МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования

«БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Направление специальности Информационные системы и технологии

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

**ПО ДИСЦИПЛИНЕ**

**«ЗАЩИТА ИНФОРМАЦИИ**

**И НАДЕЖНОСТЬ ИНФОРМАЦИОННЫХ СИСТЕМ»**

Выполнил: студент 3 курса 1 группы

Кашперко Василиса Сергеевна

Проверил: ассистент

Нистюк Ольга Александровна

Минск 2023

**Лабораторная работа №6**

**Исследование потоковых шифров**

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

**Теоретические сведения**

***Классификация и общие свойства потоковых шифров***

Потоковый шифр (иногда говорят «поточный») – симметричный шифр, преобразующий каждый символ открытого текста в символ шифрованного , 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.

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

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

Синхронные потоковые шифры (СПШ) характеризуются тем, что поток ключей генерируется независимо от открытого текста и шифртекста. Главное свойство СПШ – нераспространение ошибок. 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. 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.

***Генераторы ключевой информации***

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

Действующий в Беларуси стандарт СТБ 34.101.47–2017 определяет базовые понятия в рассматриваемой предметной области:

– 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.

***Линейный конгруэнтный генератор***

Часто используемый алгоритм генерирования ПСП реализуется на основе так называемого линейного конгруэнтного генератора, описываемого следующим рекуррентным соотношением:  


где и – 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 псевдослучайных чисел на основе алгоритма RSA***

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



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 .

Выходом генератора на *t*-м шаге является младший бит числа .

Безопасность генератора опирается на сложность взлома алгоритма RSA, т. е. на разложение числа *n* на простые сомножители.

**Ход работы**

По условию лабораторной работы нам нужно разработать приложение (или несколько приложений), которое позволяет выполнить 2 задачи в соответствии с вариантом (выполнен вариант 3):

* генерация псевдослучайной последовательности на основе алгоритма BBS;
* зашифрование и расшифрование строки с помощью алгоритма RC4 с параметрами: n=8; ключ = {43, 45, 100, 21, 1}.

Также разработанное приложение позволяет оценить скорость выполнения операций генерации ПСП.

В качестве шифруемого сообщения выбраны мои фамилия и имя «Kashperko Vasilisa».

***Генератор псевдослучайных чисел на основе алгоритма BBS***

Алгоритмом BBS (от фамилий авторов: L. Blum, M. Blum, M. Shub) или генератором на основе квадратичных вычетов.

Начальное значение x0 генератора вычисляется на основе соотношения:

****

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 :



***Генерация ПСП на основе алгоритма BBS***

Для реализации генерации псевдослучайной последовательности (ПСП) на основе алгоритма BBS была разработана функция, представленная на рисунке 1.

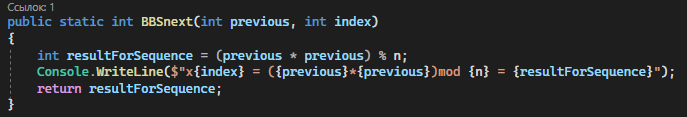


Рисунок 1 – Реализация генерации числа ПСП

Данная функция вычисляет каждое число последовательности отдельно по формуле квадратичных вычетов:

,

где xt – вычисляемое значение генератора,

xt-1 – предыдущий элемент ПСП,

n = p\*q (число Блюма), причем простые числа p и q должны быть сравнимы с числом 3 по модулю 4 (например, возьмем такие числа как 11 и 23).

Для вычисления начального значения генератора x0 используется то же соотношение, но вместо предыдущего элемента ПСП xt-1в функцию подается х=5 – число, взаимно простое с числом Блюма n (было выбрано самостоятельно).

В главной функции мы циклично вызываем функцию выше, последовательно заполняя массив, который хранит числа генерируемой ПСП.

Также мы оценили скорость выполнения генерации ПСП. Вычисленное время составило 113 мс (рисунок 2).

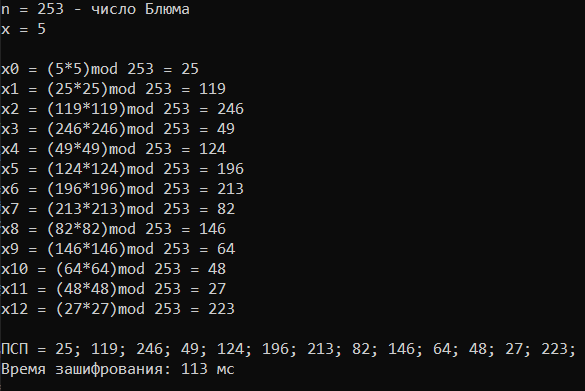


Рисунок 2 – Результат генерации ПСП на основе алгоритма BBS

Это подтверждает, что алгоритм является сравнительно медленным. Для ускорения можно использовать не последний бит числа, а несколько последних битов. Однако понятно, что при этом алгоритм является менее криптостойким.

***Реализация алгоритма RC4***

Для решения поставленной задачи был разработан отдельный класс «RC4».

При создании экземпляра данного класса вызывается конструктор, код которого приведен на рисунке 3.

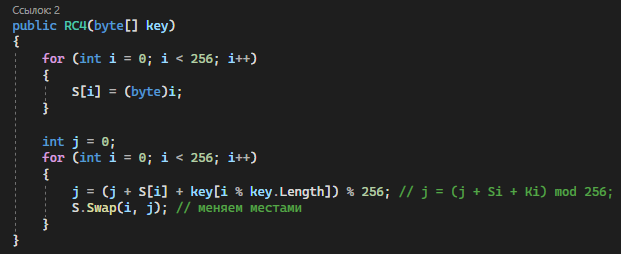


Рисунок 3 – Конструктор RC4

Входными данными у нас выступает массив байт. Ключ тоже выступает в качестве входных данных.

Чтобы зашифровать сообщение "Kashperko Vasilisa" с ключами { 43, 45, 100, 21, 1 }:

В соответствии с вариантом n = 256.

Сначала S-блок пополняется линейно: 0,1…255.

Затем заполняется секретным ключом другой массив [256].

Если необходимо, ключ повторяется многократно чтобы заполнить весь массив K0…K255.

Далее массив S перемешивается путем перестановок, определяемых ключом.

Для зашифрования и расшифрования исходного сообщения используется функция Encode, представленная на рисунке 4.

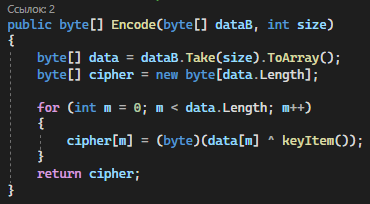


Рисунок 4 – Реализация функции Encode

В параметры данной функции передается массив исходных байтов и их размер.

Для каждого исходного байта запрашивается текущий байт ключа, после чего они объединяются при помощи XOR для получения 8-битного зашифрованного сообщения.

В теле функции выше также вызывается функция ItemKey, представленная на рисунке 5.

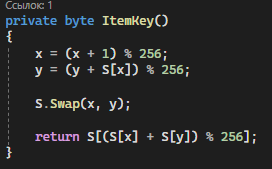


Рисунок 5 – Реализация функции ItemKey

Эта часть алгоритма называется генератором ПСП при n=8 (в соответствии с вариантом). При каждом вызове функция отдает следующий байт ключевого потока.

Проведем оценку времени выполнения: RC4 затрачивает меньшее время для шифрования и дешифрования, чем BBS. Зашифрования сообщения происходит дольше, чем расшифрование.

Результат выполнения данного консольного приложения представлен на рисунке 6.

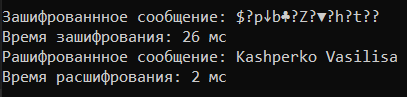


Рисунок 6 – Результат реализации алгоритма RC4

**Вывод:** в ходе лабораторной работы были приобретены практические навыки разработки и использования приложений для реализации потоковых шифров. Было разработано приложение для реализации BBS-алгоритма генерации.

**Контрольные вопросы**

*1. В чем состоит особенность потоковых шифров?*

Потоковые шифры могут шифровать данные в режиме реального времени, обрабатывая их по мере поступления и не зависят от размера данных. Они используют ключ, который генерирует последовательность псевдослучайных битов, комбинирующихся с данными для шифрования и дешифрования потоковых данных, таких как аудио и видео. Это позволяет использовать потоковые шифры для защиты телефонных разговоров и других потоков данных без необходимости дополнительных ресурсов для обработки больших объемов данных.

*2. В чем состоят преимущества и недостатки синхронных и асинхронных потоковых шифров?*

Синхронные потоковые шифры имеют высокую скорость шифрования и низкую задержку при передаче данных, но отсутствует механизм для обнаружения ошибок в данных, и низкий уровень защиты от атак на основе известного или выбранного шифротекста.

Асинхронные потоковые шифры имеют высокий уровень защиты данных благодаря использованию открытого и закрытого ключей и механизм обнаружения ошибок в данных благодаря использованию контрольной суммы, но низкая производительность, сложность ключевого управления, уязвимость к атакам по времени, неточность аутентификации и зависимость от сложных математических операций.

*3. Какими свойствами должен обладать генератор псевдослучайных чисел для использования в криптографических целях?*

* Непредсказуемость;
* Длинный период;
* Статистическая равномерность;
* Невоспроизводимость;
* Эффективность;
* Криптографическая стойкость.

*4. Дать характеристику линейным конгруэнтным генераторам. Области их применения.*

Линейные конгруэнтные генераторы (ЛКГ) — это простой и распространенный тип генераторов псевдослучайных чисел, использующий рекуррентную формулу с начальным значением, множителем и приращением. Хотя они быстры и легко реализуемы, у них короткий период и не подходят для криптографических приложений. Однако они могут быть полезны в играх, моделировании, тестировании программного обеспечения и других приложениях, где не требуется высокая степень случайности.

*6. Представить общую структурную схему генератора ПСП на основе регистров сдвига с линейной обратной связью.*

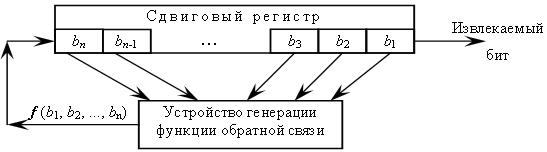


Рисунок 7 – Сдвиговый регистр с обратной связью

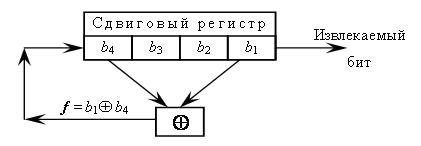


Рисунок 8 – Пример 4-разрядного линейного сдвигового регистра

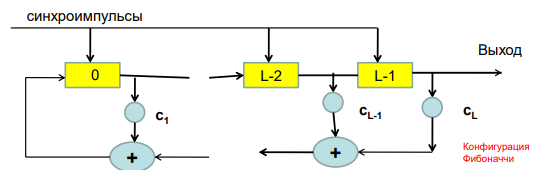
****

Рисунок 9 – Схема генератора ПСП

*8. Определить первые 12 бит ПСП, задаваемого формально в виде чисел 5410, если начальные состояния ячеек (слева направо) соответствуют последовательности 10101.*

Описание ПСП в виде числа 5410 означает, что следующее значение ПСП вычисляется по формуле:

xn = (xn-1 \* 5 + 4) mod 10,

где xn - текущее значение ПСП,

xn-1 - предыдущее значение ПСП,

mod 10 - означает взятие остатка от деления на 10.

Начальные состояния ячеек заданы последовательностью 10101, что означает, что первые пять значений ПСП равны: x0 = 1, x1 = 5, x2 = 7, x3 = 6, x4 = 0.

Для определения первых 12 бит ПСП нужно представить xi в двоичной системе счисления, а затем объединить полученные биты в одну последовательность.

Полученная последовательность будет первыми 12 битами ПСП: 1 = 0001, 5 = 0101, 7 = 0111, 6 = 0110, 0 = 0000.

Объединением первые 12 бит – 0001 0101 0111.

Ответ: 0001 0101 0111.

*9. Как устроен генератор ПСП на основе RSA? На чем основана криптостойкость реализуемого алгоритма?*

Генератор псевдослучайных чисел (ПСП) на основе RSA (Rivest–Shamir–Adleman) основан на вычислительной сложности факторизации больших простых чисел.

Генератор псевдослучайных чисел на основе RSA использует сложность факторизации больших простых чисел для обеспечения криптостойкости.

Он работает по формуле, которая использует два случайных числа, включая взаимно простое с произведением выбранных простых чисел.

Хотя атака на генератор прямым подбором неэффективна, анализ последовательности значений может стать уязвимостью при недостаточно большой длине ключа. Для обеспечения криптостойкости необходимо выбирать достаточно большие простые числа.

*10. Вычислить х1, х5, х9, х11 по методу генерации псевдослучайных чисел BBS, если p = 11, q = 19, х = 3.*

1. Вычислим произведение p и q:

N = p*q =* 11*\**19 = 209.

1. Вычислить значение x0:

x0= х2 mod N: x0 = 3^2 mod 209 = 9.

1. И далее, для i от 1 до 11:
   1. Вычислить значение xi:

xi = x(i-1)2 mod N.

* 1. Получить очередное псевдослучайное число zi = xi mod 2.

x1 = x02 mod N = 92 mod 209 = 81 mod 209 = 81

x2 = x12 mod N = 812 mod 209 = 165 mod 209 = 165

x3 = x22 mod N = 1652 mod 209 = 86 mod 209 = 86

x4 = x32 mod N = 862 mod 209 = 68 mod 209 = 68

x5 = x42 mod N = 682 mod 209 = 104 mod 209 = 104

x6 = x52 mod N = 1042 mod 209 = 20 mod 209 = 20

x7 = x62 mod N = 202 mod 209 = 109 mod 209 = 109

x8 = x72 mod N = 1092 mod 209 = 42 mod 209 = 42

x9 = x82 mod N = 422 mod 209 = 141 mod 209 = 141

x10 = x92 mod N = 1412 mod 209 = 127 mod 209 = 127

x11 = x102 mod N = 1272 mod 209 = 45 mod 209 = 45

Для выполнения задания выпишем значения: x1 = 81, x5 = 104, x9 = 141, x11 = 45.

*11. Пояснить базовый алгоритм, реализованный в шифре RC4.*

RC4 - поточный шифр, используемый для шифрования данных в разных приложениях и протоколах. Он использует генератор псевдослучайных чисел для создания ключевого потока байтов для шифрования данных.

Генератор псевдослучайных чисел основан на алгоритме генерации ключевого потока, который включает инициализацию массива S, перестановку элементов массива с использованием ключа K и генерацию ключевого потока байтов.

Это позволяет выполнить операцию XOR с открытым текстом, чтобы получить зашифрованный текст, или с зашифрованным текстом, чтобы получить исходный текст.

Однако RC4 имеет недостатки и уязвимости, связанные с генерацией ключевого потока и инициализацией массива S, что может привести к утечке данных и нарушению конфиденциальности. Рекомендуется использовать более современные и безопасные алгоритмы шифрования.

*12. Пояснить принципы формирования истинных случайных последовательностей, основанных на «естественных случайностях»*

Формирование истинных случайных последовательностей, основанных на «естественных случайностях», основывается на использовании естественных процессов и явлений, которые могут быть использованы для создания случайных битовых последовательностей.

Для получения случайных битовых последовательностей используются естественные процессы, такие как радиоактивный распад, шумы, атмосферные шумы и термальный шум.

Полученные данные подвергаются фильтрации и тестированию для исключения систематических ошибок и зависимостей между битами.

Эти истинные случайные последовательности могут быть использованы для создания криптографических ключей и других приложений, требующих случайных битовых последовательностей.