Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Информационная безопасность**

Студент: Козека Е. М.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О. А.

Минск 2025

**Лабораторная работа №8. Исследование потоковых шифров**

**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 и 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.

Термин «потоковый шифр» обычно используется в том случае, когда шифруемые символы открытого текста представляются одной буквой, битом или реже – байтом.

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 – нераспространение ошибок. Ошибки отсутствуют, пока работают синхронно шифровальное и дешифровальное устройства отправителя и получателя информации. Один из методов борьбы с рассинхронизацией – разбить отрытый текст на отрезки, начало и конец которых выделить вставкой контрольных меток.

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.

Недостаток этих потоковых шифров – распространение ошибок, так как искажение одного бита в процессе передачи шифртекста приведет к искажению нескольких битов гаммы и, соответственно, расшифрованного сообщения.

Потоковый шифр максимально должен имитировать одноразовый блокнот. В соответствии с этим ключ должен по своим свойствам максимально походить на случайную числовую последовательность.

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.

*Генератор ПСП на основе RSA* устроен следующим образом. Последовательность генерируется с использованием соотношения

*xt* ≡ (*xt –* 1)*е* mod *n*.

Начальными параметрами служат *n*, большие простые числа *p* и *q* (причем *n* = *pq*), целое число *е*, взаимно простое с произведением (*р* – 1)(*q* – 1), а также некоторое случайное начальное значение *x*0.

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 – 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**

**Задание 1.** Разработать авторские многооконные приложения в соответствии с целью лабораторной работы. При этом можно воспользоваться готовыми библиотеками либо программными кодами, реализующими заданные алгоритмы.

В начале, нужно было разработать приложение, реализующее генерацию ПСП на основе линейного алгоритма RSA. Начальными параметрами служат *n*, большие простые числа *p* и *q* (причем *n* = *pq*), целое число *е*, взаимно простое с произведением (*р* – 1)(*q* – 1), а также некоторое случайное начальное значение *x0*. Выходом генератора на *t*-м шаге является младший бит числа *xt*.

Функция generatePrime генерирует случайное простое число с заданной длиной бит.

function generatePrime(bits) {

    while (true) {

        const bytes = crypto.randomBytes(bits / 8);

        let p = BigInt('0x' + bytes.toString('hex')) | 1n;

        if (isProbablePrime(p))

            return p;

    }

}

Листинг 8.1 – Функция generatePrime

Для реализации алгоритма генерации псевдослучайных последовательностей с помощью алгоритма RSA был создан метод generateRSA.

function generateRSA(bits, length) {

    const p = generatePrime(bits);

    const q = generatePrime(bits);

    const n = p \* q;

    const phi = (p - 1n) \* (q - 1n);

    let e;

    do {

        e = generatePrime(bits);

    } while (e >= phi || gcd(e, phi) !== 1n);

    let x = BigInt('0x' + crypto.randomBytes(bits / 8).toString('hex'));

    const bitsArr = [];

    for (let i = 0; i < length; i++) {

        x = modPow(x, e, n);

        bitsArr.push((x & 1n) === 1n ? '1' : '0');

    }

    return {

        bitString: bitsArr.join(''),

        decimalValue: BigInt('0b' + bitsArr.join('')).toString(),

        params: {

            p: p.toString(),

            q: q.toString(),

            n: n.toString(),

            e: e.toString(),

            x0: x.toString()

        }

    };

}

Листинг 8.2 – Функция generateRSA

Результат работы алгоритма, создающий 16-битное псевдослучайное число, представлен на рисунке 8.1.

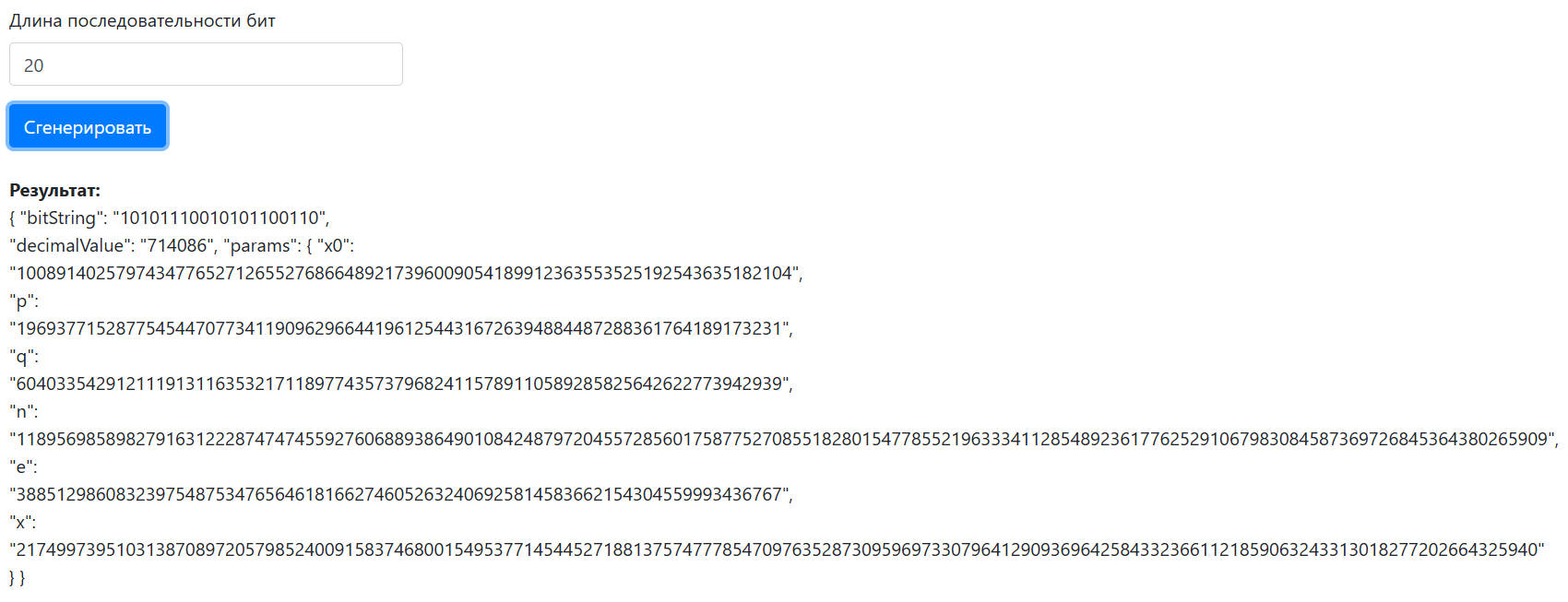


Рисунок 8.1 – Вывод алгоритма генерации ПСП

Далее, нужно было разработать приложение, реализующее алгоритм RC4.

Функция RC4encrypt принимает исходный текст и шифрует его.

function RC4encrypt(data) {

    const n = 6;

    const key = [20, 21, 22, 23, 60, 61];

    const m = Math.pow(2, n);

    let x = 0;

    let y;

    const box = [...Array(m).keys()];

    const startTime = performance.now();

    for (let i = 0; i < m; i++) {

        x = (x + box[i] + key[i % key.length]) % m;

        [box[i], box[x]] = [box[x], box[i]];

    }

    const endTime = performance.now();

    const generationTime = (endTime - startTime).toFixed(4);

    x = y = 0;

    const out = [];

    for (const char of data) {

        x = (x + 1) % m;

        y = (y + box[x]) % m;

        [box[x], box[y]] = [box[y], box[x]];

        const k = box[(box[x] + box[y]) % m];

        out.push(String.fromCharCode(char.charCodeAt(0) ^ k));

    }

    return {

        result: out.join(''),

        generationTime

    };

}

Листинг 8.3 – Функция RC4encrypt

Результат шифрования текста «Kozeka Elizaveta Maksimovna» (размер блока – 6, ключ – 20, 21, 22, 23, 60, 61) представлен на рисунке 8.2.

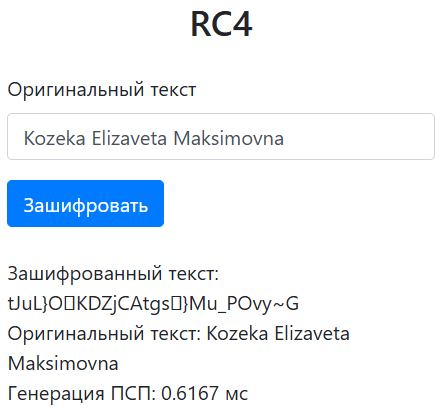


Рисунок 8.2 – Шифрование и дешифрование текста c помощью RC4

Кроме того, приложение выполняет оценку скорости генерации ПСП. Можно заметить, что при значении ключа [20, 21, 22, 23, 60, 61] ПСП сгенерировалась всего за 0.6 мс.

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