Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Исследование потоковых шифров**

Студент: Шулаков А.А.

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*=8; ключ={20, 21, 22, 23, 60, 61}.

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

1. **Методика выполнения поставленных задач**

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

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

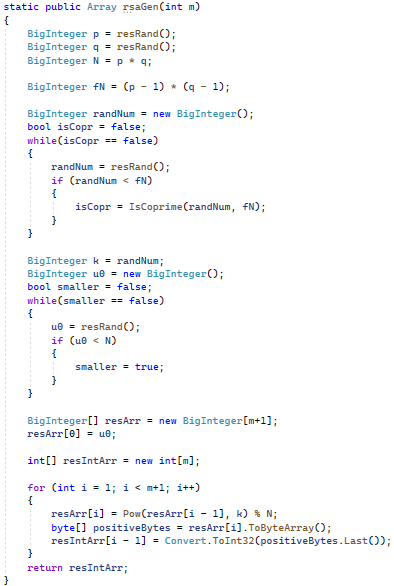


Рис. 2.1 – Реализация генерации числа ПСП

Генерируются два случайных простых числа p и q.Вычисляется произведение: . Вычисляется функция Эйлера *f(N)*, которая определяет количество взаимно простых чисел с *N*. Генерируется случайное число *e*, которое является открытым ключом. Вычисляется закрытый ключ *d*, удовлетворяющий условию , используя расширенный алгоритм Евклида.

В функции main() мы циклично вызываем выше изложенную функцию, последовательно заполняя массив resIntArr, 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. 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.

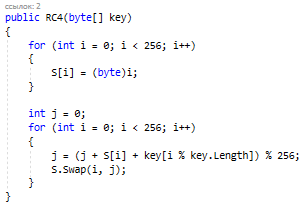


Рис. 2.2 – Конструктор RC4

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.

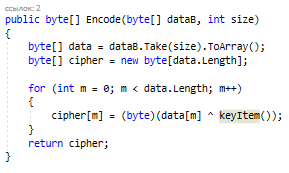


Рис. 2.3 – Реализация функции Encode

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.

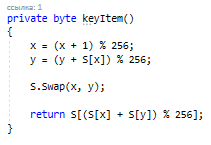


Рис. 2.4 – Реализация функции keyItem

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.

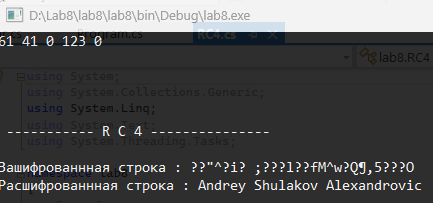


Рис. 2.5 – Результат работы приложения

**Вывод**

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