МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

**«Исследование потоковых шифров»**

Студент:

Храмых Владислав Олегович

Вариант 2

Преподаватель:

Блинова Евгения Александровна

Минск 2020

**ЗАДАНИЕ 1.** Генерация ПСП используя алгоритм RSA. (Используемый язык - C#)

Собственно алгоритм RSA разработан для систем асимметричного зашифрования/расшифрования и будет более детально рассмотрен с практической точки зрения ниже.

Для работы с этим алгоритмом, потребуются некоторые входные параметры:

**p, q** – простые числа.

**n** – которое будет равняться произведению: **n = p\*q**.

**e** – целое число, которое является **взаимно простым с (p-1) \* (q-1).**

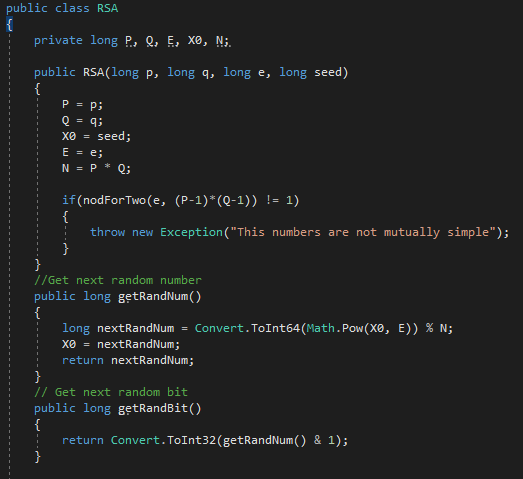
**X0** – некоторое случайное число (начало последовательности)

Генератор ПСП на основе RSA построен следующим образом:

**Xt = (xt-1)^e mod n**

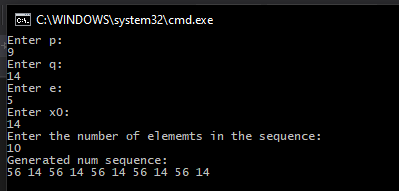
Выходом генератора является на t-м шаге является младший бит числа xt. Безопасность генератора опирается на сложности взлома алгоритма RSA, т. е. на разложении числа n на простые сомножители.

Реализация класса RSA:



Метод getRandNum() возвращает каждое следующее значение генератора в числовом виде, а т.к выходом генератора на t- м шаге является младший бит этого значения, то для его получения используется метод getRandBit().

Результат работы программы представлен ниже.



**ЗАДАНИЕ 2**. Реализовывать алгоритм RC4 в соответствии с вариантом (вар-т 2)

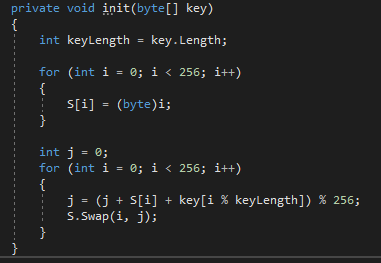




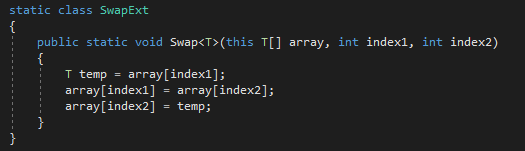
Входными данными у нас будет выступать массив байт. Ключ тоже выступает в качестве входных данных. Для алгоритма RC4 он может быть от 8 до 2048 бит, но обычно используется диапазон 40 — 256 бит.

Но данные для шифрования у нас — массив байт, а ключ почему-то в битах. Дело в том, что существует такое понятие как размер блока n. Тут используется n = 8, т. е за один шаг шифруется 1 байт. При n = 8 элементы блока представляют собой перестановку чисел от 0 до 255, а сама перестановка зависит от ключа переменной длины.

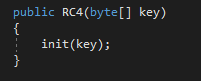
Для начальной инициализация вектора-перестановки ключом, используется алгоритм ключевого расписания (Key-Scheduling Algorithm):



Сначала заполняем 256-байтный массив ключом. Если необходимо, ключ повторяется многократно, чтобы заполнить весь массив. Для перестановки используется метод Swap, который меняет два элемента местами и расширяет стандартный список методов класса Array. Реализация метода представлена ниже.

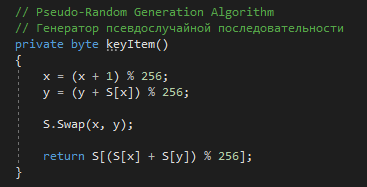


Метод init нужно вызвать перед шифровкой/расшифровкой, когда известен ключ. Можно сделать это в конструкторе:

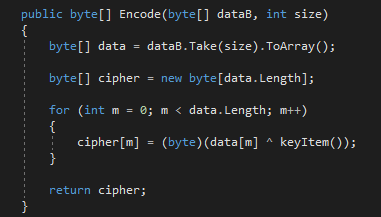


Дальше нужно реализовать генератор псевдослучайной последовательности. При каждом вызове метод будет выдавать последующий байт ключевого потока, который мы и будем объединять XOR'ом c байтом исходных данных. Генератор ключевого потока RC4 переставляет значения, хранящиеся в S,

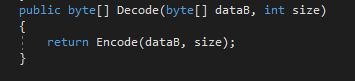
и каждый раз выбирает различное значение из S в качестве результата.



Для каждого байта массива/потока входных незашифрованных данных запрашиваем байт ключа и объединяем их при помощи xor (^):



Для расшифровки можно использовать этот же метод. Завернем его в отдельный метод для наглядности:



Результат работы представлен ниже.

