Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Исследование потоковых шифров**

Студент: Рубашек А. А.

ФИТ 3 курс 5 группа

Преподаватель:

Савельева Маргарита Геннадьевна

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

**Задачи:**

1. 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

# Ход работы

**1.1** **Задание 1**

Необходимо разработать приложение, реализующее генерацию ПСП на основе линейного конгруэнтного генератора с параметрами *а* = 430, *с* = 2531, *n* = 11 979.

В качестве первого числа последовательности, возьмем любое число. Далее каждый следующий член считается. В приложении реализуем ввод количества генерируемых чисел.

Код функции представлен на рисунке 1.1

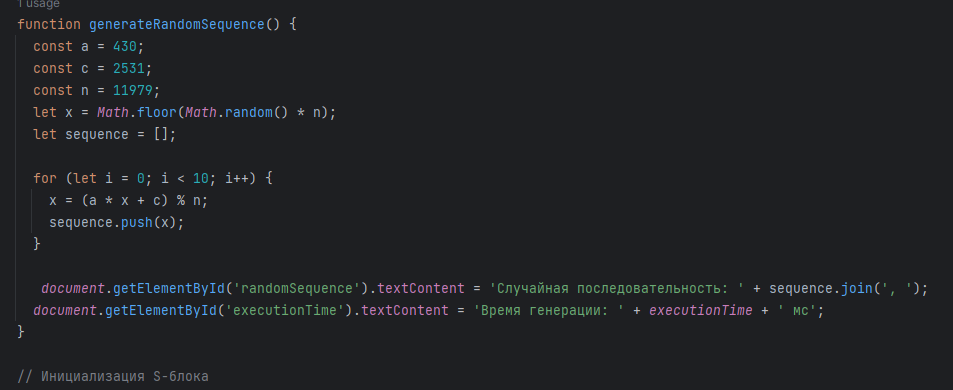


Рисунок 1.1 – Код функции генерации ПСП

Результат работы приложения приведен на рисунке 1.2 При вводе генерации 10 значений.

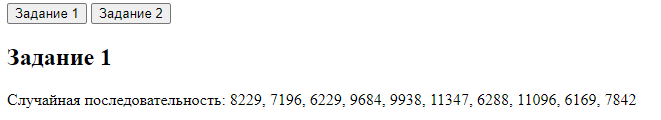


Рисунок 1.2 – Результат генерации ПСП на основе КЛГ

**1.1** **Задание 2**

Необходимо разработать приложение, реализующее алгоритм RC4.

Приложение позволяет вводить исходный текст, шифровать и дешифровать его.

Алгоритм шифрования RC4 начинается с инициализации *s*–блока. Сначала *s*–блок заполняется значениями от 1 до , где *n* – параметр, который указывает на количество бит для каждого элемента ключа. Далее в *s*–блоке происходят перестановки (Рисунок 1.2). После этого, уже сформированный *s*–блок используется для генерации Гаммы (Рисунок 1.3). Гамма необходима для операции XOR с исходным текстом.

Функция encrypt зашифровывает исходный текст. Код функции представлен в Приложении 1.

В функции encrypt используются функции initializeSBlock и generateKeyStream необходимые для инициализации *s*–блока и генерации потока Гаммы соответственно. Функции представлены в Приложении 2.

Дешифрование производит функция decrypt, представленная в Приложении 3. Алгоритм дешифрования идентичен с алгоритмом шифрования ­– Также производится операция XOR над шифрованным текстом и гаммой

Результат работы приложения представлен на рисунке 1.3. Можно заметить, что размер исходного текста и размер шифрограммы совпадают.

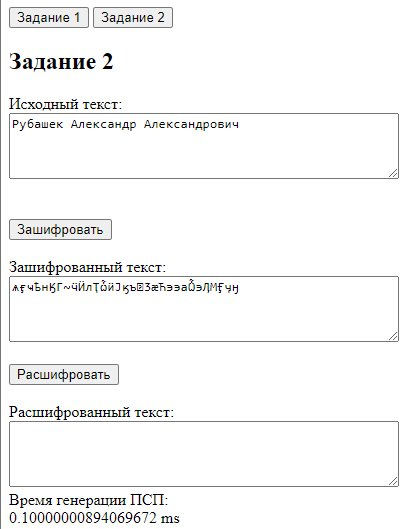


Рисунок 1.3 – Шифрование и дешифрование текста c помощью RC4

# Вывод

В ходе выполнения лабораторной работы был изучен принцип реализации потоковых шифров и генераторов ПСП. Также было разработано приложение для генерации ПСП на основе линейного конгруэнтного генератора и приложение, выполняющее шифрование и дешифрование с помощью шифра RC4 с оценкой скорости генерации ПСП.

**Приложение 1**

Код функции encrypt

 function encrypt() {

    const plaintext = document.getElementById('plaintext').value;

    textLength = plaintext.length;

    const sBlock = initializeSBlock(key);

    let startTime = performance.now();

    const keyStream = generateKeyStream(sBlock);

    let endTime = performance.now();

    let result = '';

    for (let k = 0; k < plaintext.length; k++) {

      let cipherByte = plaintext.charCodeAt(k) ^ keyStream[k % m];

      result += String.fromCharCode(cipherByte);

    }

    let executionTime = endTime – startTime;

    document.getElementById('ciphertext').value = result;

    document.getElementById('time').innerText = executionTime + ' ms';

  }

**Приложение 2**

Код функции initializeSBlock и generateKeyStream

   // Инициализация S–блока

  function initializeSBlock(key) {

    let s = [];

    for (let i = 0; i < m; i++) {

      s[i] = i;

    }

    let j = 0;

    for (let i = 0; i < m; i++) {

      j = (j + s[i] + key[i % key.length]) % m;

      // Swap s[i] and s[j]

      let temp = s[i];

      s[i] = s[j];

      s[j] = temp;

    }

    return s;

  }

  // Генерация ключевого потока RC4

  function generateKeyStream(sBlock) {

    let i = 0;

    let j = 0;

    let keyStream = [];

    for (let k = 0; k < m; k++) {

      i = (i + 1) % m;

      j = (j + sBlock[i]) % m;

      // Swap s[i] and s[j]

      let temp = sBlock[i];

      sBlock[i] = sBlock[j];

      sBlock[j] = temp;

      let t = (sBlock[i] + sBlock[j]) % m;

      let keyStreamByte = sBlock[t];

      keyStream.push(keyStreamByte);

    }

    return keyStream;

  }

**Приложение 3**

Код функции decrypt

function decrypt() {

    const ciphertext = document.getElementById('ciphertext').value;

    const sBlock = initializeSBlock(key);

    const keyStream = generateKeyStream(sBlock);

    let result = '';

    for (let k = 0; k < ciphertext.length; k++) {

      let plainByte = ciphertext.charCodeAt(k) ^ keyStream[k % m];

      result += String.fromCharCode(plainByte);

    }

    document.getElementById('decryptedText').value = result;

  }