МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программирование интернет-приложений

**ОТЧЁТ К ЛАБОРАТОРНОЙ РАБОТЕ №8:**

по дисциплине «Криптографические методы защиты информации»

Исполнитель

студентка 3 курса группы 5 Шкода Кристина Михайловна

(Ф.И.О.)

Руководитель работы преподаватель Савельева М. Г.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Минск 2023

**Исследование потоковых шифров**

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).

Синхронные потоковые шифры (СПШ) — шифры, в которых поток ключей генерируется независимо от открытого текста и шифротекста.

Самосинхронизирующиеся потоковые шифры (асинхронные потоковые шифры (АПШ)) — шифры, в которых ключевой поток создаётся функцией ключа и фиксированного числа знаков шифротекста.

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

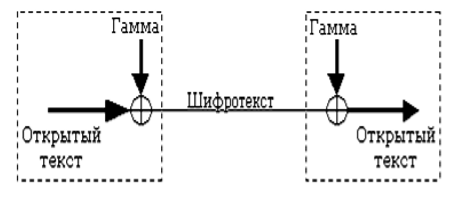


Рисунок 1 – Схема потокового шифра

Случайные числа (последовательности) – последовательность элементов, каждый из которых не может быть предсказан (вычислен) только на основе знания предшествующих ему элементов данной последовательности.

Псевдослучайные числа – последовательность элементов, полученная в результате выполнения некоторого алгоритма и используемая в конкретном случае вместо последовательности случайных чисел.

Алгоритм **линейного конгруэнтного генератора**, описывается следующим рекуррентным соотношением:

*xt+*1 = (*a× xt + c*) mod *n*.

*а*, *с* и *n* – константы. Период такого генератора (период ПСП) не превышает n.

**Генератор ПСП на основе регистров сдвига.** РС с линейной обратной связью (РСЛОС) состоит из двух частей: собственно РС и функции обратной связи. На рис. 2 представлена общая схема РС с линейной обратной связью. Функция обратной связи реализуется с помощью сумматоров сложения по модулю два.

РСЛОС строятся на основе примитивных порождающих полиномов (многочленов), которые мы подробно анализировали при изучении циклических помехоустойчивых кодов. Если многочлен является неприводимым, то период ПСП при ненулевом начальном условии (ненулевом состоянии) регистра будет максимально возможным: 2*L* – 1.

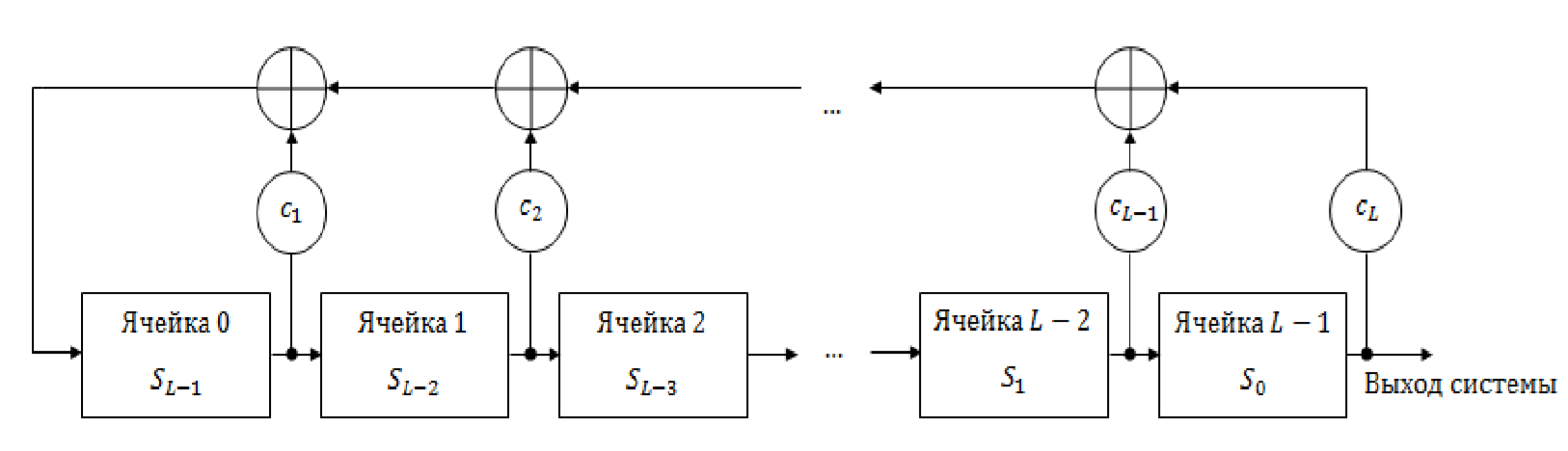


Рисунок 2 – Общая схема регистра сдвига с линейной обратной связью

Генератор псевдослучайных чисел на основе **алгоритма RSA** разработан для систем асимметричного зашифрования/расшифрования. 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* = *p× q*), целое число е, взаимно простое с произведением   
(*р –* 1) *×* (*q –*1), а также некоторое случайное начальное значение, *x0*. Выходом генератора является на *t*-м шаге является младший бит числа *xt*. Безопасность генератора опирается на сложности взлома алгоритма RSA, т. е. на разложении числа n на простые сомножители.

**Алгоритм BBS** или генератором на основе квадратичных вычетов. Начальное значение *x0* генератора вычисляется на основе соотношения:

*x0*= *x2* mod *n*

где n, как и в генераторе на основе RSA, является произведением простых чисел *p* и *q*, однако в нашем случае эти простые числа должны быть сравнимы с числом 3 по модулю 4, т. е. при делении *p* и *q* на 4 должен получаться одинаковый остаток: 3; число x должно быть взаимно простым с *n*; число *n* называют числом Блюма. Выходом генератора на *t*-м шаге является младший бит числа *xt*:

*xt* = (*xt-1*)2 mod *n*.

**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 бит.

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.

Начало формы

**Ход работы**

Приложени было разработано на языке JS.

**Задание №1**

В данном задании реализован метод генерации ПСП на основе метода BBS, где *n*=256. *p* и *q* были выбраны так, что *n* = *p* × *q*. В данном случае можно выбрать, например, следующие числа: *p* = 17 и *q* = 15. В функции *getGenerate()* мы задаем начальное значение x равным 7, константы *p* и *q* равными 17 и 15 соответственно, и *n* равным 256. Затем мы генерируем 10 псевдослучайных чисел, возводя значение *x* в квадрат по модулю *n* и беря младший бит. Полученные числа выводятся в консоль. Код и вывод функции представлены на рисунках 3, 4 соответственно.

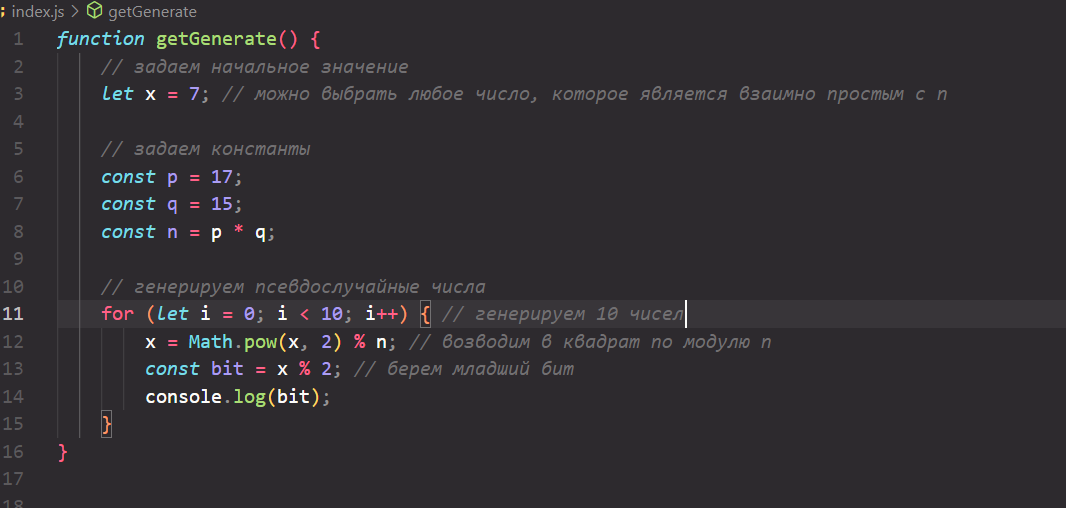


Рисунок 3 – Реализация метод генерации ПСП на основе метода BBS

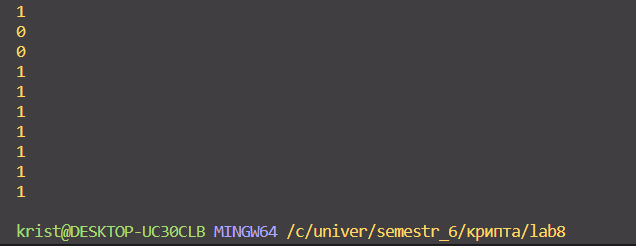


Рисунок 4 – Пример вывода

**Задание №2**

В данном задании представлена реализация алгоритма RC4 при *n* = 8, а ключи = [61, 60, 23, 22, 21, 20], а также дополнительно выполнена оценка скорости выполнения операций генерации ПСП.

*Алгоритм* *RC4* состоит из двух этапов. На первом, подготовительном этапе производится *инициализация* таблицы замен *S*. На втором, основном этапе вычисляются псевдослучайные числа.

Посмотрим, как инициализируется *таблица* S. Вначале она заполняется последовательно числами от 0 до 15. *Ключ* представляется в виде последовательности 4-битовых слов, которыми заполняется другой *массив* *K*, такого же размера, как *S*. Если *ключ* оказался короче, чем надо, он повторяется нужное число раз. Затем выполняются следующие действия (алгоритм 1):

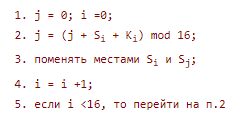


Рисунок 5 – Алгоритм 1

В результате выполнения этого алгоритма производится начальное заполнение таблицы замен *S*, причем это начальное перемешивание значений производится в зависимости от секретного ключа.

После того, как таблица *S* подготовлена, можно начинать генерацию случайных n-битовых слов. Для этого счетчикам *i* и *j* присваивается начальное значение 0. Затем для получения каждого нового значения *zi* выполняются следующие действия (алгоритм 2):

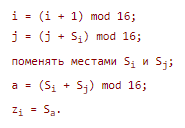
****

Рисунок 6 – Алгоритм 2

Реализация и вывод представлены на рисунках 5, 6 соответственно.

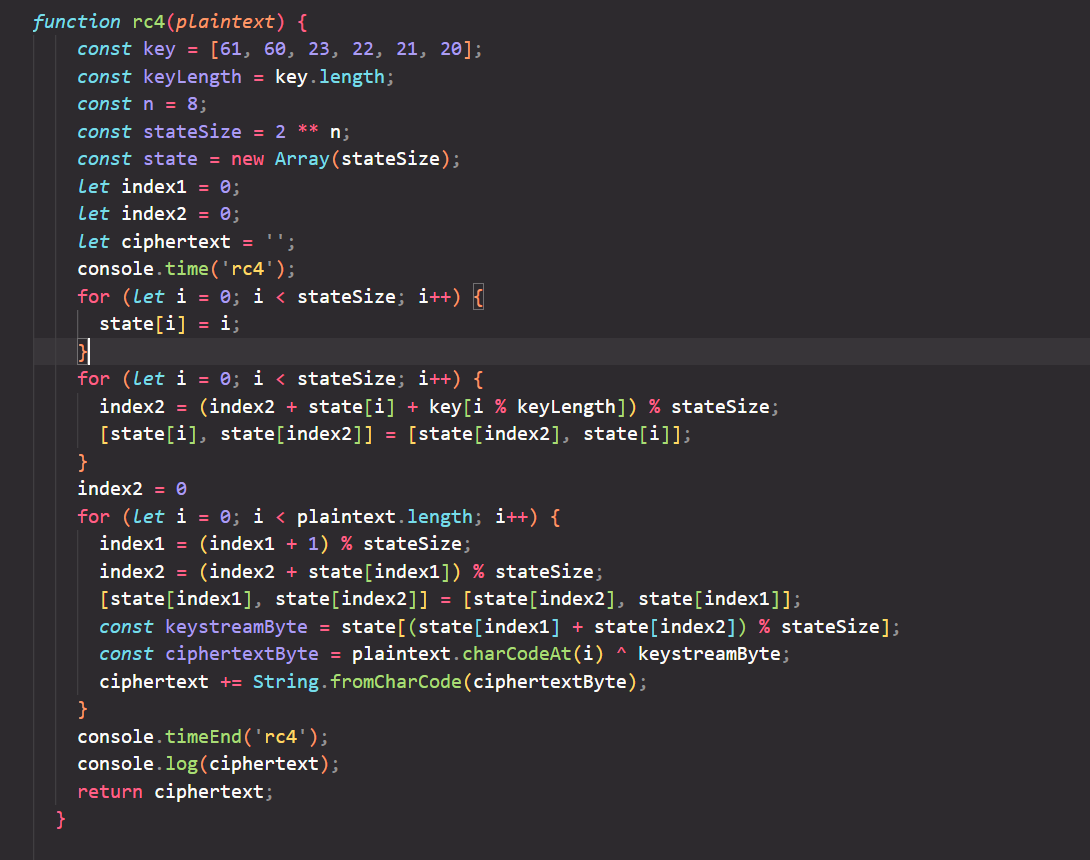


Рисунок 7 – Реализация



Рисунок 8 – Пример вывода

**Вывод:** таким образом, в данной лабораторной работе я изучила и приобрела практические навыки разработки и использования приложений для реализации потоковых шифров, а именно генерации ПСП при помощи алгоритма BBS, шифрования/расшифрования сообщений при помощи алгоритма RC4.