Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 8**

«Исследование потоковых шифров»

Выполнил:

Студент: Герман А.Е.

ФИТ 3 курса 5 группы

Преподаватель: Савельева М.Г

Минск 2024

# **1 Описание приложения**

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

* генерация псевдослучайной последовательности на основе алгоритма LGC;

зашифрование и расшифрование строки с помощью алгоритма RC4 с параметрами: *n*=8; ключ={123, 125, 41, 84, 203}.

Также разработанное приложение позволяет оценить скорость выполнения операций генерации ПСП. В качестве шифруемого сообщения выбран произвольный текст «Herman Alexander».

# **2 Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, реализующее потоковое шифрование RC4 и LGC. На листингах 2.1-2.2 представлены методы, реализующий данную функциональность.

Для решения поставленной задачи разработаем C#-класс RC4.

|  |
| --- |
| public RC4(byte[] key)  {  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;  S.Swap(i, j); // Поменять местами  }  }  // Для каждого байта массива исходных данных запрашиваем байт ключа и объединяем их при помощи XOR (^)  public byte[] Encode(byte[] dataB, int size)  {  byte[] data = dataB.Take(size).ToArray();  byte[] cipher = new byte[data.Length];  for (int m = 0; m < data.Length; m++)  {  cipher[m] = (byte)(data[m] ^ keyItem());  }  return cipher;  }  private byte keyItem()  {  x = (x + 1) % 256;  y = (y + S[x]) % 256;  S.Swap(x, y);  return S[(S[x] + S[y]) % 256];  } |

Листинг 2.1 – Методы класса **RC4**, реализующий шифр RC4

Для зашифрования и расшифрования исходного сообщения используется функция **Encode**, представленная на рисунке 2.1. В теле выше изложенной функции также вызывается функция **keyItem**, представленная на рисунке 2.1.

|  |
| --- |
| public static int LCGnext(int prev, int index)  {  int res = (a \* prev + c) % n;  Console.WriteLine($"x{index} = ({a}\*{prev} + {c}) mod {n} = {res}");  return res;  } |

Листинг 2.1 – Реализация алгоритма LGC

Реализация данного алгоритма представлена на листинге 2.1.

# **3 Результаты работы приложения**

Для выполнения расчетов достаточно необходимо запустить приложение. Рисунки 3.1 – 3.4 показывают необходимые расчеты и вызовы методов, требуемые в данной лабораторной работе.

## **3.1 Линейный конгруэнтный генератор**

|  |
| --- |
| **Вариант 5: Линейный конгруэнтный генератор; а = 421, с = 1663, n = 7875;** |

Входные данные:

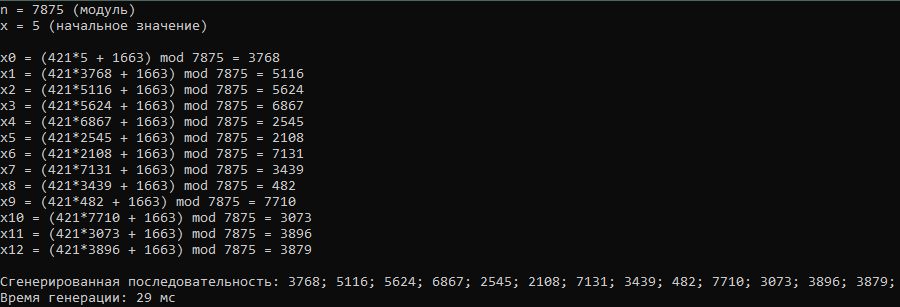


Рисунок 3.1 – Результат работы метода **LGCnext** с входными данными из варианта

Функция LCGnext является основным строительным блоком для генерации последовательности псевдослучайных чисел с использованием линейного конгруэнтного метода. Она принимает предыдущее значение и вычисляет следующее значение, используя фиксированные параметры множителя, прибавки и модуля, обеспечивая последовательность чисел в определённых пределах.

## **3.2 Шифр RC4**

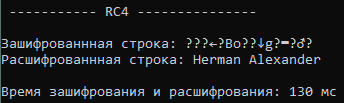


Рисунок 3.2 – Результат работы метода **RC4**

Сначала *S*-блок пополняется линейно: 0,1…255. Затем заполняется секретным ключом другой массив [256]. Если необходимо, ключ повторяется многократно чтобы заполнить весь массив *K*0…*K*255. Далее массив *S* перемешивается путем перестановок, определяемых ключом.

В параметры данной функции передается массив исходных байтов и их размер. Для каждого исходного байта запрашивается текущий байт ключа, после чего они объединяются при помощи XOR для получения 8-битного шифротекста.

## **3.3 График времени для LGC**

Рисунок 3.3 – Время генерации LGC

## **3.4 График времени для RC4**

Рисунок 3.4 – Время шифрования RC4

# **4 Вывод**

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