МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Отчет по лабораторной работе №6**

**ИССЛЕДОВАНИЕ ПОТОКОВЫХ ШИФРОВ**

Выполнил:

Cтудент 3 курса 1 группы

Парибок И. А.

Вариант 5

Минск 2023

**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 шифрования/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 c i, 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. В общем виде схема потокового шифра изображена на рис. 6.1.

Изображение выглядит как диаграмма

Автоматически созданное описание

Риc. 1 – Схема потокового шифра

**Синхронные потоковые шифры (СПШ)** характеризуются тем, что поток ключей генерируется независимо от открытого текста и шифротекста. Главное свойство СПШ – нераспространение ошибок. Ошибки отсутствуют, пока работают синхронно шифровальное и дешифровальное устройства отправителя и получателя информации. Один из методов борьбы с рассинхронизацией – разбить открытый 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. 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.

**Генераторы ключевой информации**

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. В Беларуси в настоящее время действует стандарт СТБ 34.101.47–2017 «Информационные технологии и безопасность. Алгоритмы генерации псевдослучайных чисел» [32]. Стандарт устанавливает криптографические алгоритмы генерации псевдослучайных чисел. Алгоритмы стандарта могут применяться для построения ключей, синхропосылок, одноразовых паролей, других непредсказуемых или уникальных параметров криптографических алгоритмов и протоколов. Стандарт применяется при разработке, испытаниях и эксплуатации средств криптографической защиты информации.

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.

**Линейный конгруэнтный генератор**

Часто используемый алгоритм генерирования (программно или аппаратно) ПСП реализуется на основе так называемого линейного конгруэнтного генератора, описываемого следующим рекуррентным соотношением:  


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*.

Если параметры a, b и c выбраны правильно, то генератор будет порождать случайные числа с максимальным периодом, равным c. При программной реализации значение с обычно устанавливается равным 2b – 1 или 2b , где b – длина слова в битах.

Достоинством линейных конгруэнтных генераторов псевдослучайных чисел является их простота и высокая скорость получения псевдослучайных значений. Линейные конгруэнтные генераторы находят применение при решении задач моделирования и математической статистики, однако в криптографических целях их нельзя рекомендовать к использованию, так как специалисты по криптоанализу научились восстанавливать всю последовательность ПСП по нескольким ее значениям.

Генератор практически не используется в криптографии в силу низкой криптостойкости. Тем не менее он полезен для решения задач моделирования.

Комбинации нескольких (чаще двух) линейных конгруэнтных генераторов позволяют значительно повысить период ПСП. Б. Шнайер, например, приводит данные о том, как на 32-разрядных ПК реализовать генератор в виде комбинации двух, каждый из которых обеспечивает период соответственно 231 – 85 и 231 – 249, а комбинированный генератор позволяет достичь периода ПСП, равного произведению указанных чисел [5].

**Генератор ПСП на основе регистров сдвига**

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. На рис. 6.2 представлена общая схема РС с линейной обратной связью. Функция обратной связи реализуется с помощью сумматоров сложения по модулю два (элементы XOR; на рис. 1 обозначены в виде кружочков со знаком сложения).

**Изображение выглядит как диаграмма

Автоматически созданное описание**

Рис.1 – Общая схема регистра сдвига с линейной обратной связью

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: 2L – 1.

**Генератор псевдослучайных чисел на основе алгоритма RSA**

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 псевдослучайных чисел на основе алгоритма BBS**

Широкое распространение получил алгоритм генерации ПСП, называемый алгоритмом BBS (от фамилий авторов: L. Blum, M. Blum, M. Shub) или генератором на основе квадратичных вычетов. Для целей криптографии этот метод предложен в 1986 г. 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.*

**Практическая часть**

На рисунках 1–2 представлены варианты заданий для первого и второго приложения. Практическое приложение должно реализовывать генерацию ПСП в соответствии с вариантом. Приложение 2 должно реализовывать алгоритм RC4 в соответствии с вариантом, а также дополнительно выполнять оценку скорости выполнения операций генерации ПСП. В качестве шифруемого сообщения может быть выбран произвольный текст.

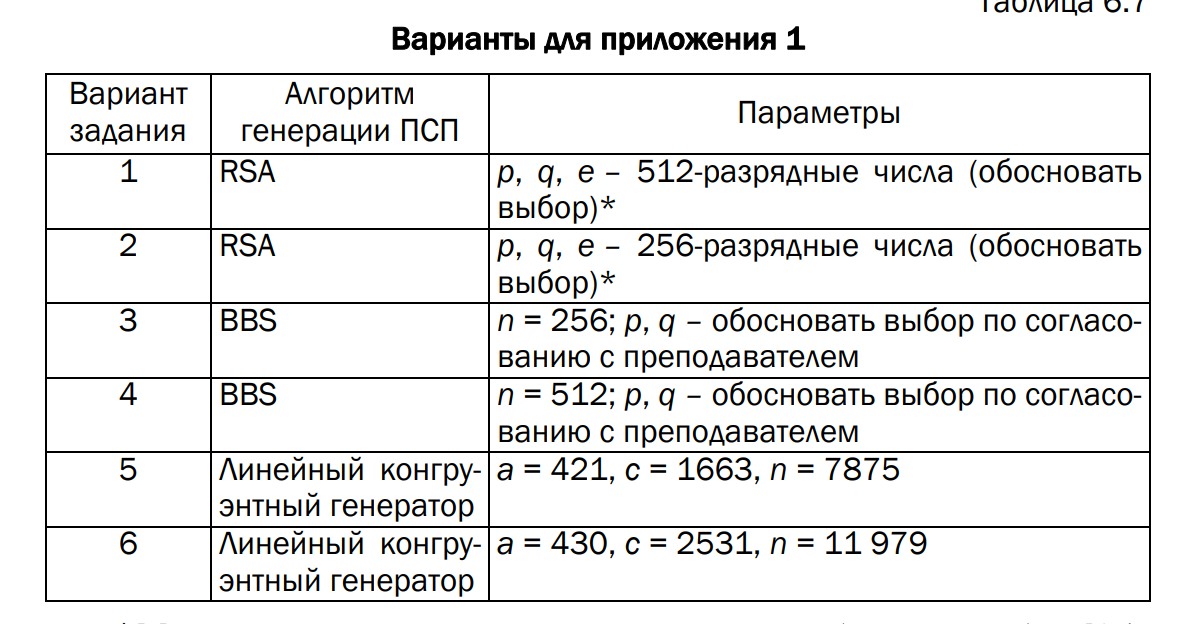


Рисунок 1 – Варианты заданий для первого приложения

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 2 – Варианты заданий для второго приложения

ПСП означает "последовательность псевдослучайных чисел" (англ. PRN - pseudorandom number sequence). Это последовательность чисел, которые могут быть использованы для шифрования сообщений, генерации случайных чисел или для других целей, где требуется случайная информация.

ПСП являются псевдослучайными, потому что они генерируются алгоритмом, который использует начальные значения, такие как "семя" или "ключ", для создания последовательности чисел, которые выглядят случайными. Однако эти числа в действительности являются детерминированными и могут быть повторно воспроизведены, если известны начальные значения.

На рисунке 3 представлен код реализующий алгоритм линейного конгруэнтного генератора (Linear Congruential Generator - LCG), который генерирует последовательность псевдослучайных чисел на основе заданных начальных параметров.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Реализация ПСП

Параметры LCG определены внутри функции **lcg()**. Они включают начальное значение **seed**, константы **a**, **c** и **n**. Значение **x** инициализируется начальным значением **seed**, а затем на каждой итерации цикла генерируется новое случайное число, используя формулу **x = (a \* x + c) % n**.

Функция **lcg()** принимает два аргумента: **seed** - начальное значение для генератора LCG, и **count** - количество случайных чисел, которое необходимо сгенерировать. Функция возвращает список сгенерированных чисел.

Наконец, после определения функции **lcg()**, код вызывает эту функцию, чтобы сгенерировать 10 псевдослучайных чисел на основе начального значения 123, и выводит результаты в консоль.

Вывод проиложения представлен на рисунке 4. Каждое полученное число находится в диапазоне от 0 до 1, что может быть полезно для генерации равномерно распределенных случайных чисел, например, для моделирования ряда случайных событий.



Рисунок 4 – Последовательность псевдослучайных чисел

Реализация второго приложения представлена в приложение А. Этот код реализует алгоритм RC4 - поточного шифра.

Класс **RC4** имеет два метода: **\_\_init\_\_** и **generate\_keystream**. Метод **\_\_init\_\_** инициализирует объект класса, используя ключевую последовательность **key**.

Алгоритм RC4 использует внутреннее состояние **S** (которое инициализируется в **\_\_init\_\_()**), состоящее из 256 байтов, и два индекса **i** и **j**, которые используются для генерации ключевого потока.

Метод **generate\_keystream** использует текущее состояние **S** и индексы **i** и **j** для генерации ключевого потока длины **n**. Он изменяет внутреннее состояние **S** на каждой итерации, чтобы гарантировать, что ключевой поток будет псевдослучайным.

После этого код определяет ключ **key** и шифрует исходный текст **plaintext** с помощью RC4. Для этого сначала создается экземпляр RC4 с ключем **key**. Затем используется метод **generate\_keystream**, чтобы сгенерировать ключевую последовательность той же длины, что и сообщение, и затем применяется операция **XOR** между каждым символом исходного сообщения и соответствующим символом ключевой последовательности, чтобы зашифровать сообщение.

Затем, используя тот же ключ, исходное сообщение расшифровывается, повторно использованием объекта RC4 и той же ключевой последовательности. Расшифрованное сообщение выводится в консоль.

И наконец, в этом коде используется функция **time()** из модуля **time**, чтобы измерить время, потраченное на генерацию ключевой последовательности длины 1 МБ. Результат определяется в секундах и выводится в консоль в виде строки (рис 5).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 –Результат работы второго приложения

**Вывод:**

В ходе выполнения лабораторной работы были реализованы два алгоритма генерации псевдослучайных чисел - LCG и RC4. Эти алгоритмы могут использоваться для защиты данных при передаче по сети, генерации ключей шифрования, генерации случайных чисел для тестирования программного обеспечения и других задач.

LCG генерирует числа, которые выглядят случайными и находятся в диапазоне от 0 до 1. Однако эти числа не являются идеально случайными, и злоумышленник может восстановить начальное значение генератора, используя некоторые из сгенерированных чисел. Поэтому LCG не рекомендуется для использования в криптографических приложениях.

RC4 — это более современный алгоритм, который генерирует байты, которые выглядят случайными и могут использоваться для зашифрования данных. RC4 использует начальное значение - ключ, который является общим для отправителя и получателя, что делает его более надежным в защите переданных данных.

Для проверки корректности работы RC4 было написано два тестовых примера: зашифрование сообщения и расшифрование его обратно и измерение скорости генерации ПСП. В обоих случаях RC4 демонстрирует хорошую производительность и может быть использован для защиты данных в различных приложениях.

**1. В чем состоит особенность потоковых шифров?**

Особенностью потоковых шифров является то, что они шифруют информацию непрерывным потоком символов, в отличие от блочных шифров, которые шифруют информацию блоками фиксированного размера. Это позволяет потоковым шифрам работать с произвольным объемом данных и обеспечивать высокую скорость шифрования и расшифрования.

В потоковых шифрах используется генератор псевдослучайной последовательности, который генерирует ключевой поток символов на основе начального секретного ключа и/или некоторого случайного вектора инициализации. Этот ключевой поток символов затем используется для шифрования и расшифрования информации.

Еще одной особенностью потоковых шифров является их возможность работать в режиме потоковой передачи данных, когда шифрование и расшифрование данных происходит непрерывно в режиме реального времени. Это делает потоковые шифры подходящими для защиты информации, передаваемой по сети, в том числе для защиты потокового видео и аудио.

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

Синхронные и асинхронные потоковые шифры имеют свои преимущества и недостатки, которые зависят от конкретных задач и требований к уровню безопасности.

**Преимущества синхронных потоковых шифров:**

* Высокая скорость шифрования и расшифрования данных
* Гибкость и возможность работы в режиме потоковой передачи данных
* Легкость реализации и малый размер ключа
* Отсутствие необходимости в передаче большого объема секретных данных, что делает их подходящими для защиты потокового трафика в сети

**Недостатки синхронных потоковых шифров:**

* Небольшая стойкость к атакам, основанным на анализе известных шифротекстов
* Возможность потери синхронизации при передаче данных с ошибками

**Преимущества асинхронных потоковых шифров:**

* Высокая стойкость к различным атакам, включая атаки на основе известных шифротекстов
* Возможность работы в неравномерной сетевой среде
* Возможность использования открытых ключей, что облегчает процесс обмена ключами

**Недостатки асинхронных потоковых шифров:**

* Ниже скорость шифрования и расшифрования данных по сравнению с синхронными шифрами
* Больший размер ключа, что может затруднять передачу ключей в некоторых случаях
* Сложность реализации

В целом, выбор между синхронными и асинхронными потоковыми шифрами зависит от требований к уровню безопасности, скорости передачи данных и других факторов, таких как доступность ключей и среда передачи данных.

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

Генератор псевдослучайных чисел (ГПСЧ) для использования в криптографии должен обладать рядом свойств, которые обеспечивают его стойкость и непредсказуемость. Некоторые из таких свойств могут включать в себя:

* Статистическую непредсказуемость: ГПСЧ должен быть способен генерировать числа, которые нельзя предсказать, даже если злоумышленник знает все предыдущие числа, сгенерированные ГПСЧ.
* Универсальность: ГПСЧ должен генерировать числа, равномерно распределенные в определенном диапазоне значений. Это важно для того, чтобы исключить возможность угадывания следующего числа.
* Равномерность: ГПСЧ должен генерировать числа, равномерно распределенные по всему диапазону значений.
* Длинная периодичность: ГПСЧ должен иметь длинный период генерации чисел, чтобы повторение чисел было маловероятным.
* Стойкость к атакам: ГПСЧ должен быть устойчив к различным атакам, включая атаки на основе известных значений генерируемых чисел.
* Легкость и быстрота генерации: ГПСЧ должен генерировать числа быстро и с минимальными вычислительными затратами.
* Стойкость к криптоанализу: ГПСЧ должен быть устойчив к атакам, которые могут быть направлены на анализ его внутреннего алгоритма или состояния.

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

Характеристики LCG включают следующее:

* Простота реализации и высокая скорость работы. LCG можно реализовать с помощью небольшого количества арифметических операций, что делает его быстрым и легким в использовании.
* Предсказуемость. Поскольку LCG использует рекуррентную формулу для генерации последовательности чисел, эта последовательность может быть предсказана и повторена, если злоумышленник получит достаточно информации о генерируемых числах.
* Низкий уровень стойкости. LCG не является достаточно безопасным для использования в криптографии, так как он не генерирует случайные числа, а псевдослучайные числа. Это означает, что последовательность, генерируемая LCG, может быть повторена с помощью анализа предыдущих значений.
* Ограниченный период. Последовательность чисел, генерируемых LCG, имеет ограниченный период, который определяется начальными значениями и коэффициентами генератора. Если период LCG достаточно мал, это может привести к проблемам с повторяемостью генерируемой последовательности.

Одной из областей применения LCG является моделирование случайных событий в различных областях, таких как физика, экономика и технические науки. LCG также может быть использован в системах, которые не требуют высокой стойкости криптографической защиты, таких как генерация паролей или случайных чисел для игр. Однако, для криптографических целей, LCG не рекомендуется использовать, так как он не обладает достаточной стойкостью.

**5. Значения x0, x1, x2, x3, полученные с помощью линейного конгруэнтного генератора, равны соответственно: 1, 12, 3, 6. Найти параметры а, с и n генератора ПСЧ, удовлетворяющие выражению (6.1).**

Задача была решена с помощью программы на языке Python. Итого получилось:

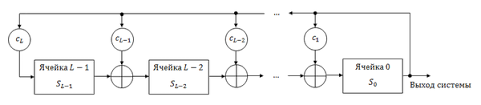
12

3

6

n,a,c (48,21,39)

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



**7. Синтезировать структурную схему генератора ПСП на основе регистров сдвига с линейной обратной связью, формально обозначаемого следующим образом: 3210. Составить таблицу состояний генератора и определить период ПСП.**

Изображение выглядит как диаграмма

Автоматически созданное описание

|  |  |  |  |
| --- | --- | --- | --- |
| ЦОС | 1 | 2 | 3 |
| 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |

Периодичность равна 4.

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

Для решения задачи необходимо определить последовательность значений ПСП, соответствующих начальным состояниям ячеек и заданному формальному описанию.

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

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

где x\_n - текущее значение ПСП, x\_n-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:

x\_0 = 1

x\_1 = 5

x\_2 = 7

x\_3 = 6

x\_4 = 0

Для определения первых 12 бит ПСП, необходимо преобразовать значения x\_0 - x\_4 в двоичную систему счисления и объединить полученные биты в одну последовательность. Полученная последовательность будет первыми 12 битами ПСП, соответствующей заданным начальным состояниям и формальному описанию.

1 = 0001

5 = 0101

7 = 0111

6 = 0110

0 = 0000

Объединение первых 12 бит дает последовательность 0001 0101 0111, что является ответом на задачу.

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

Алгоритм состоит из следующих шагов:

* Генерация ключевой пары RSA (открытый и закрытый ключ).
* Генерация случайного начального числа, которое будет использоваться как первый элемент последовательности.
* Применение RSA-шифрования для каждого следующего элемента последовательности:
* Берется предыдущий элемент последовательности и шифруется открытым ключом RSA.
* Полученное зашифрованное значение используется в качестве следующего элемента последовательности.
* Последовательность псевдослучайных чисел генерируется путем повторения шага 3 до достижения требуемой длины последовательности.

Криптостойкость генератора ПСП на основе RSA основана на криптостойкости криптосистемы RSA. В основе криптосистемы RSA лежит проблема факторизации больших целых чисел, которая считается вычислительно трудной для достаточно больших значений. Следовательно, без знания закрытого ключа, невозможно вычислить следующий элемент последовательности на основе предыдущего элемента и открытого ключа RSA.

Однако, следует отметить, что генератор ПСП на основе RSA может быть уязвим к атакам, если используется недостаточно длинный ключ RSA, что может привести к разгадыванию закрытого ключа и, следовательно, предсказанию последующих элементов последовательности. Поэтому для обеспечения криптостойкости генератора ПСП на основе RSA необходимо использовать достаточно длинный ключ RSA (например, 2048 бит), что усложнит атаку на систему.

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

Для вычисления последовательности псевдослучайных чисел методом BBS (Blum-Blum-Shub) необходимо выполнить следующие шаги:

1. Вычислить произведение p и q: N = p*q = 11*19 = 209.
2. Вычислить значение x\_0 = х^2 mod N: x\_0 = 3^2 mod 209 = 9.
3. Для i от 1 до 11:
   1. Вычислить значение x\_i = x\_(i-1)^2 mod N.
   2. Получить очередное псевдослучайное число z\_i = x\_i mod 2.

Таким образом, для данного примера получаем следующие значения:

* x\_1 = x\_0^2 mod N = 9^2 mod 209 = 81 mod 209 = 81
* x\_2 = x\_1^2 mod N = 81^2 mod 209 = 165 mod 209 = 165
* x\_3 = x\_2^2 mod N = 165^2 mod 209 = 86 mod 209 = 86
* x\_4 = x\_3^2 mod N = 86^2 mod 209 = 68 mod 209 = 68
* x\_5 = x\_4^2 mod N = 68^2 mod 209 = 104 mod 209 = 104
* x\_6 = x\_5^2 mod N = 104^2 mod 209 = 20 mod 209 = 20
* x\_7 = x\_6^2 mod N = 20^2 mod 209 = 109 mod 209 = 109
* x\_8 = x\_7^2 mod N = 109^2 mod 209 = 42 mod 209 = 42
* x\_9 = x\_8^2 mod N = 42^2 mod 209 = 141 mod 209 = 141
* x\_10 = x\_9^2 mod N = 141^2 mod 209 = 127 mod 209 = 127
* x\_11 = x\_10^2 mod N = 127^2 mod 209 = 45 mod 209 = 45

Таким образом, получаем значения: x1 = 81, x5 = 104, x9 = 141, x11 = 45.

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

Базовый алгоритм, реализованный в шифре RC4, основывается на генерации ключевого потока, который затем используется для шифрования или дешифрования данных. Генерация ключевого потока осуществляется на основе ключа, переданного в шифр, и использует алгоритм псевдослучайной генерации, который состоит из следующих шагов:

* Инициализация состояния шифра: создание вектора состояния длиной 256 байт, заполненного значениями от 0 до 255 в порядке возрастания.
* Инициализация вектора состояния на основе ключа: для этого происходит перестановка элементов вектора состояния с использованием ключа.
* Генерация ключевого потока: на основе измененного вектора состояния генерируется последовательность псевдослучайных байтов, которая используется для шифрования или дешифрования данных.
* Обновление вектора состояния: на основе использованных элементов вектора состояния, генерируется новый вектор состояния.

Ключевой поток, сгенерированный в результате выполнения этих шагов, используется для шифрования или дешифрования данных методом XOR (исключающее ИЛИ) с исходными данными.

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

Для формирования истинных случайных последовательностей, основанных на "естественных случайностях", используются различные методы, основанные на природных явлениях. Один из таких методов - это использование шума, происходящего в природе, например, шума радиоволн, шума от дождя или шума от термальных колебаний в полупроводниках. Такой шум считается истинно случайным, поскольку его источник не поддается предсказанию.

Другой метод — это использование ядерного распада в качестве источника случайности. Ядерный распад происходит с определенной вероятностью, которая, хотя и мала, но непредсказуема и может быть использована для формирования случайных последовательностей.

Третий метод — это использование случайного движения молекул, которое может быть измерено с помощью физических сенсоров. Этот метод используется в устройствах, называемых генераторами шума, которые используются в качестве источника случайности.

Четвертый метод — это использование алгоритмов на основе квантовой физики, которые могут генерировать случайные числа, используя квантовые явления, такие как фотонный шум или измерение спина электрона.

Важно отметить, что в отличие от псевдослучайных чисел, которые могут быть предсказаны, если известен алгоритм и начальное значение, истинно случайные последовательности, сгенерированные на основе естественных случайностей, не могут быть предсказаны, что делает их полезными для шифрования и других приложений, где требуется высокая степень случайности. Однако, они могут быть дорогостоящими и сложными в производстве, что может быть непрактично для широкого использования.

**Приложение А**

class RC4:

    def \_\_init\_\_(self, key):

        self.S = list(range(256))

        j = 0

        for i in range(256):

            j = (j + self.S[i] + key[i % len(key)]) % 256

            self.S[i], self.S[j] = self.S[j], self.S[i]

        self.i = self.j = 0

    def generate\_keystream(self, n):

        keystream = []

        for \_ in range(n):

            self.i = (self.i + 1) % 256

            self.j = (self.j + self.S[self.i]) % 256

            self.S[self.i], self.S[self.j] = self.S[self.j], self.S[self.i]

            keystream.append(self.S[(self.S[self.i] + self.S[self.j]) % 256])

        return keystream

key = [123, 125, 41, 84, 203]

plaintext = "This is a test message."

rc4 = RC4(key)

keystream = rc4.generate\_keystream(len(plaintext))

# XOR plaintext and keystream to get the ciphertext

ciphertext = [ord(p) ^ k for p, k in zip(plaintext, keystream)]

# Print the ciphertext in hex format

print("Ciphertext:", "".join(f"{c:02x}" for c in ciphertext))

# Reuse the same RC4 object with the same key

rc4 = RC4(key)

# Decrypt the ciphertext using the same keystream

keystream = rc4.generate\_keystream(len(ciphertext))

plaintext = "".join(chr(c ^ k) for c, k in zip(ciphertext, keystream))

print("Plaintext:", plaintext)

import time

# Generate 1MB keystream and measure the time

start\_time = time.time()

keystream = rc4.generate\_keystream(1024\*1024)

end\_time = time.time()

print("Time taken to generate 1MB keystream:", end\_time - start\_time, "seconds")