|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Информационная безопасность

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

***Разработка программы по хранению данных почтовых сервисов***

Студент ИУ8-33 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

*2020 г.*

Оглавление

[**Введение** 3](#_Toc26805181)

[**Цель работы** 3](#_Toc26805182)

[**Задачи** 3](#_Toc26805183)

[**Ход работы** 4](#_Toc26805184)

[**Результаты**](#_Toc26805187) 18

[**Литература**](#_Toc26805188) 18

# **Введение**

Предметная область курсового проекта включает в себя теоретический материал из области защиты информации, использования шифра «Кузнечик» и генерации паролей.

**Цель работы**

Целью данной работы является разработка программы для хранения зашифрованных данных от почтовых сервисов в файле с возможностью их расшифровки и вывода на экран.

**Задачи**

В данной работе можно выделить следующие поставленные задачи, необходимые для достижения поставленной цели:

1) Выполнить анализ требований и разработать внешние спецификации;

2) Выполнить проектирование программы;

3) Реализовать программу;

4) Протестировать программу;

**Ход работы**

Теоретическая часть

Хранение, защита и своевременное обновление данных от почтовых ресурсов является основной проблемой безопасности данных пользователей. В данный этап времени большинство программ имеют авторизацию по данным почты или любой другой аналог привязки к почтовому ресурсу. Из-за чего организация и хранение данных от почтовых ресурсов в безопасной и защищенной форме является неотъемлемой частью управления и взаимодействия с почтовыми сервисами и сохранения конфиденциальных данных. Также я сам столкнулся с такой проблемой, так как большинство моих сервисов с авторизацией делило один пароль, который был составлен из чисел и только. В следствие чего происходила утечка паролей от моего почтового ящика, а потом и публичное распространение конфиденциальной информации.

Теперь стоит поговорить о платформе, на которой реализуется описанное выше приложение. Реализовано оно на языке С++, поэтому есть возможность запуска на ПК под управлением разных операционных систем.

Как у любого продукта, у С++ есть свои положительные аспекты и отрицательные. К преимуществам можно отнести:

1)Масштабируемость. На языке C++ разрабатывают программы для самых различных платформ и систем.  
2)Возможность работы на низком уровне с памятью, адресами, портами. Что, при неосторожном использовании, может легко превратиться в недостаток.  
3)Возможность создания обобщенных алгоритмов для разных типов данных, их специализация, и вычисления на этапе компиляции, используя шаблоны.

Практическая часть

Перед объяснением процесса реализации курсового проекта, следует разобраться в основных моментах разработки на С++. Для разработки я использовал IDE Visual Studio 2019 Community. Причиной выбора IDE послужило мое хорошее знакомство со средой разработки, а также удобные инструменты по разработке, анализу, сборке и компиляции кода.

Для реализации я решил использовать макет консольного приложения, т. к. имел горький опыт взаимодействия с интерфейсами. С проектом можно взаимодействовать с помощью команд записываемых в консоль. Весь вывод осуществляется в консоль.

Работа программы начинается с ввода путей до текстовых файлов, содержащих ключ и зашифрованные данные. Файлы ключа читаются моментально и сохраняются в специальном классе ByteBlock;

Дальше выводится приветственная фраза, обозначающая готовность программы к вводу команд. Список доступных команд приведен ниже.

Список команд:

open data file / open– открывает файл содержащий данные с почтой и паролем в зашифрованном виде и расшифровывает их, записывая в лист данных, при повторном вызове команды программа предлагает ввести новое положение файла данных. Добавляет полученные объекты в конец текущего листа.

add new mail /add – добавляет новый элемент в массив почтовых данных, если поле введенных данных пусто, вводится ”<blank>”, чтобы избежать конфликта во время шифрования.

remove mail / remove *—*  запрашивает почту для которой будет производиться удаление, далее производится поиск, если элементов с введенной почтой больше одного, предлагается выбрать элемент для удаления. Также есть возможность отменить удаление.

edit / e *—*  запрашивает почту для которой будет производиться изменение, далее производится поиск, если элементов с введенной почтой больше одного, предлагается выбрать элемент для изменения. Также есть возможность отменить удаление.

generate new password / gnp — генерирует новый пароль для элемента из массива почтовых данных, есть возможность выбрать добавлять лм символы для дополнения пароля. По умолчанию определены заглавные, прописные буквы Английского алфавита, а также цифры от 0 до 9.

sort by mail *—*  сортирует лист объектов по возрастанию, сортировка производится по имени почты (поле mail).

sort by name *—*  сортирует лист объектов по возрастанию, сортировка производится по имени объекта (поле name).

save defaults *—*  предлагает сохранить текущее положение файлов данных и ключа, как значения по умолчанию, иначе предлагает ввести иные значения.

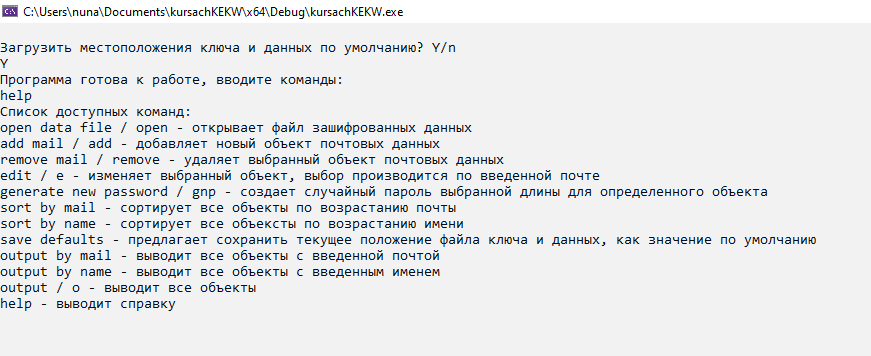
output by mail *—*  запрашивает почту объекта, выводит лист объектов имеющих введенную почту.

output by name *—*  запрашивает имя объекта, выводит лист объектов имеющих введенное имя.

output — выводит содержимое массива почтовых данных на экран при помощи методов класса unit\_class.

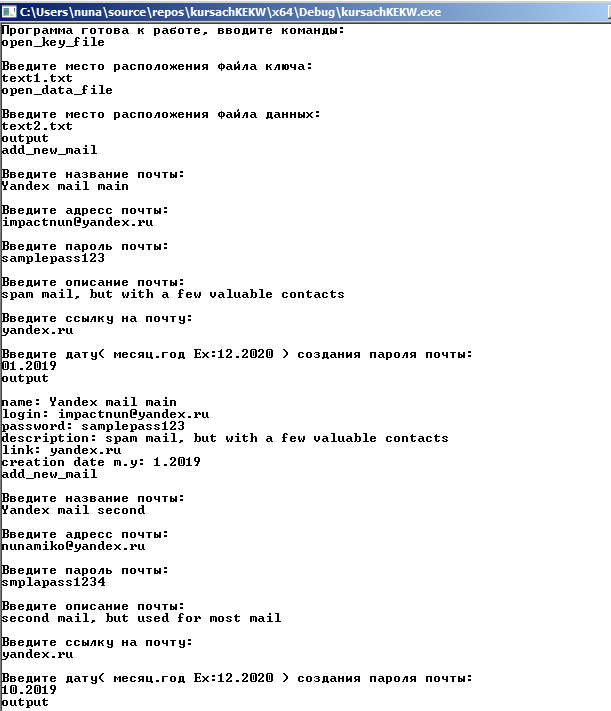
help — выводит список команд с поянением.

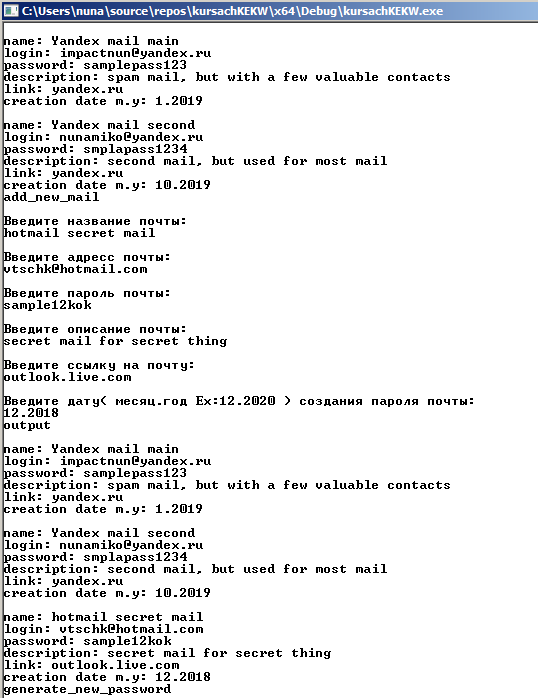
Пример:

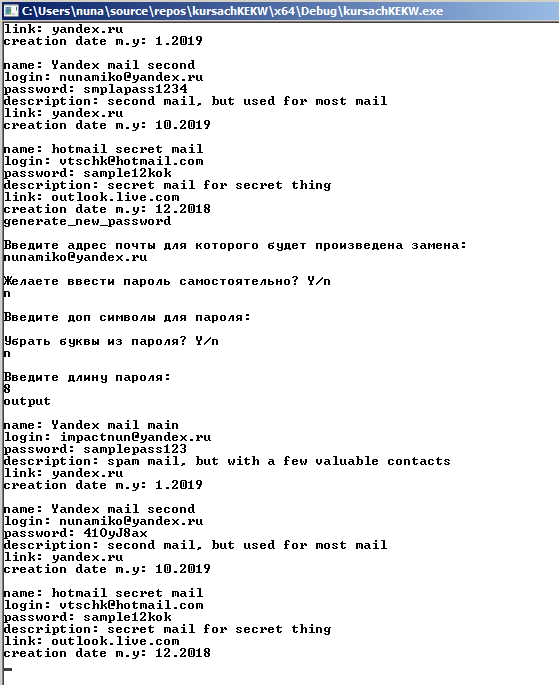


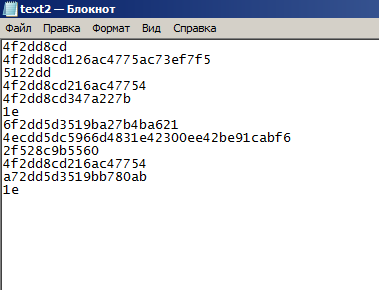
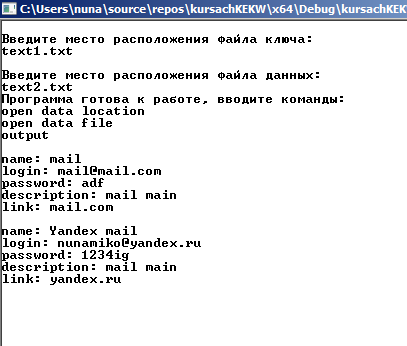
quit / q *—*  завершает ввод команд и переходит к шифровке и последующему записыванию содержимого массива почтовых данных в исходный файл ввода данных.

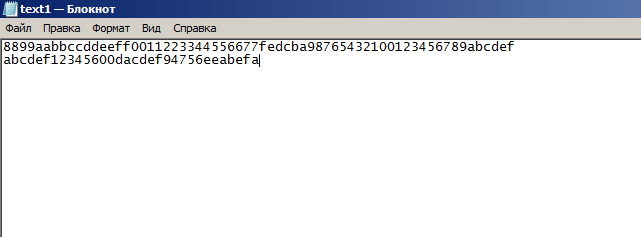
Пример выполнения команд, а также вид текстовых файлов с зашифрованной информацией:

**









Моя программа содержит основной файл выполнения main.cpp

А также дополнительные классы ресурсов и заголовков:

converterofmystupidity.hpp - конвертирует расшифрованные данные перед тем как программа начнет с ними работу и после окончания работы программы методы класса конвертируют строки в подходящий для шифрования вид.

Kyznechik.hpp – блочный шифр Кузнечик(реализует шифрование и расшифровывание кузнечиком для битового блока)

mycrypto.hpp – содержит заголовок класса реализующий хранение данных , а также имеет описание режима обратной связи шифра(превращает блочный шифр в самосинхронизирующийся шифр потока).

unit\_class.hpp – содержит заголовок класса, реализующего хранение данных во время выполнения программы, также имеет функцию генерации пароля.

\*.cpp файлы с аналогичными названиями содержат реализацию данных заголовочных классов.

Шифр Кузнечика:

«Кузнечик»— симметричный алгоритм блочного шифрованияс размером блока 128 бит и длиной ключа 256 бит, использующий для генерации раундовых ключей SP-сеть.

Общие сведения о шифре:

Данный шифр утверждён (наряду с блочным шифром «Магма») в качестве стандарта в ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры» приказом от 19 июня 2015 года № 749-ст.[1] Стандарт вступил в действие с 1 января 2016 года.[2]. Шифр разработан Центром защиты информации и специальной связи ФСБ России с участием ОАО «Информационные технологии и коммуникационные системы» (ОАО «ИнфоТеКС»). Внесён Техническим комитетом по стандартизации ТК 26 «Криптографическая защита информации»[3][4].

Протоколом № 54 от 29 ноября 2018 года, на основе ГОСТ Р 34.12-2015, Межгосударственным советом по метрологии, стандартизации и сертификации был принят межгосударственный стандарт ГОСТ 34.12-2018. Приказом Федерального агентства по техническому регулированию и метрологии от 4 декабря 2018 года № 1061-ст стандарт ГОСТ 34.12-2018 введен в действие в качестве национального стандарта Российской Федерации с 1 июня 2019 года.

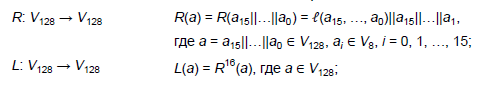
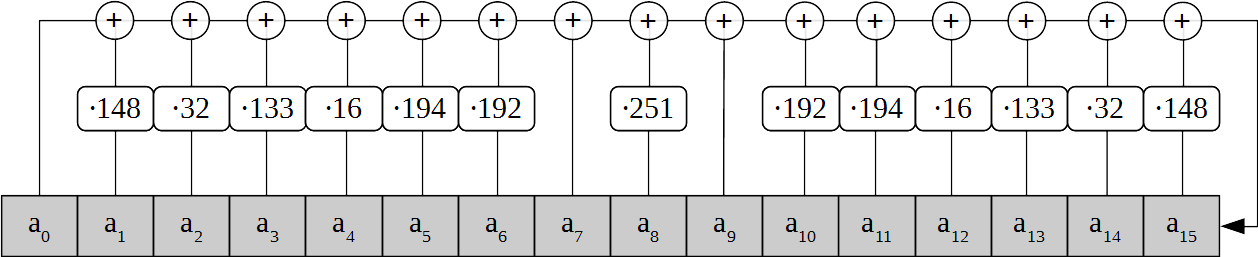
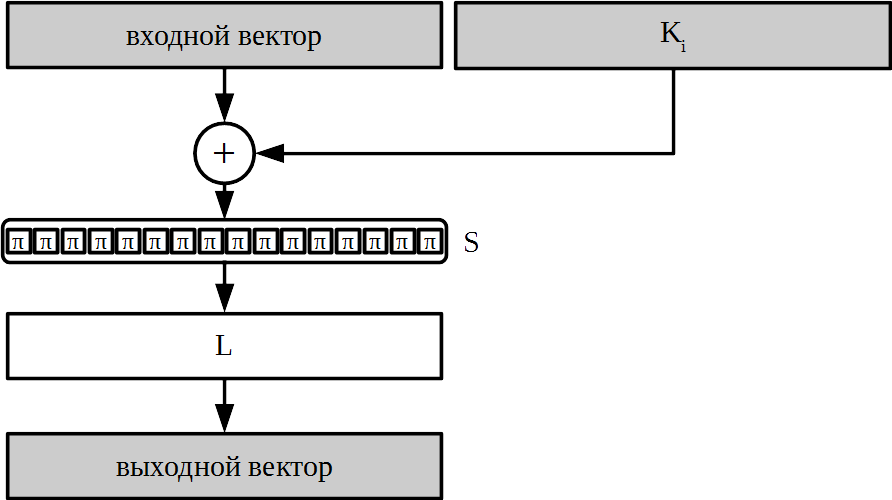
Согласно ГОСТ Р 34.12-2015 и RFC 7801 шифр может именоваться на английском как Kuznyechik, а согласно ГОСТ 34.12-2018 как Kuznechik. К тому же некоторые реализации шифра с открытым исходным кодом используют наименование Grasshopper.

SP-сеть: преобразование, состоящее из нескольких одинаковых раундов, при этом каждый раунд состоит из нелинейного и линейного преобразований, а также операции наложения ключа. В отличие от сети Фейстеля, при использовании SP-сети преобразуется весь входной блок, а не его половина. Такая структура иногда также называется AES-like (похожей на AES), однако, в отличие от последнего у «Кузнечика» есть ряд своих «фишек»:

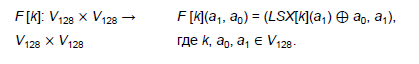
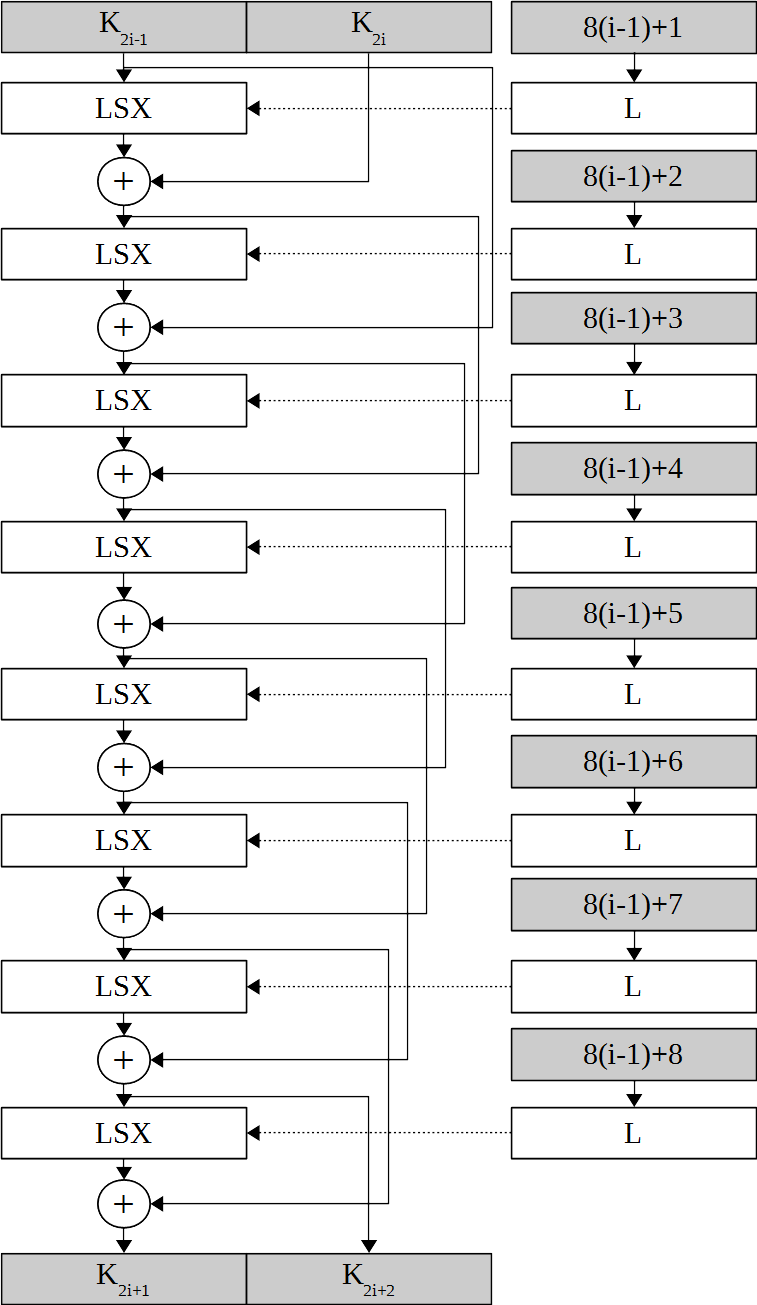
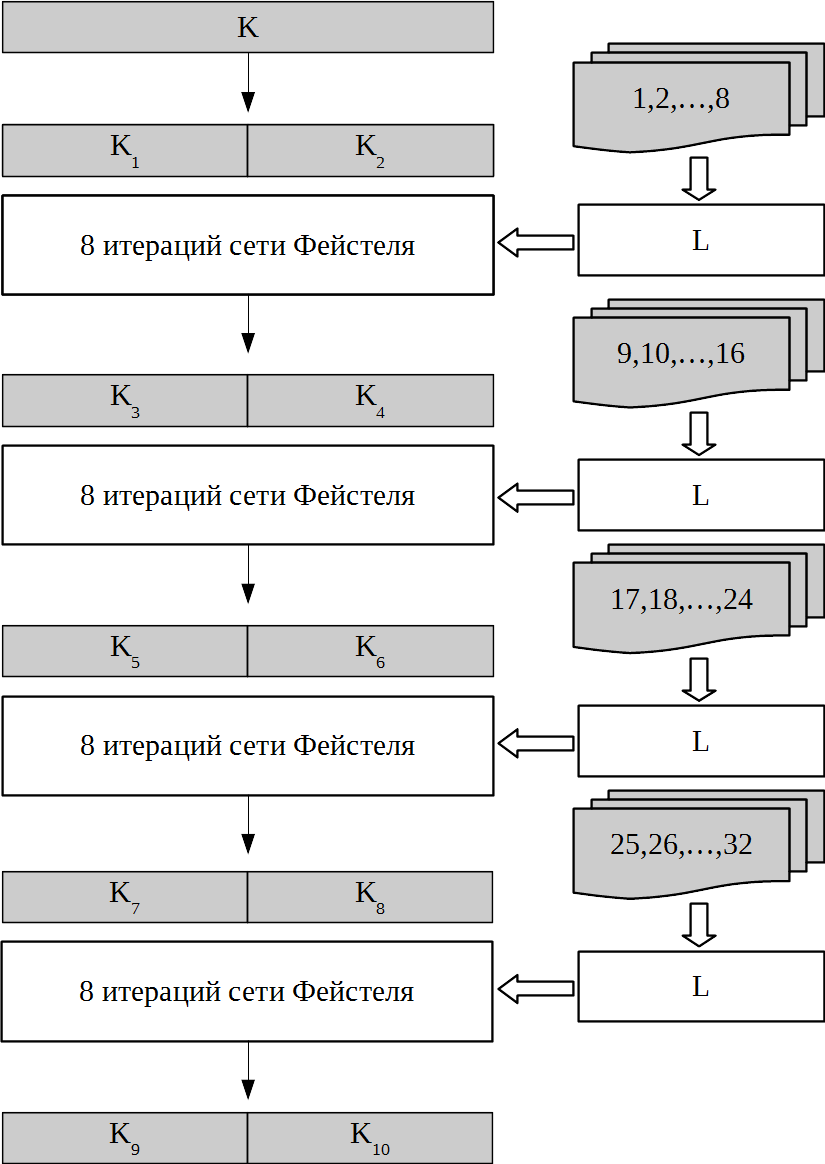
* линейное преобразование может быть реализовано в с помощью регистра сдвига;
* ключевая развертка реализована с помощью сети Фейстеля, в которой в качестве функции используется раундовое преобразование исходного алгоритма.

Длина входного блока «Кузнечика» — 128 бит, ключа — 256 бит.

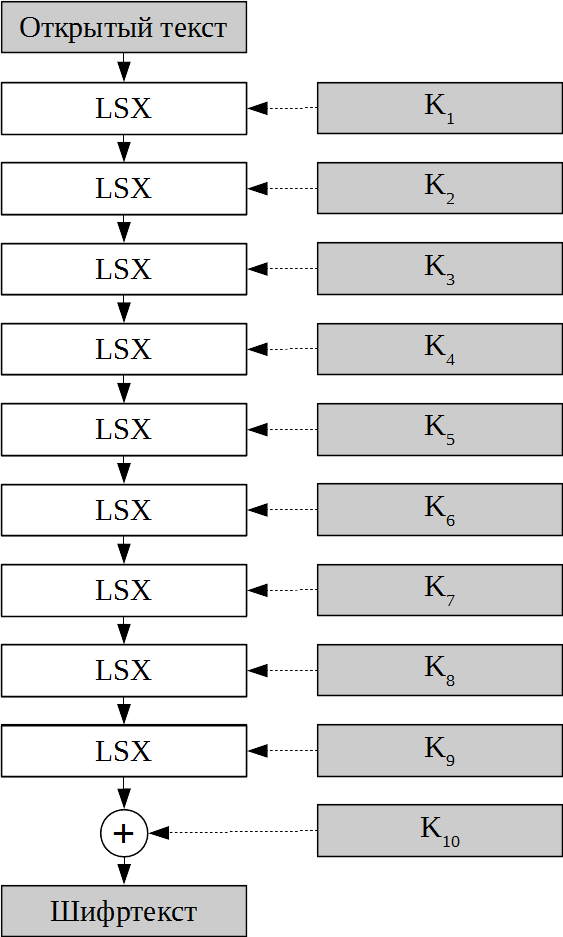
#### Преобразования

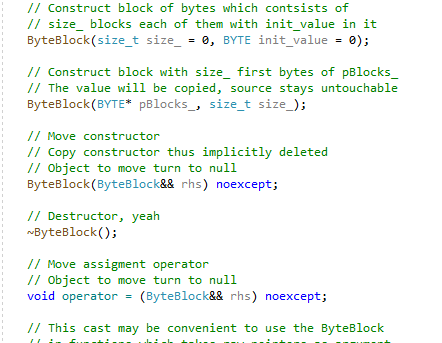
Шифрование основано на последовательном применении нескольких однотипных раундов, каждый из которых содержит три преобразования: сложение с раундовым ключом, преобразование блоком подстановок и линейное преобразование.  
  
128-битный входной вектор очередного раунда складывается побитно с раундовым ключом:  
  
  
  
Нелинейное преобразование представляет собой применение к каждому 8-битному подвектору 128-битного входного вектора фиксированной подстановки:  
  
  
  
В «Кузнечике» используется та же подстановка, что и в хэш-функции «Стрибог».  
  
Линейное преобразование, как я уже сказал, может быть реализовано не только как обычно в блочных шифрах — матрицей, но и с помощью РСЛОС — линейного регистра сдвига с обратной связью, который движется 16 раз.  
  
  
  
Сам регистр реализуется над полем Галуа по модулю неприводимого многочлена степени 8: :  
  
  
  
Раундовое преобразование можно изобразить следующим образом:  
  


#### Выработка раундовых ключей

Рассмотрим теперь процедуру генерации раундовых ключей из мастер-ключа. Первые два получаются разбиением мастер-ключа пополам. Далее для выработки очередной пары раундовых ключей используется 8 итераций сети Фейстеля, где, в свою очередь, в качестве раундовых ключей используется счетчиковая последовательность, прошедшая через линейное преобразование алгоритма:  
  
  
  
  
  
Раунд ключевой развертки можно представить следующим образом:  
  
  
  
А всю процедуру выработки раундовых ключей так:  
  


#### Шифрование и расшифрование

В результате, шифрование одного 128-битного входного блока описывается следующим уравнением:  
  
  
  
А в виде блок-схемы может быть представлено так:  
  
  
  
Расшифрование реализуется обращением базовых преобразований и применением их в обратном порядке:  
  


Рисунок 1.2 - пример задокументированного интерпретируемого файла на С++

**Результаты**

Подводя итоги, следует обобщить, что было сделано в ходе курсового проекта. Мною была разработана и реализована программа на платформе для Visual Studio, языке C++.

**Литература**

[ГОСТ Р 34.12-2015](http://docs.cntd.ru/document/1200121983)«Информационная технология. Криптографическая защита информации. Блочные шифры»

* [ГОСТ 34.12-2018](http://protect.gost.ru/v.aspx?control=7&id=232146)«Информационная технология. Криптографическая защита информации. Блочные шифры»

С. В. Дорохин1,2 , С. С. Качков1,3 , A. A. Сидоренко1,3 1Московский физико-технический институт (государственный университет) 2АО «ПКК Миландр» 3АО «Интел» Реализация блочного шифра «Кузнечик» с иcпользованием векторных инструкций

Википедия [https://ru.wikipedia.org/wiki/%D0%9A%D1%83%D0%B7%D0%BD%D0%B5%D1%87%D0%B8%D0%BA\_(%D1%88%D0%B8%D1%84%D1%80)](https://ru.wikipedia.org/wiki/Кузнечик_(шифр))

Хабр <https://habr.com/ru/post/459004/>

<https://habr.com/ru/post/266359/>