Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование потоковых шифров

Студент: Авсюкевич П.В.

ФИТ 3 курс 5 группа

Преподаватель: Савельева М. Г.

Минск 2024

Задание 1: разработка авторского приложения для генерации ПСП по алгоритму RSA. *P, q, e* выбраны 512 разрядные числа

Для реализации алгоритма генерации псевдослучайных последовательностей с помощью алгоритма RSA был создан класс, пример представлен на листинге 1.1.

|  |
| --- |
| public class RSARandomGener  {  private BigInteger p;  private BigInteger q;  private BigInteger n;  private BigInteger e;  private BigInteger x;  public RSARandomGener(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;  }  } |

Листинг 1.1 – Код генерации ПСП с помощью RSA

В соответствии с вариантом, на вход алгоритма подаются числа. Пример которых представлен на листинге 1.2.

|  |
| --- |
| BigInteger p = BigInteger.Parse("57896044618658097711785492504343953926634992332820282019728792003956564819967");  BigInteger q = BigInteger.Parse("76743985164523602061355291354880921859389331661131769782895224264006620558367");  BigInteger e = BigInteger.Parse("65537");  BigInteger x0 = BigInteger.Parse("12345"); |

Листинг 1.2 – Пример значений чисел

Результат работы алгоритма, представлен на рисунке 1.1



Рисунок 1.1 – Результат работы

Задание 2: реализация алгоритма RC4, выполнение оценки скорости выполнения операций генерации ПСП

Первый пункт для начала работы является инициализации *S*-блока начальных замен, пример реализации представлен на рисунке 1.2.

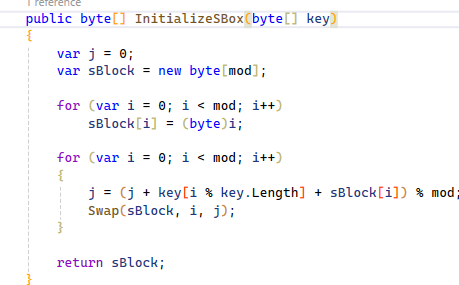


Рисунок 1.2 – Инициализация S-блока

Вторым шагом является создание псевдослучайной последовательности ключей. Алгоритм представлен на рисунке 1.3.

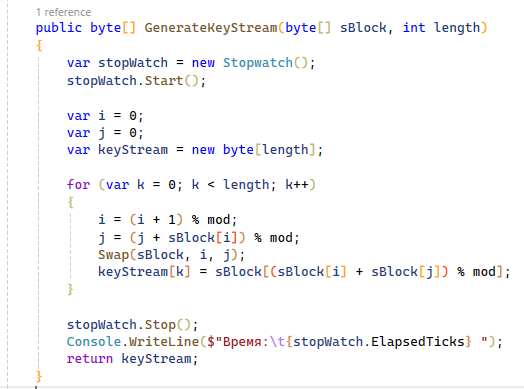


Рисунок 1.3 – Алгоритм создания псевдослучайной последовательности

Алгоритм шифрования и дешифрования представлен на рисунке 1.4

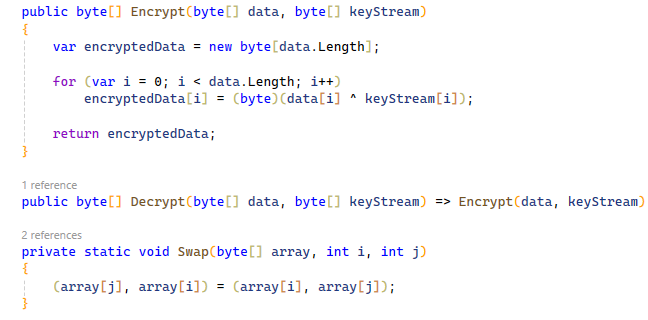


Рисунок 1.4 – Алгоритм зашифрования и расшифрования

Результат работы программы представлен на рисунке 1.5.



Рисунок 1.5 – Результат работы программы

Для оценки времени работы алгоритма были созданы псевдослучайные последовательности разной длины. График представлен на рисунке 1.6.

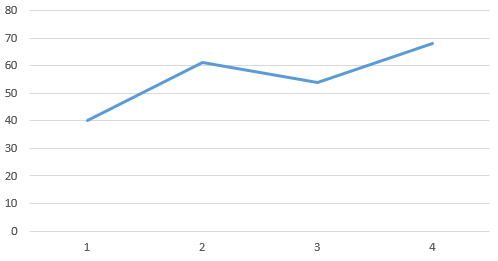


Рисунок 1.6 – Время работы алгоритма

Эта лабораторная работа позволила приобрести практические навыки в разработке и использовании потоковых шифров. Потоковые шифры работают, шифруя отдельные биты текста по мере их передачи, что особенно полезно в сетевых приложениях или системах реального времени из-за низкой задержки. Однако они чувствительны к синхронизации и могут быть уязвимы при повторном использовании ключей. Важно выбирать параметры, такие как ключи и начальные векторы, с осторожностью. В целом, потоковые шифры представляют собой мощный инструмент, но их безопасное применение требует понимания и осторожности.