Министерство образования Республики Беларусь

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет к лабораторной работе № 5:

«Исследование потоковых шифров»

Выполнила:

студентка 3 курса 2 группы

Шастовская М. С.

Вариант: 12

Преподаватель:

Сазонова Д.В.

Минск 2023

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

**Задачи:**

* Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости потоковых шифров.
* Разработать приложение для реализации указанных преподавателем методов генерации ключевой информации и ее использования для потокового зашифрования/расшифрования.
* Выполнить анализ криптостойкости потоковых шифров.
* Оценить скорость зашифрования/расшифрования реализованных шифров.
* Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

1 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. Один из методов борьбы с рассинхронизацией – разбить отрытый текст на отрезки, начало и конец которых выделить вставкой контрольных меток (специальных маркеров).

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.

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.

Ключевые последовательности (случайные последовательности (СП), либо псевдослучайные последовательности (ПСП)) вырабатываются специальными блоками систем потокового шифрования – генераторами. В Беларуси в настоящее время действует стандарт СТБ 34.101.47–2017 «Информационные технологии и безопасность. Алгоритмы генерации псевдослучайных чисел».

Указанный стандарт 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.

Достаточно распространенным является использование регистров сдвига (РС) в качестве генераторов ПСП в силу простоты реализации на основе цифровой логики. РС с линейной обратной связью (РСЛОС) состоит из двух частей: собственно РС и функции обратной связи.

Алгоритм RSA разработан для систем асимметричного зашифрования/расшифрования и будет более детально рассмотрен с практической точки зрения ниже.

Генератор же 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 x0. Выходом генератора на t-м шаге является младший бит числа xt.

Генератор 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.

В основе алгоритма RC4 – 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 (каждый размером в одно слово; обозначим их i и j) с нулевыми начальными значениями. В основе вычислений лежит операция по mod 2n.

2 Практическая часть

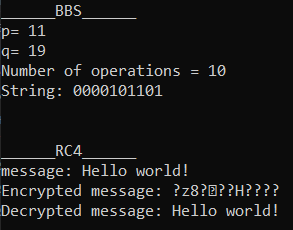


Рисунок 2.1 – Результат шифрования/дешифрования исходного сообщения алгоритмом BBC и RC4

На данном скриншоте отображен процесс работы программы с шифрованием и дешифрованием исходного сообщения методом BBC и RC4.

Основные функции программы представлены на рисунке 2.2 и рисунке 2.3.

public byte[] Encode(byte[] dataB, int size)

{

byte[] data = dataB.Take(size).ToArray();

byte[] cipher = new byte[data.Length];

for (int m = 0; m < data.Length; m++)

{

cipher[m] = (byte)(data[m] ^ keyItem());

}

return cipher;

}

public byte[] Decode(byte[] dataB, int size)

{

return Encode(dataB, size);

}

// Pseudo-Random Generation Algorithm

// Генератор псевдослучайной последовательности

private byte keyItem()

{

x = (x + 1) % 256;

y = (y + S[x]) % 256;

S.Swap(x, y);

return S[(S[x] + S[y]) % 256];

}

Рисунок 2.2 – Основные функции реализации RC4

public bool PrimeNumberIsValid(BigInteger primeNumber)

{

if (primeNumber % 4 == 3) return true;

else return false;

}

public BigInteger GetRandomNumber() //Random number from range <Int32.minValue;Int32.maxValue)

{

Random random = new Random();

BigInteger randomNumber;

if (n > (BigInteger)Int32.MaxValue) randomNumber = (BigInteger)random.Next(1, Int32.MaxValue);

else randomNumber = (BigInteger)random.Next(1, (Int32)n);

return randomNumber;

}

public string GetSeries(int LengthOfSeries)

{

string series = string.Empty;

string binaryValue = string.Empty;

BigInteger[] xValues = new BigInteger[LengthOfSeries];

BigInteger randomNumber = GetRandomNumber();

xValues[0] = (randomNumber \* randomNumber) % n;

if (xValues[0] % 2 == 1) series += 1;

else series += 0;

for (int i = 1; i < LengthOfSeries; i++)

{

xValues[i] = (xValues[i - 1] \* xValues[i - 1]) % n;

if (xValues[i] % 2 == 1) series += 1;

else series += 0;

}

return series;

}

Рисунок 2.3 – Основные функции реализации BBS.

Вывод

В ходе выполнения лабораторной работы были изучены и приобретены практические навыки разработки и использования приложений для реализации потоковых шифров.