Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

«Исследование потоковых шифров»

Студент: Белицкий В.Д.

ФИТ 3 курс 5 группа

Преподаватель:

Савельева Маргарита Геннадьевна

Минск 2023

1. **Описание приложения**

Приложение написано на языке программирования *C#* и позволяет выполнить 2 задачи:

* генерация псевдослучайной последовательности на основе алгоритма RSA;
* зашифрование и расшифрование строки с помощью алгоритма RC4 с параметрами: *n* = 8; ключ = {121, 14, 89, 15}.

Также разработанное приложение позволяет оценить скорость выполнения операций генерации ПСП.

**2.1. Генерация ПСП на основе RSA**

Для реализации генерации псевдослучайной последовательности (ПСП) на основе алгоритма *RSA* был разработан класс *RSARandomGenerator*, класс представлен на рисунке 2.1.

public class RSARandomGenerator

{

private BigInteger p;

private BigInteger q;

private BigInteger n;

private BigInteger e;

private BigInteger x;

public RSARandomGenerator(BigInteger p, BigInteger q, BigInteger e, BigInteger x)

{

this.p = p;

this.q = q;

this.n = p \* q;

this.e = e;

this.x = x;

}

public byte GenerateRandomBit()

{

x = BigInteger.ModPow(x, e, n);

byte randomBit = (byte)(x % 2);

return randomBit;

}

Рис. 2.1 – Реализация генерации числа ПСП

Класс ***RSARandomGenerator*** представляет генератор случайных битов с использованием алгоритма *RSA (Rivest-Shamir-Adleman).* Конструктор класса принимает значения простых чисел ***p*** и ***q***, публичного ключа ***e*** и начального значения***x***.

Основной метод ***GenerateRandomBit*()** генерирует случайный бит, используя операцию возведения в степень по модулю. Сначала применяется операция ***BigInteger.ModPow(x, e, n)****,* которая возводит значение ***x*** в степень ***e*** по модулю***n***. Затем полученный результат ***x*** сокращается по модулю 2, приводя к значению 0 или 1, и возвращается в виде ***byte*** - случайного бита.

Таким образом*,* ***RSARandomGenerator*** позволяет генерировать последовательность случайных битов на основе заданных параметров RSA.



Рис. 2.1 – Результат работы первого задания (RSA)

**2.2. Реализация алгоритма RC4**

Для решения поставленной задачи разработаем C#-класс RC4.

class RC4

{

private byte[] S;

private int i;

private int j;

public RC4(int[] key)

{

byte[] keyBytes = new byte[key.Length];

for (int k = 0; k < key.Length; k++)

{

keyBytes[k] = (byte)key[k];

}

Initialize(keyBytes);

}

private void Initialize(byte[] key)

{

S = new byte[256];

for (int i = 0; i < 256; i++)

{

S[i] = (byte)i;

}

int j = 0;

for (int i = 0; i < 256; i++)

{

j = (j + S[i] + key[i % key.Length]) % 256;

Swap(S, i, j);

}

i = 0;

j = 0;

}

private void Swap(byte[] array, int i, int j)

{

byte temp = array[i];

array[i] = array[j];

array[j] = temp;

}

public byte[] Encrypt(byte[] plaintext)

{

byte[] ciphertext = new byte[plaintext.Length];

for (int k = 0; k < plaintext.Length; k++)

{

i = (i + 1) % 256;

j = (j + S[i]) % 256;

Swap(S, i, j);

byte key = S[(S[i] + S[j]) % 256];

ciphertext[k] = (byte)(plaintext[k] ^ key);

}

return ciphertext;

}

public byte[] Decrypt(byte[] ciphertext)

{

return Encrypt(ciphertext); // RC4 encryption and decryption are the same operation

}

}

Рис. 2.3 – Реализация класса RS4

Сначала *S*-блок пополняется линейно: 0,1…255. Затем заполняется   
Класс **RC4** реализует алгоритм *RC4 (Rivest Cipher 4),* который является потоковым шифром.

В конструкторе класса **RC4** принимается ключ в виде массива целых чисел ***key***, который преобразуется в массив байтов **keyBytes**. Затем вызывается приватный метод ***Initialize***, который инициализирует внутреннее состояние шифра на основе ключа.

Метод ***Initialize*** создает и инициализирует массив***S*** размером 256 байтов. Значения массива ***S*** устанавливаются от 0 до 255. Затем выполняется перестановка элементов массива **S** на основе ключа и индексов ***i*** и ***j****.* Каждая итерация цикла производит обмен значениями элементов ***S[i]*** и ***S[j]****,* где***i*** и ***j***обновляются в зависимости от значений текущего элемента ***S[i]****,* индекса ***i***, текущего элемента ***S[j]*** и индекса ***j***. Это выполняется для достижения инициализации начального состояния шифра.

Метод ***Encrypt*** выполняет шифрование переданного массива байтов ***plaintext*** с использованием алгоритма *RC4*. Он итерирует по каждому байту входного массива и выполняет операции на основе текущего состояния шифра. Обновляются значения индексов***i*** и ***j***, производится обмен значениями элементов массива **S**, а затем выполняется операция *XOR* между текущим байтом ***plaintext*[k]** и ключевым байтом ***key***, полученным из массива ***S***. Результат записывается в массив ***ciphertext***, который возвращается в конце метода.

Метод ***Decrypt*** просто вызывает метод ***Encrypt***, так как шифрование и дешифрование *RC4* являются одной и той же операцией. Таким образом, переданный массив ***ciphertext*** будет преобразован обратно в исходный текст.

Результат выполнения данного консольного приложения представлен на рисунке 2.4.

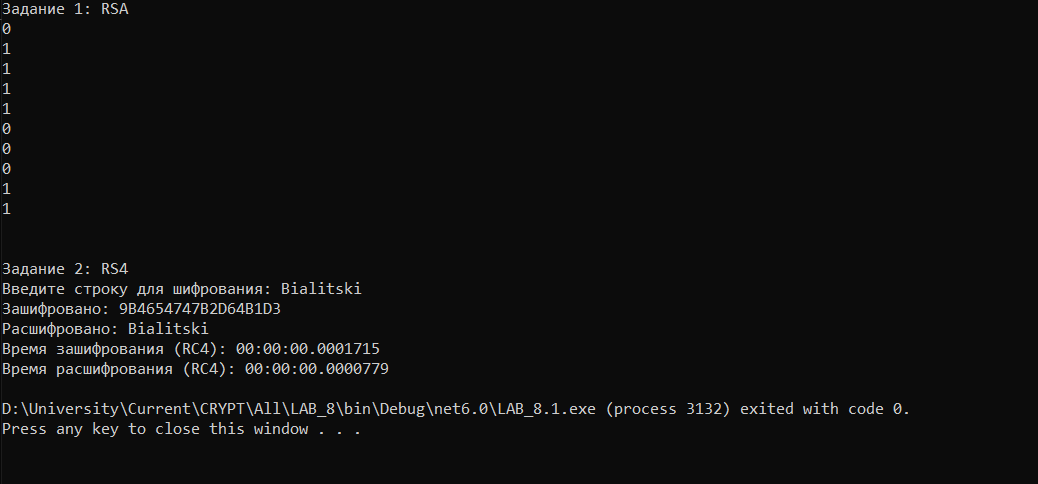


Рис. 2.4 – Результат работы приложения

**Вывод**

В ходе лабораторной работы были приобретены практические навыки разработки и использования приложений для реализации потоковых шифров. Было разработано приложение для реализации RSA генерации псевдослучайной последовательности. Также, был реализован алгоритм RC-4.