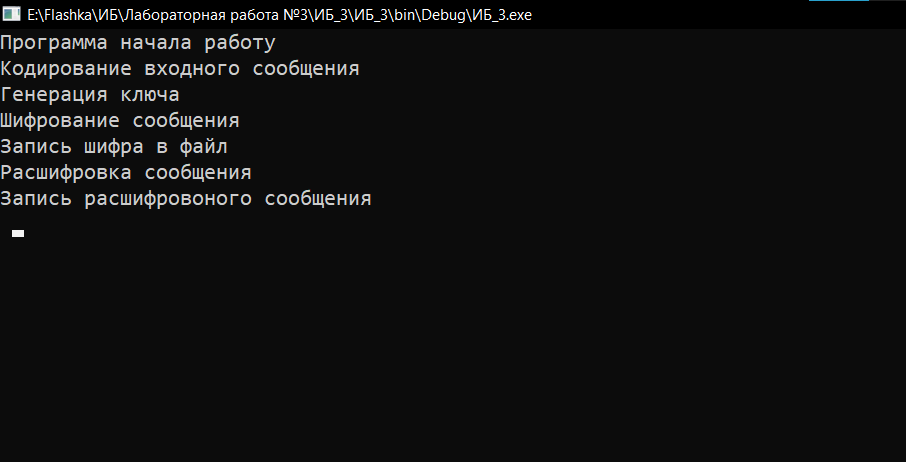


Рисунок 5.2 – Экран программы с информацией об этапах работы

По завершению работы программа запишет зашифрованный и расшифрованный текст в файлы (рисунок 5.3-5.4).

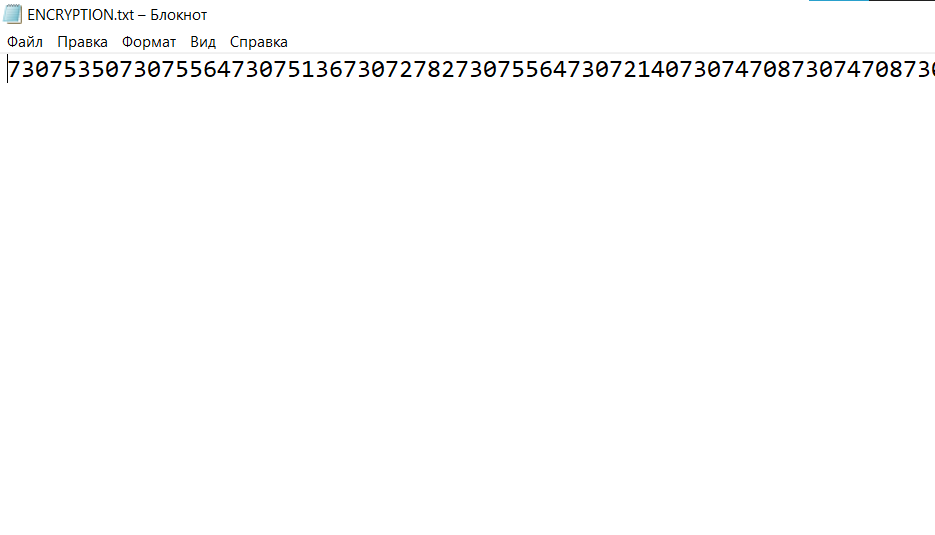


Рисунок 5.3 – Файл ENCRYPTION.txt с зашифрованным текстом

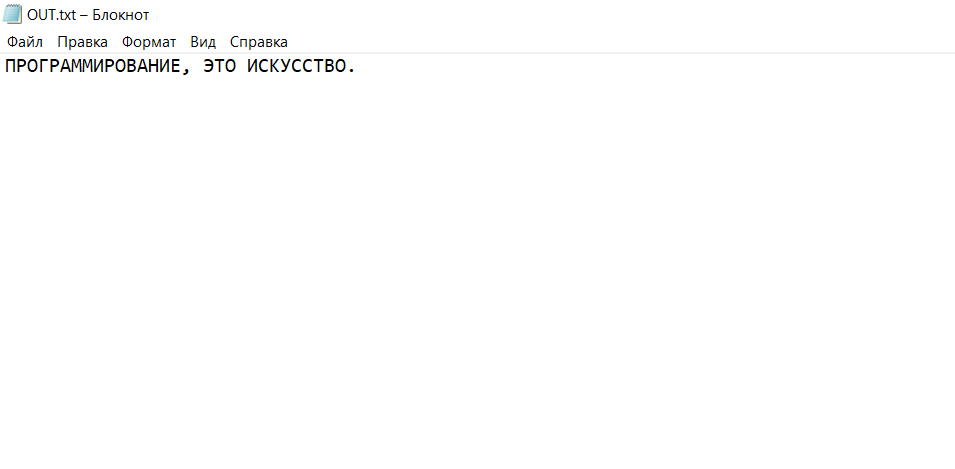


Рисунок 5.4 – Файл OUT.txt с расшифрованным текстом

Во втором пример будем использовать такой входной файл (рисунок 5.5).

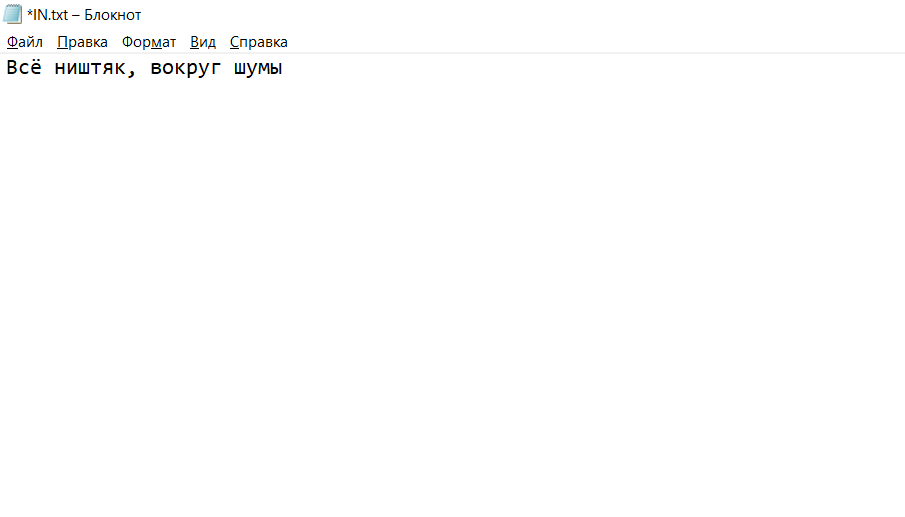


Рисунок 5.5 – Входной файл IN.txt

На выходе мы получим файлы с таким содержанием (рисунок 5.6-6.7).

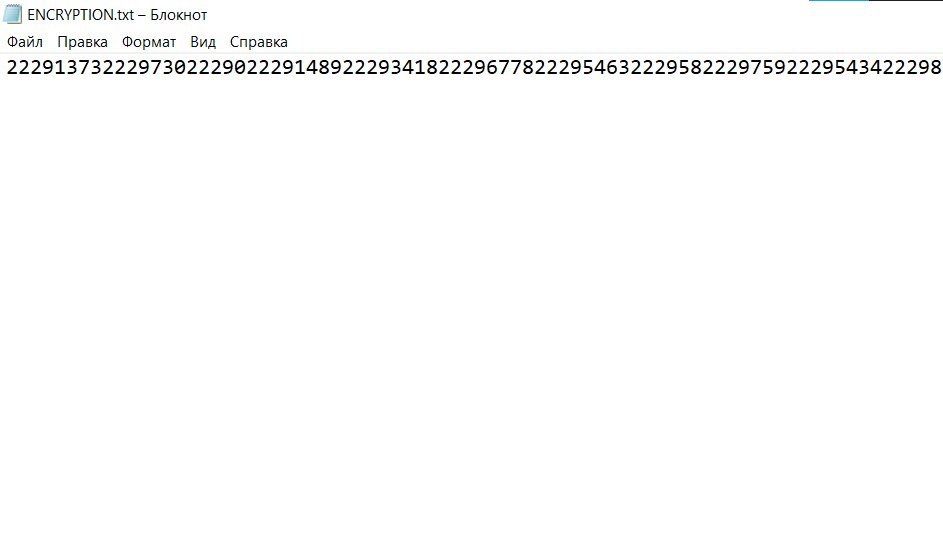


Рисунок 5.6 – Файл ENCRYPTION.txt с зашифрованным текстом

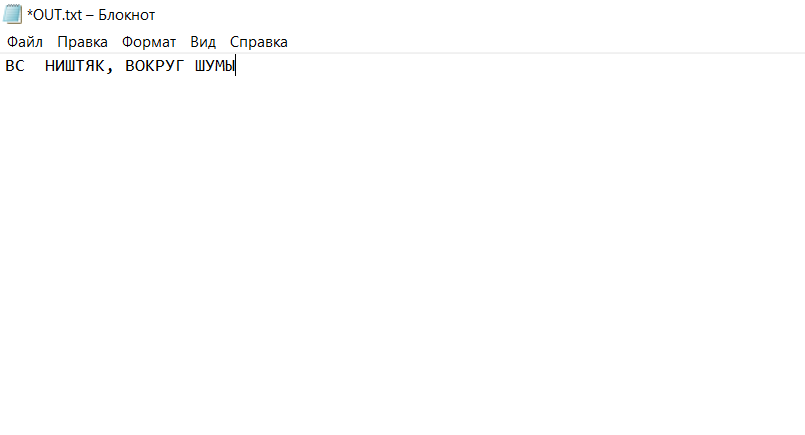


Рисунок 5.7 – Файл OUT.txt с расшифрованным текстом

В данном примере мы видим, что текст был расшифрован не полностью, а именно не была расшифрована буква ё. Это произошло из-за того, что буквы ё нет в таблице кодировки сообщения.

# Вывод

В ходе выполнения лабораторной работы я познакомился и научился работать с асимметричным алгоритмом шифрования Эль-Гамаля.