МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8 НА ТЕМУ:**

**ИССЛЕДОВАНИЕ ПОТОКОВЫХ ШИФРОВ**

                                                                 Выполнил студент 3 курса 5 группы

Дмитрук Илья Игоревич

Минск 2024

# Задание 1.

Для генерации последовательности случайных чисел (ПСП) использовался алгоритм линейный конгруэнтный генератор. Согласно ему, каждое новое число образуется из предыдущего по формуле:

*xt*+1 ≡ (*axt* + *c*) mod *n*

где *xt* и *xt* + 1 – соответственно *t*-й (предыдущий) и (*t* + 1)-й (текущий, вычисляемый) члены числовой последовательности; *а*, *с* и *n* – константы. Период такого генератора (период ПСП) не превышает *n*. Данный алгоритм реализован в классе LinearCongruentialGenerator. Код данного класса представлен в листинге 1.1.

class LinearCongruentialGenerator {

    constructor(seed, a, c, n) {

        this.a = a;

        this.c = c;

        this.n = n;

        this.state = seed;

    }

    next() {

        this.state = (this.a \* this.state + this.c) % this.n;

        return this.state;

    }

}

Листинг 1.1 – Класс, реализующий алгоритм линейного конгруэнтного генератора

Установив параметры со значениями *a*=421, *c*=1663, *n*=7875, и seed=1234, мы получили результат, представленный на рисунке 1.1.



Рисунок 1.1 – Числа, сгенерированные линейным конгруэнтным генератором

# Задание 2.

Алгоритм шифрования RC4 является потоковым шифром. Он используется для шифрования данных путем создания псевдослучайного потока байтов, который затем комбинируется с данными с помощью операции XOR. Алгоритм состоит из двух основных этапов: инициализации и генерации псевдослучайного потока. Инициализация включает в себя подготовку состояния (*S*-блока) и ключевого расписания. После инициализации алгоритм переходит к генерации псевдослучайного потока байтов, который используется для шифрования и расшифрования. Шифрование и расшифрование реализовано в функции rc4EncryptDecrypt и вспомогательных для неё функциях, код которых представлен в листинге 2.1.

function rc4KeyScheduling(key) {

    let S = [];

    for (let i = 0; i < 256; i++) {

        S[i] = i;

    } let j = 0;

    for (let i = 0; i < 256; i++) {

        j = (j + S[i] + key[i % key.length]) % 256;

        [S[i], S[j]] = [S[j], S[i]];

    }

    return S;

}

function rc4PseudoRandomGeneration(S, messageLength) {

    let i = 0, j = 0; let keyStream = [];

    for (let k = 0; k < messageLength; k++) {

        i = (i + 1) % 256;

        j = (j + S[i]) % 256;

        [S[i], S[j]] = [S[j], S[i]]; keyStream.push(S[(S[i] + S[j]) % 256]);  }

    return keyStream;

}

function rc4EncryptDecrypt(message, key) {

    const keyScheduling = rc4KeyScheduling(key);

    const keyStream = rc4PseudoRandomGeneration(keyScheduling, message.length);

    let result = Buffer.alloc(message.length);

    for (let i = 0; i < message.length; i++) [i] = message[i] ^ keyStream[i];return result;

}

Листинг 2.1 – Реализация алгоритма шифрования и расшифрования RC4

Результат шифрования алгоритмом RS4 представлен на рисунке 2.1, и время генерации ПСП на рисунке 2.2.

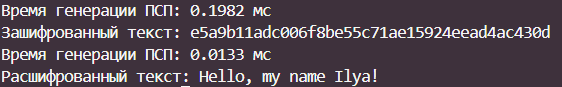


Рисунок 2.1 – Результат шифрования алгоритмом RS4



Рисунок 2.2 – Время выполнения генерации ПСП в алгоритме RC4

# Вывод

В ходе лабораторной работы были приобретены навыки в создании генераторов ПСП и шифрования алгоритмом RC4. Алгоритм линейного конгруэнтного генератора ПСП и алгоритм шифрования RC4 были реализован программными средствами.