Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра информационных](https://www.belstu.by/fakultety/fit/vm) систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №8**

по дисциплине Информационная безопасность

Тема: Исследование потоковых шифров

        Исполнитель:

Студент 3 курса группы 4

Станчик Максим Андреевич

Руководитель:

Ассистент Савельева М. Г.

Минск, 2025

**Лабораторная работа №8**

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации потоковых шифров.

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости потоковых шифров.

2. Разработать приложение для реализации указанных преподавателем методов генерации ключевой информации и ее использования для потокового зашифрования/расшифрования.

3. Выполнить анализ криптостойкости потоковых шифров.

4. Оценить скорость зашифрования/расшифрования реализованных шифров.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента

# Теоретические сведения

Потоковый шифр (иногда говорят «поточный») – симметричный шифр, преобразующий каждый символ *mi* открытого текста в символ шифрованного *ci*, зависящий от ключа и расположения символа в тексте.

Все потоковые шифры делятся на 2 класса: синхронные и асинхронные (или самосинхронизирующиеся).

Основной задачей потоковых шифров является выработка некоторой последовательности (гаммы) для зашифрования, т. е. выходная гамма является ключевым потоком (ключом) для сообщения.

Синхронные потоковые шифры (СПШ) характеризуются тем, что поток ключей генерируется независимо от открытого текста и шифртекста. Главное свойство СПШ – нераспространение ошибок. Ошибки отсутствуют, пока работают синхронно шифровальное и дешифровальное устройства отправителя и получателя информации.

Синхронные потоковые шифры уязвимы к атакам на основе изменения отдельных битов шифртекста.

В самосинхронизирующихся потоковых шифрах символы ключевой гаммы зависят от исходного секретного ключа шифра и от конечного числа последних знаков зашифрованного текста. Основная идея заключается в том, что внутреннее состояние генератора потока ключей является функцией фиксированного числа предыдущих битов шифртекста. Поэтому генератор потока ключей на приемной стороне, приняв фиксированное число битов, автоматически синхронизируется с генератором гаммы.

Ключевые последовательности (случайные последовательности (СП), либо псевдослучайные последовательности (ПСП)) вырабатываются специальными блоками систем потокового шифрования – генераторами.

Часто используемый алгоритм генерирования (программно или аппаратно) ПСП реализуется на основе так называемого линейного конгруэнтного генератора, описываемого следующим рекуррентным соотношением:

*xt*+1 ≡ (*a*×*xx* + *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*.

Алгоритм RC4 представляет собой потоковый шифр с переменным размером ключа. Здесь гамма не зависит от открытого текста. Алгоритм RC4, как и любой потоковый шифр, строится на основе генератора псевдослучайных битов (генератора ПСП). На вход генератора записывается ключ, а на выходе читаются псевдослучайные биты. Длина ключа может составлять от 40 до 2048 битов. Ядро алгоритма состоит из функции генерации ключевого потока. Другая часть алгоритма – функция инициализации, которая использует ключ переменной длины *Ki* для создания начального состояния генератора ключевого потока. В основе алгоритма – размер блока или слова, определяемый параметром *n*. Обычно *n* = 8, но можно использовать и другие значения. Внутренне состояние шифра определяется массивом слов (*S*-блоком) размером 2n. При *n* = 8 элементы блока представляют собой перестановку чисел от 0 до 255, а сама перестановка зависит от ключа переменной длины. Другими элементами внутреннего состояния являются 2 счетчика с нулевыми начальными значениями. В основе вычислений лежит операция по mod2n.

# Ход работы

В начале, нужно было разработать приложение, реализующее генерацию ПСП на основе линейного конгруэнтного генератора с параметрами *а* = 421, *с* = 1663, *n* = 7875.

Метод lcg принимает на вход начальное число последовательности, длину генерируемой последовательности и параметры линейного конгруэнтного генератора. Код метода представлен в листинге 2.1.

public List<Integer> lcg(int seed, int sequenceLength, int a, int c, int n) {

List<Integer> sequence = new ArrayList<>();

for (int i = 0; i < sequenceLength; i++) {

sequence.add(seed);

seed = (a \* seed + c) % n;

}

return sequence;

}

Листинг 2.1 – Код метода шифрования

Результат работы приложения с начальным числом 124 приведен на рисунке 2.1.

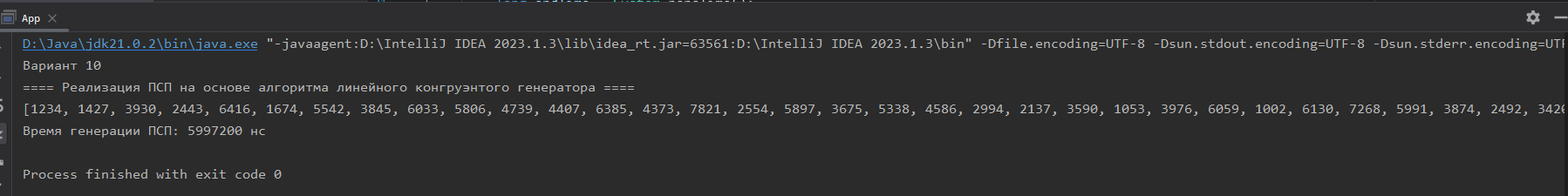


Рисунок 2.1 – Результат генерации последовательности длиной 20 чисел

Далее, нужно было разработать приложение, реализующее алгоритм RC4.

Метод RC4encrypt принимает исходный текст и шифрует его. Код функции представлен на рисунке 2.3.

private String process(String data, int key[]) {

Map<Integer, Integer> box = new HashMap<>();

for (int i = 0; i < 256; i++) {

box.put(i, i);

}

for (int i = 0; i < 256; i++) {

int j = (box.get(i) + key[i % key.length]) % 256;

int temp = box.get(i);

box.put(i, box.get(j));

box.put(j, temp);

}

StringBuilder result = new StringBuilder();

List<Character> out = new ArrayList<>();

int x = 0;

for (char c : data.toCharArray()) {

x = (x + 1) % 256;

int y = box.get(x);

out.add((char) (c ^ box.get((box.get(x) + box.get(y)) % 256)));

}

for (char c : out) {

result.append(c);

}

return result.toString();

}

Листинг 2.2 – Код метода, реализующего RC4

Результат шифрования текста «Maxim Stanchik» представлен на рисунке 2.4.

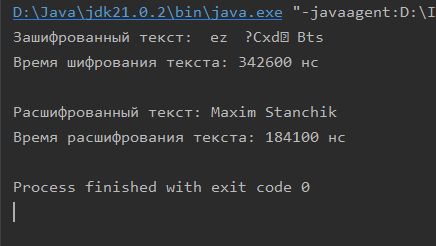


Рисунок 2.2 – Шифрование и дешифрование текста c помощью RC4

# Время выполнения генерации псевдослучайной последовательности

Для оценки времени выполнения операций генерации ПСП были созданы псевдослучайные последовательности разной длины. График показан на рисунке 3.1

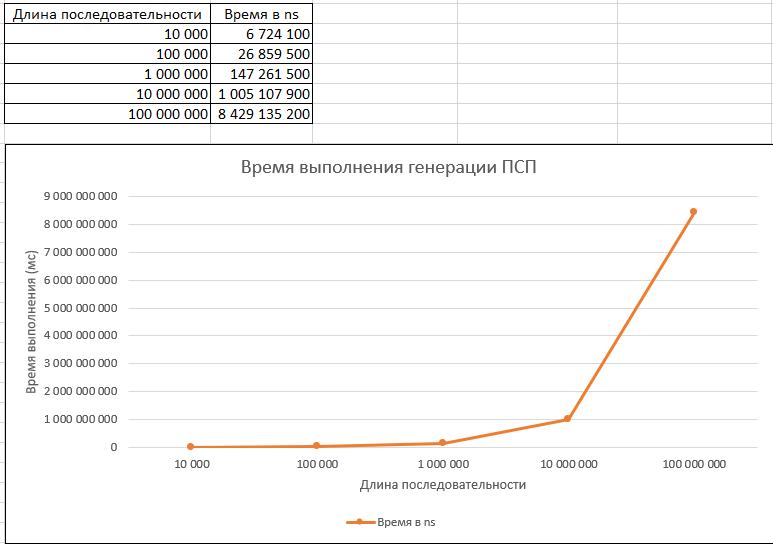


Рисунок 3.1 – График времени выполнения генерации ПСП

При анализе данного графика можно заметить, что генерация ПСП происходит гораздо быстрее, чем операции зашифрования и расшифрования в остальных алгоритмах, из чего можно сделать вывод, что алгоритм RC4 может быть довольно производительным для создания псевдослучайных последовательностей.

**Вывод:** в ходе выполнения лабораторной работы был изучен принцип реализации потоковых шифров. Также было разработано приложение для генерации ПСП на основе линейного конгруэнтного генератора, а также приложение, выполняющее шифрование и расшифрование с помощью шифра RC4. Была оценена скорость генерации ПСП шифра RC4.