Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Исследование потоковых шифров**

Студент: Круглик А.В.

ФИТ 3 курс 5 группа

Преподаватель:Савельева М.Г.

Минск 2023

1. **Описание приложения**

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

* генерация псевдослучайной последовательности на основе алгоритма BBS;
* зашифрование и расшифрование строки с помощью алгоритма RC4 с параметрами: *n*=6; ключ={61, 60, 23, 22, 21, 20}.

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

1. **Методика выполнения поставленных задач**

**2.1. Генерация ПСП на основе алгоритма BBS**

Для реализации генерации псевдослучайной последовательности (ПСП) на основе алгоритма BBS была разработана функция, представленная на рисунке 2.1.

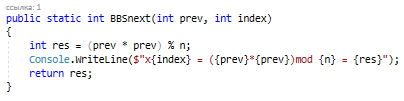


Рис. 2.1 – Реализация генерации числа ПСП

Данная функция вычисляет каждое число последовательности отдельно по формуле квадратичных вычетов: , где *xt* – вычисляемое значение генератора, *xt*-1 – предыдущий элемент ПСП, *n* = *pq* (число Блюма), причем простые числа *p* и *q* должны быть сравнимы с числом 3 по модулю 4 (в нашем случае 11 и 23).

Для вычисления начального значения генератора *x*0 используется то же соотношение, но вместо предыдущего элемента ПСП *xt-*1в функцию подается *х*=5 – число, взаимно простое с числом Блюма n (выбрано нами самостоятельно).

В функции main() мы циклично вызываем выше изложенную функцию, последовательно заполняя массив seq, который хранит числа генерируемой ПСП.

Также, нами была оценена скорость выполнения генерации ПСП с помощью встроенной возможности C# - DateTime.Now.Ticks. Вычисленное время составило 40 мс (рисунок 2.3). Следовательно, данный алгоритм является сравнительно медленным.

**2.2. Реализация алгоритма RC4**

Для решения поставленной задачи разработаем C#-класс RC4. При создании экземпляра данного класса вызывается конструктор, код которого приведен на рисунке 2.2.

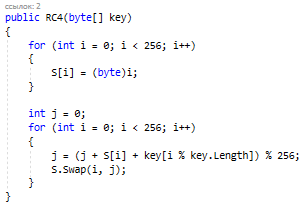


Рис. 2.2 – Конструктор RC4

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.

Для зашифрования и расшифрования исходного сообщения используется функция Encode, представленная на рисунке 2.3.

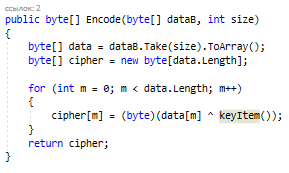


Рис. 2.3 – Реализация функции Encode

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.

В теле выше изложенной функции также вызывается функция keyItem, представленная на рисунке 2.4.

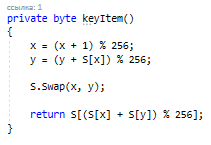


Рис. 2.4 – Реализация функции keyItem

Эта часть алгоритма называется генератором ПСП при *n*=6. При каждом вызове функция отдает следующий байт ключевого потока.

Результат выполнения данного консольного приложения представлен на рисунке 2.5.

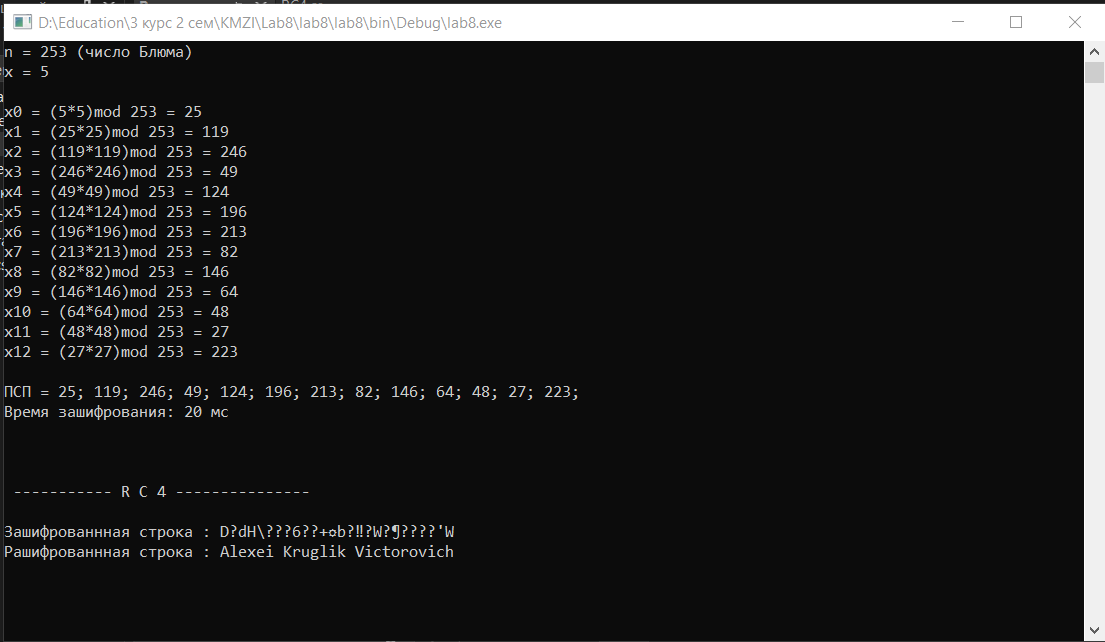


Рис. 2.5 – Результат работы приложения

**Вывод**

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