МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
**«Вятский государственный университет»**
**(ФГБОУ ВО «ВятГУ»)**
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Допущено к защите
Руководитель работы
_____/ Караваева О.В./
(подпись) (Ф.И.О)
«__»_____2021г.

# РАЗРАБОТКА ПРОГРАММЫ МОДЕЛИРОВАНИЯ ПЕРЕДАЧИ СООБЩЕНИЙ С СИММЕТРИЧНЫМ ШИФРОВАНИЕМ ДАННЫХ

Пояснительная записка курсовой работы по дисциплине
«Комплекс знаний бакалавра»
ТПЖА.09.03.01.014 ПЗ

Разработал студент группы ИВТ-41 _____/Седов М.Д./

Руководитель
Преподаватель кафедры ЭВМ _____/ Караваева О.В./

Проект защищен с оценкой «_____» _____
(оценка) (дата)

Члены комиссии _____/ _____/
(подпись) (Ф.И.О)

_____ /_____/

Киров 2021

# Реферат

Седов М.Д. Разработка программы моделирования передачи сообщений с симметричным шифрованием данных: ТПЖА.090301014 ПЗ: Курс. работа / ВятГУ, каф. ЭВМ; рук. Караваева О.В. - Киров, 2021. – ПЗ 62 с, 14 рис., 2 табл., 9 источников.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ, СИММЕТРИЧНОЕ ШИФРОВАНИЕ, ДЕШИФРОВАНИЕ, БЛОЧНЫЙ АЛГОРИТМ, СЕТЬ ФЕЙСТЕЛЯ, ЗАКРЫТЫЙ КЛЮЧ, ЗАЩИТА ИНФОРМАЦИИ, QT, CMAKE, C++

Цель курсового проекта — моделирование передачи сообщений с симметричным шифрованием данных при выполнении лабораторных работ по дисциплине «Защита информации».

Программное обеспечение, разработанное в рамках данного курсового проекта — лабораторная установка «Модель передачи сообщений».

В ходе выполнения курсового проекта был выполнен анализ предметной области, проектирование и разработка программного обеспечения.

## Содержание

## Введение

В настоящее время в области образования все больше производится автоматизация контроля знаний учащихся и освоения нового материала.

Одним из способов автоматизации являются специальные программные модели, эмулирующие работу какой-либо системы. С их помощью студенты имеют возможность достаточно подробно изучить ее работу, принципы и особенности. В совокупности с теоретическим материалом это позволяет увеличить степень освоения новых знаний по данной дисциплине и повысить качество обучения в целом

Такие программные модели достаточно широко используются при выполнении лабораторных работ по дисциплине «Защита информации». По данной дисциплине уже есть реализованные установки, но ни одна из них не позволяла осуществить механизм передачи сообщений. Поэтому было принято решение выполнить разработать программу, которая бы моделировала передачу сообщений симметричным шифрованием данных.

# 1. **Анализ предметной области**

На данном этапе работы необходимо рассмотреть функции, необходимые для шифрования, дешифрования данных, рассмотреть генерацию закрытого ключа и алгоритмы передачи сообщений, провести анализ предметной области, найти аналоги, обосновать актуальность разработки программной модели и сформировать требования к программному обеспечению в виде технического задания.

## 1.1 Симметричное шифрование

Симметричное шифрование - это способ шифрования данных, при котором один и тот же ключ используется и для кодирования, и для восстановления информации. До 1970-х годов, когда появились первые асимметричные шифры, оно было единственным криптографическим методом. В целом симметричным считается любой шифр, использующий один и тот же секретный ключ для шифрования и дешифрования.

Например, если алгоритм предполагает замену букв числами, то и у отправителя сообщения, и у его получателя должна быть одна и та же таблица соответствия букв и чисел: первый с ее помощью шифрует сообщения, а второй -дешифровывает.

Однако такие простейшие шифры легко взломать - например, зная частотность разных букв в языке, можно соотносить самые часто встречающиеся буквы с самыми многочисленными числами или символами в коде, пока не удастся получить осмысленные слова. С использованием компьютерных технологий такая задача стала занимать настолько мало времени, что использование подобных алгоритмов утратило всякий смысл.

В настоящее время все симметричные шифры можно поделить на два типа:

1) Блочные шифры
2) Поточные шифры

Блочные алгоритмы шифруют данные блоками фиксированной длины (64, 128 или другое количество бит в зависимости от алгоритма). Если все сообщение или его финальная часть меньше размера блока, система дополняет его предусмотренными алгоритмом символами, которые так и называются дополнением.

Потоковое шифрование данных предполагает обработку каждого бита информации с использованием гаммирования, то есть изменения этого бита с помощью соответствующего ему бита псевдослучайной секретной последовательности чисел, которая формируется на основе ключа и имеет ту же длину, что и шифруемое сообщение. Как правило, биты исходных данных сравниваются с битами секретной последовательности с помощью логической операции XOR (исключающее ИЛИ, на выходе дающее 0, если значения битов совпадают, и 1, если они различаются).

### 1.1.1 Генерация ключей

Симметричные алгоритмы требуют меньше ресурсов и демонстрируют большую скорость шифрования, чем асимметричные алгоритмы. Большинство симметричных шифров предположительно устойчиво к атакам с помощью квантовых компьютеров, которые в теории представляют угрозу для асимметричных алгоритмов.

Слабое место симметричного шифрования — обмен ключом. Поскольку для работы алгоритма ключ должен быть и у отправителя, и у получателя сообщения, его необходимо передать; однако при передаче по незащищенным каналам его могут перехватить и использовать посторонние. На практике во многих системах эта проблема решается шифрованием ключа с помощью асимметричного алгоритма.

### 1.1.2 Обзор аналогов

Во время обзора аналогов была выявлена программа Cryptool, содержащая множество алгоритмов шифрования симметричного и ассиметричного вида. Данная программа не рассматривалась как установка для сдачи лабораторной работы студентами, поскольку:

1) Интерфейс программы написан на английском языке, что усложняет восприятие;
2) Имеется много алгоритмов, которые нужны лишь для изучения. То есть такие алгоритмы, которые не применяются на практике в настоящее время;
3) Каждый из алгоритмов построен не на проверки знаний об алгоритме, а на обучении данному алгоритму.

К примеру, в одной из лабораторных работ дисциплины «Защита информации» рассматривался алгоритм шифрования RSA. В данной работе студентам было необходимо разработать программы, которая позволяли бы сгенерировать открытый и закрытый ключи, с их помощью зашифровать и дешифровать свою фамилию и инициалы, а также для вычисления цифровой подписи. Также в лабораторной работе рассматривался метод шифрования ГОСТ 28147-89, но он представляет из себя устаревший алгоритм симметричного блочного шифрования.

При анализе имеющихся программ в интернете не была выявлена программа, позволяющая сохранить понимание работы каждого из алгоритмов, а также сохранить простоту работу.

### 1.1.3 Используемые в программной модели алгоритмы шифрования

1) AES (Advanced Encryption Standard) - симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит), принятый в качестве стандарта шифрования правительством США по результатам конкурса AES. Этот алгоритм хорошо проанализирован и сейчас широко используется, как это было с его предшественником DES.
2) ГОСТ 28147-89 - блочный шифр с 256-битным ключом и 32 циклами (называемыми раундами) преобразования, оперирующий 64-битными блоками. Основа алгоритма шифра — сеть Фейстеля.

### 1.1.4 Недостатки

В лабораторной работе, связанной с шифрованием данных, были найдены следующие недостатки:

- лабораторная работа состоит из заданий, в каждом из которых студенту предоставляется возможность написать программную реализацию того или иного алгоритма;
- реализация только одного алгоритма симметричного шифрования (ГОСТ 28147-89) и одного ассиметричного шифрования (RSA);
- шифрация данных лишь на одном примере;
- отсутствие возможности дешифрации имея уже зашифрованное сообщение;
- отсутствие модели передачи данных и создания двух и более собеседников

## 1.2 Актуальность разработки

Лабораторная работа по данной теме (передача сообщений) имеет важную роль в закреплении лекционного материала и предоставляет студентам возможность изучить более подробно механизмы генерации открытого и закрытого ключей, алгоритмы шифрации данных, а также механизмы создания атак на сообщения и способы защиты от них.

Поэтому было принято решение разработать программную модель, в которой будут исправленные вышеописанные ошибки и недостатки, а также будет упрощена возможность изучения материала.

## 1.3 Техническое задание

### 1.3.1 Наименование программы

Наименование программы - «Модель передачи сообщений симметричным шифрованием данных».

### 1.3.2 Краткая характеристика области применения

Программа предназначена для закрепления студентами лекционного материала по дисциплине «Защита информации», а именно: шифрование данных.

### 1.3.3 Назначение разработки

Функциональным назначением программы является предоставление студентам возможности изучить процесс генерации ключей и также шифрации и дешифрации данных, во время выполнения лабораторных работ.

Программа должна эксплуатироваться на ПК студентов, преподавателей и на ПК, установленных в учебных аудиториях Вятского государственного университета. Особые требования к конечному пользователю не предъявляются.

### 1.3.4 Требования к программе

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

1) функции генерации задания;
2) функции сохранения текущего прогресса выполнения задания в файл;
3) функции загрузки задания с сохраненным прогрессом выполнения из файла;
4) функции генерации закрытого ключа;
5) функции шифрации данных;
6) функции дешифрации данных;
7) функции, предоставляющие модель передачи сообщений.
8) функции для создания ситуаций атак на сообщения
9) функции для предотвращения атак

Надежное (устойчивое) выполнение программы должно быть обеспечено выполнением пользователем совокупности организационно- технических мероприятий, перечень которых приведен ниже:

1) организацией бесперебойного питания технических средств;

2) использованием лицензионного программного обеспечения.

Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных прав.

В состав технических средств должен входить IBM — совместимый персональный компьютер, включающий в себя:

1) x86 — совместимый процессор с тактовой частотой не меньше 1.0 ГГц;

2) дисплей с разрешением не меньше, чем 1024x768;

3) не менее 500 мегабайта оперативной памяти;

4) не менее 100 мегабайт свободного дискового пространства;

5) клавиатура, мышь.

Системные программные средства, используемые программой, должны быть представлены следующими операционными системами:

- 64 — разрядная ОС Window 7/8/8.1/10;
- 64 — разрядная ОС Ubuntu 18.04 и выше/ либо другой дистрибутив Linux

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса и предоставлять возможность выполнять наиболее частые операции с помощью сочетаний клавиш на клавиатуре.

### 1.3.5 Требования к программной документации

Состав программной документации должен включать в себя:

1) техническое задание;
2) программу и методики испытаний;
3) руководство пользователя;
4) техническую документацию;
5) исходный код.

### 1.3.6 Стадии и этапы разработки

Разработка должна быть проведена в три стадии:

1) разработка технического задания;
2) рабочее проектирование;
3) внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласование и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1) разработка программы;
2) разработка программной документации;
3) испытания программы.

На стадии внедрения должен быть выполнен этап подготовки и передачи программы заказчику.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1) постановка задачи;
2) определение и уточнение требований к техническим средствам;
3) определение требований к программе;
4) определение стадий, этапов и сроков разработки программы и документации на

нее;

5) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77 с требованиями п. Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные виды работ:

1) разработка, согласование, утверждение программы и методики испытаний;

2) проведение приемо — сдаточных испытаний;

3) корректировка программы и программной документации по результатам испытаний.

На этапе подготовки и передачи программы должна быть выполнена подготовка и передача программы и программной документации в эксплуатацию заказчику.

## 2. Разработка структуры приложения

На данном этапе работы необходимо в соответствии с требованиями, поставленными в техническом задании, разработать алгоритмы функционирования и модульную структуру приложения.

### 2.1 Разработка алгоритмов функционирования

На основе обозначенных в техническом задании требований к программному обеспечению можно построить алгоритмы, описывающие структуру и поведение разработки.

Во время обработки заявки пользователь выполняет некоторую последовательность действий из шагов определенного типа. Данные шаги можно

представить как операции, применяемые к состоянию системы. Для проверки действий, выполненных пользователем над состоянием системы, необходимо разработать алгоритмы обработки поступающих заявок.

Разработка модели будет выполняться на основе задачи, рассмотренной в книге «Прикладная криптография» Брюс Шнайер.

Криптографический протокол – протокол, использующий криптографию. Криптографический протокол включает в себя некоторый криптографический алгоритм, но, вообще говоря, предназначение протокола выходит за рамки простой безопасности.

Для демонстрации работы протоколов используются следующие «игроки»:

1) Алиса и Боб. Они участвуют во всех двусторонних протоколах. Как правило, Алиса начинает все протоколы, а Боб отвечает. Если для протокола нужна третья или четвертая сторона, то в игру вступают Кэрол и Дейв
2) Ева. Пассивный злоумышленник, который может читать сообщения, передаваемые между Алисой и Бобом, но не может воздействовать на сообщения.
3) Мэллори. Активный злоумышленник, который может не только читать сообщение, но и подменять их. В рамках протоколов его можно назвать взломщиком протоколов.
4) Трент. Доверительный посредник. Это заинтересованная третья сторона, которой доверено завершение протокола. Посредники помогают реализовать работу проколов взаимодействия недоверяющих друг другу сторон. В реальном мире в качестве посредником обычно выступают юристы.

2.1.1 Алгоритм обработки поступающих заявок

Алгоритм обработки заявки «Обмен ключами с помощью симметричной криптографии»

Этот алгоритм предполагает, что пользователи сети, Алиса и Боб, получают секретный ключ от Трента. Перед начало эти ключи уже должны быть у пользователей. Предполагается, что ключи уже у пользователей и Мэллори не имеет о них никакой информации.

1) Алиса обращается к Тренту и запрашивает сеансовый ключ для связи с Бобом;
2) Трент генерирует случайный сеансовый ключ. Он зашифровывает две копии ключа: одну для Алисы, другую – для Боба. Затем Трент посылает обе копии Алисе;
3) Алиса дешифровывает свою копию сеансового ключа;
4) Алиса посылает Бобу его копию сеансового ключа;
5) Боб дешифровывает свою копию сеансового ключа;
6) Алиса и Боб используют этот сеансовый ключ для безопасного обмена информацией.

Схема обмена ключами с помощью симметричной криптографии представлена на рисунке 1.


Рисунок 1 — Схема обмена ключами с помощью симметричной криптографии

Алгоритм обработки заявки «Человек посередине»

Мэллори не только может подслушивать сообщения Алисы и Боба, но и изменять сообщения, удалять сообщения и создавать новые. Мэллори может выдать себя за Боба, сообщающего что-то Алисе, или за Алису, сообщующую что-то Бобу.

1) Алиса посылает Бобу свой открытый ключ. Мэллори перехватывает его и посылает Бобу свой собственный открытый ключ;
2) Боб посылает Алисе свой открытый ключ, Мэллори перехватывает его и посылает Алисе собственный открытый ключ;
3) Когда Алиса посылает сообщение Бобу, зашифрованное открытый ключом «Боба», Мэллори перехватывает его. Так как сообщение в действительности зашифровано его собственным открытым ключом, он дешифровывает его, снова зашифровывает открытым ключом Боба и посылает Бобу;
4) Когда Боб посылает сообщение Алисе, зашифрованное открытый ключом «Алисы», Мэллори перехватывает его. Так как сообщение в действительности зашифровано его собственным открытым ключом, он дешифровывает его, снова зашифровывает открытым ключом Алисы и посылает Алисе.

   Данное вскрытие будет работать, даже если открытые ключи Алисы и Боба хранятся в базе данных. Мэллори может перехватить запрос Алисы к базе данных и подменить открытый ключ Боба своим собственным. То же самое он может сделать и с открытым ключом Алисы. Такая атака работает, поскольку у Алисы и Боба нет способов проверить, действительно ли они общаются именно друг с другом. Если вмешательство Мэллори не приводит к заметным задержкам, оба человека даже не подумают, что кто-то, сидящий между ними, читает все их секретные сообщения.

Схема атаки «Человек посередине» представлена на рисунке 2.


Рисунок 2 — Схема алгоритма генерации атаки «человек посередине»

Алгоритм обработки заявки «протокол держась за руки»

Протокол «держась за руки» был изобретен Роном Ривестом и Эди Шамиром, предоставляющий неплохую возможность избежать атаки «человек посередине»

1) Алиса посылает Бобу свой открытый ключ
2) Боб посылает Алисе свой открытый ключ
3) Алиса зашифровывает свое сообщение открытым ключом Боба. Половину зашифрованного сообщения она отправляет Бобу
4) Боб зашифровывает свое сообщение открытым ключом Алисы. Половину зашифрованного сообщения он отправляет Алисе
5) Алиса отправляет Бобу вторую половину зашифрованного сообщения
6) Боб складывает две части сообщения Алисы и дешифровывает его с помощью своего закрытого ключа. Боб отправляет Алисе половину своего зашифрованного сообщения
7) Алиса складывает две части сообщения Боба и дешифровывает его с помощью своего закрытого ключа

Схема протокола «держась за руки» представлена на рисунке 3.


Рисунок 3 — Схема протокола «держась за руки»

Алгоритм обработки заявки «Шифрация данных»

1) Взять открытый ключ (e,n)
2) Взять открытый текст m
3) Зашифровать сообщение с использованием открытого ключа:

Схема шифрации данных представлена на рисунке 4.

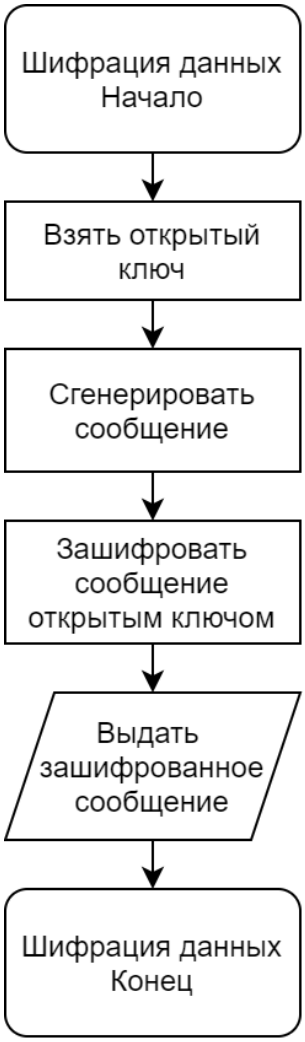



Рисунок 4 — Схема алгоритма шифрации данных

Алгоритм обработки заявки «Дешифрация данных»

1) Принять зашифрованное сообщение с
2) Взять свой закрытый ключ (d,n)
3) Применить закрытый ключ для дешифрования сообщения:

Схема дешифрации данных представлена на рисунке 5.


Рисунок 5 — Схема алгоритма дешифрации данных

Алгоритм обработки заявки «Ввод данных для генерации ключей»

1) Ожидание нажатия. Если нажат пункт меню «Генерация ключей», перейти к следующему пункту, иначе перейти к пункту 8.
2) Ввод основных значений, необходимых для формирования ключа
3) Ожидание нажатия. Если нажата кнопка «Сохранить», перейти к следующему пункту, иначе перейти к пункту 6.
4) Если введены не все данные в полях перейти к пункту 5, иначе перейти к пункту 7.
5) Выдать сообщение об ошибки. Перейти к пункту 3.
6) Ожидание нажатия. Если нажата кнопка «Отмена», перейти к следующему пункту, иначе перейти к пункту 3.
7) Сохранить значения ключей.
8) Завершить алгоритм.

Схема ввода данных для генерации ключей представлена на рисунке 6.

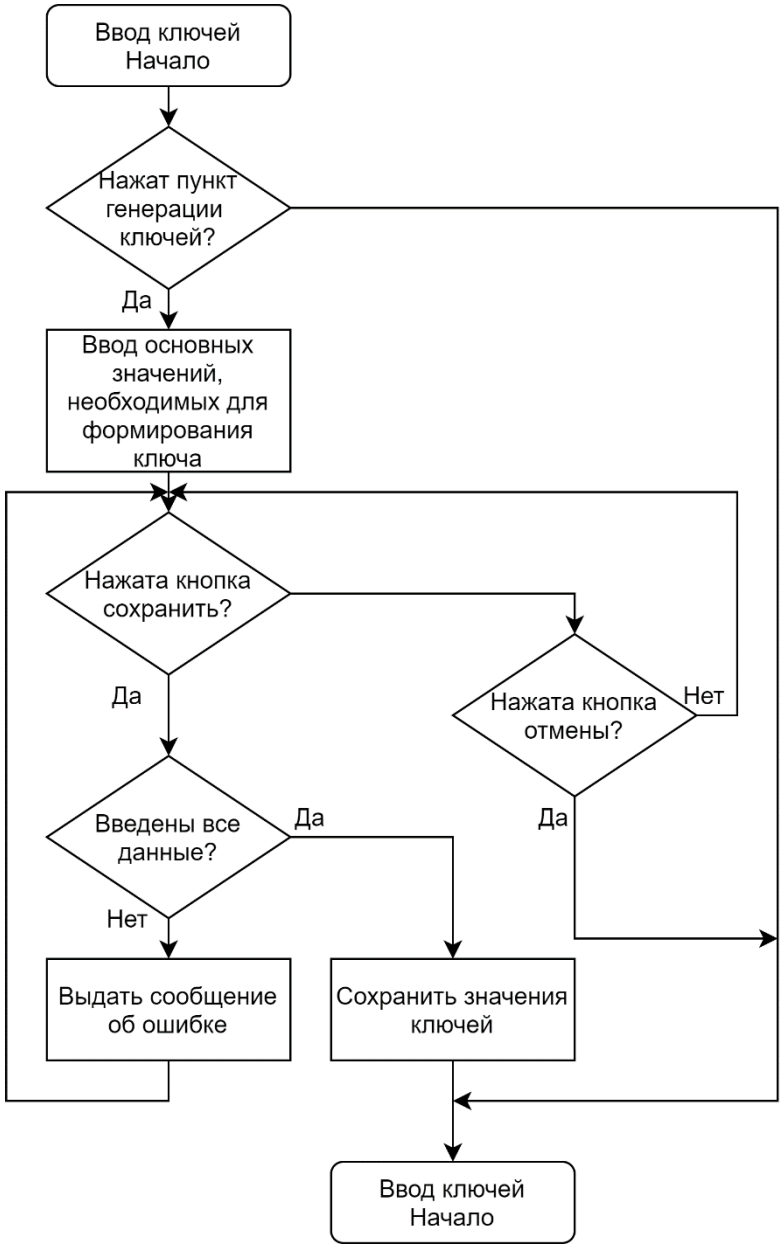



Рисунок 6 — Схема алгоритма ввода данных для генерации ключей

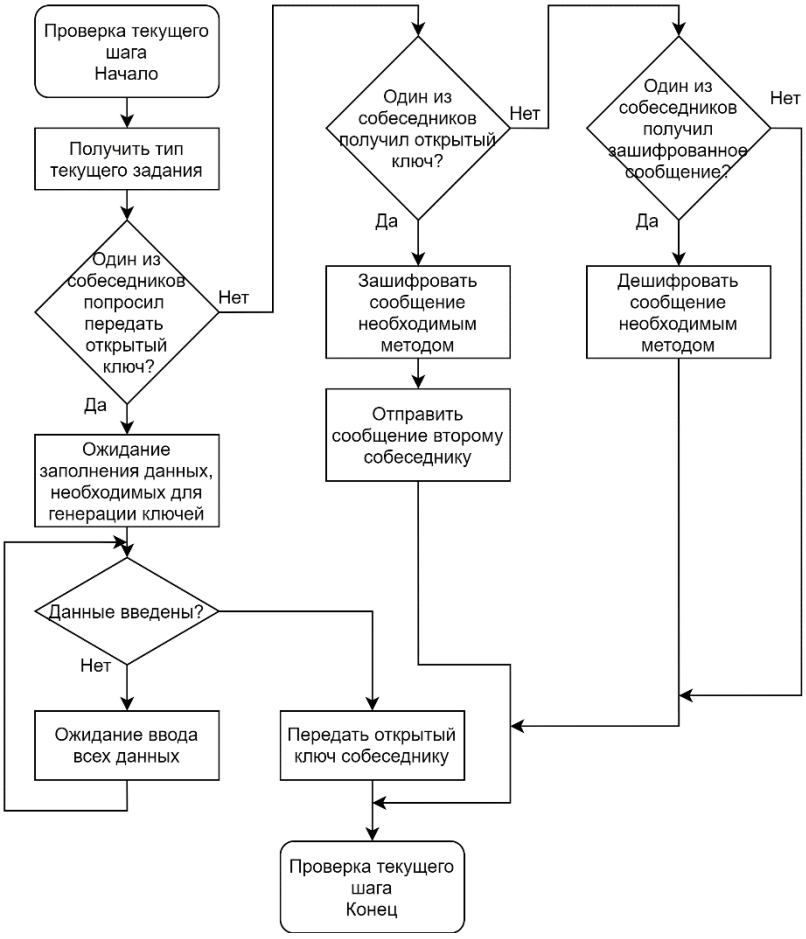



Рисунок 7 — Схема алгоритма выполнения текущего шага лабораторной работы

## 2.2 Разработка модульной структуры приложения

Для обеспечения функционирования системы разработана обобщенная структура программного продукта, представляющая собой набор взаимосвязанных модулей, которые реализуют используемые алгоритмы функционирования. Модульная структура приложения представлена на рисунке 8.

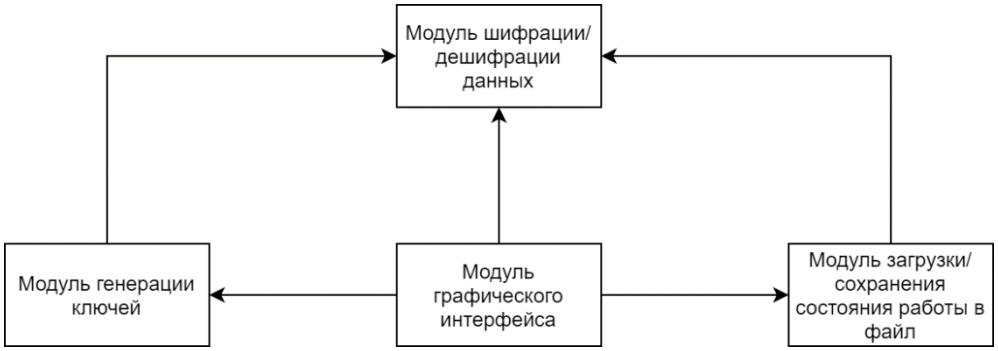



Рисунок 8 — Модульная структура приложения

Каждый из модулей имеет следующие назначения и функционал:

- модуль шифрации/дешифрации данных. Определяет алгоритмы для шифрации данных и их передачи получателю, а также алгоритмы для дешифрации данных в исходное сообщение;
- модуль-генерации ключей. Определяет алгоритмы генерации открытого и закрытого ключей;
- модуль загрузки и сохранения заданий в файл. Предоставляет возможность сохранять текущее состояние выполняемого задания между запусками приложения, а также обеспечивает защиту от непредвиденного изменения структуры файла;
- модуль графического интерфейса. Является связующим звеном между приложением и пользователем; отображает данные о ходе выполнения заданий в текстовом и графическом виде.

# 3 Программная реализация

На данном этапе работы необходимо выбрать инструменты для кодирования приложения (язык программирования, библиотеки), выполнить реализацию разработанных ранее модулей и алгоритмов, а также разработать графический интерфейс пользователя.

## 3.1 Выбор инструментов разработки

Так приложение должно запускаться на различных ОС, а также быть простым в установке и запуске, необходимо выбрать компилируемый ЯП с возможностью создания «родных» для платформы исполняемых файлов. Альтернативным вариантом будет использование интерпретируемого ЯП, который будет иметь возможность запускаться на многих ОС. Помимо прочего, для реализации графического интерфейса необходимо выбрать кроссплатформенную библиотеку для его построения.

Среди доступных ЯП имеются:

- Golang;
- C++;
- Rust;
- C#;
- Python

Среди кроссплатформенных библиотек для построения графического интерфейса имеются:

- GTK (от сокращения GIMP Toolkit) – кроссплатформенная библиотека элементов интерфейса. Написана на C, доступны интерфейсы для других языков программирования, в том числе для C++, C#, Python. Для проектирования графического интерфейса доступно приложение Glade;
- Qt - кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса

и заканчивая классами для работы с сетью, базами данных и XML. Комплектуется визуальной средой разработки графического интерфейса Qt Designer, позволяющей создавать диалоги и формы в режиме WYSIWYG;

- Tkinter (**Tk inter**face) - кросс-платформенная графическая библиотека на основе средств Tk (широко распространённая в мире GNU/Linux и других UNIX-подобных систем, портирована также и на Microsoft Windows). Tkinter входит в стандартный дистрибутив Python. Имеет множество готовых графических компонентов, которые могут понадобиться при создании приложения. Имеет небольшой порог входа в силу простоты создания приложений на языке Python.
- wxWidgets - кроссплатформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений. Основным применением wxWidgets является построение графического интерфейса пользователя, однако библиотека включает большое количество других функций и используется для создания весьма разнообразного ПО. Важная особенность wxWidgets: в отличие от некоторых других библиотек (GTK, Qt и др.), она максимально использует «родные» графические элементы интерфейса операционной системы всюду, где это возможно. Это существенное преимущество для многих пользователей, поскольку они привыкают работать в конкретной среде, и изменения интерфейса программ часто вызывают затруднения в их работе. Написана на C++. Для проектирования графического интерфейса доступно приложение wxGlade;
- AvaloniaUI - кроссплатформенный XAML-фреймворк для построения графических интерфейсов пользователя для платформ .NET Framework, .NET Core и Mono.

При этом нужно учитывать, что не все комбинации языков программирования с библиотеками графического интерфейса возможны, удобны в разработке и достаточно стабильны. Для языков Golang и Rust не имеются стабильные версии перечисленных библиотек, для GTK сборка C++- приложений под Windows

требует нетривиальной настройки окружения вплоть до эмуляции Linux-окружения в Windows, а AvaloniaUI доступна только под C# и не имеет еще стабильной версии. Среди доступных вариантов остаются:

- C++ и wxWidgets;
- C# и GTK;
- C++ и Qt;
- Python и Tkinter.

В ходе ознакомления с библиотекой wxWidgets были выявлены следующие недостатки: недостаточно подробная документация, нетривиальная настройка режима Drag'n'Drop (необходим для перемещения элементов ГПИ, представляющих блоки памяти), малообъемный функционал конструктора форм wxGlade.

У GTK графические элементы выглядят достаточно непривычно в сравнении с родными для Windows приложениями; имеет те же недостатки, что и wxWidgets.

В силу простоты, мощности и популярности был выбран язык C++ и билиотека Qt.

В качестве системы сборки выбрана Cmake — одна из наиболее популярных среди проектов, написанных на C++.

## 3.2 Реализация модулей приложения

### 3.2.1 Модуль шифрации и дешифрации данныз

В данном модуле должны быть реализованы алгоритмы шифрации и дешифрации данных.

Диаграмма классов данного модуля представлена на рисунке 9.

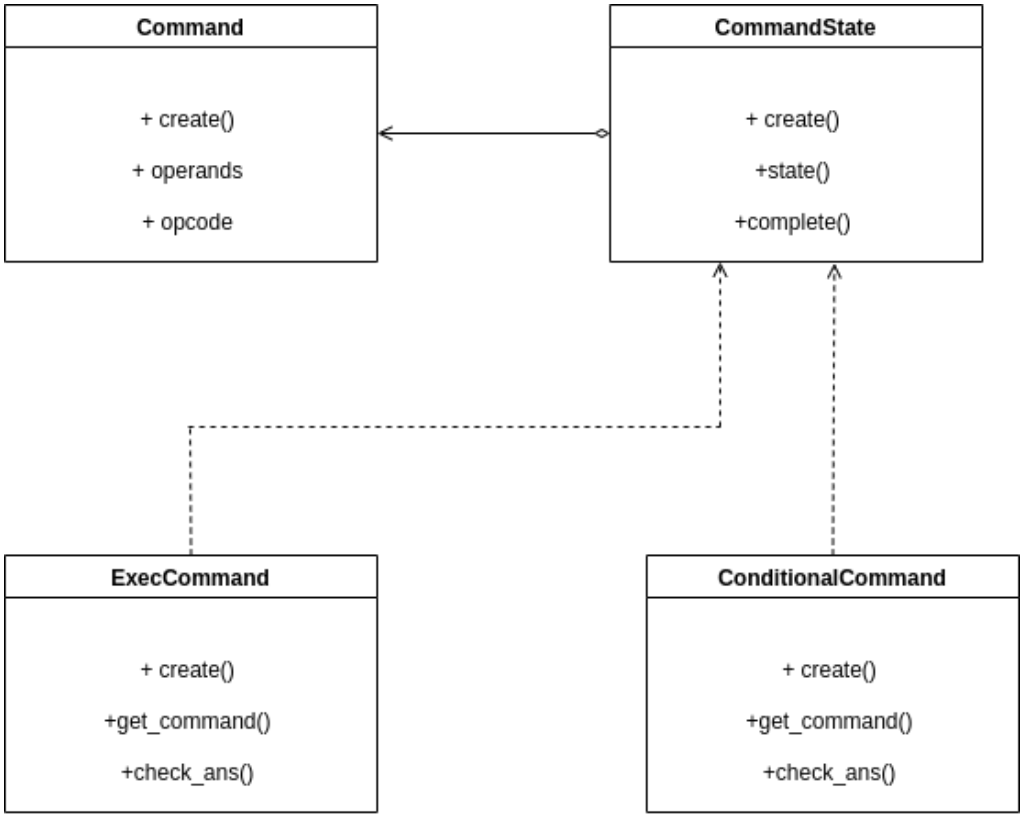



Рисунок 9 — Диаграмма классов модуля обработки команд

3.2.2 Модуль загрузки и сохранения заданий в файл

Данный модуль реализует функционал сохранения текущего прогресса выполнения задания в файл, загрузку задания из файла с защитой от непредвиденных изменений.

Состояние задание определяется такими параметрами как:

- список всех заданий;
- количество уже выполненных заданий;
- количество допущенных ошибок пользователя;
- список введенных команд;

Все перечисленные выше параметры должны быть сохранены в файл.

В качестве формата файла задания был выбран JSON, потому что он имеет синтаксис, достаточно удобный как для человека, так и для машины. JSON является текстовым форматом, а не бинарным, поэтому файлы в формате JSON можно просматривать и редактировать в любом текстовом редакторе. Задания в файле сохранены в виде массива JSON-объектов. Структура объекта задания представлена в таблице 1.

Таблица 1 — Структура обьекта задания

| Поле | Тип | Описание |
|---|---|---|
| type | number | Тип алгоритма шифрации |
| completed | number | Выполнено ли задание |
| fails | number | Количество допущенных ошибок |
| commands | array | Массив команд, которые ввел пользователь |

Структура обьекта, включащего в себя представление EXE программы представлена в таблице 2.

Таблица 2 — Структура сообщения

| Поле | Тип | Описание |
|---|---|---|
| message | array | Массив, включающий в себя передаваемое сообщение |
| public key | number | Открытый ключ |

3.2.3 Модуль генерации ключей

Поскольку основная часть в генерации задания заключается в генерации сообщения и некоторых начальных значений было принято решение написать модуль, выполняющий функции выбора некоторого сообщения для отправки из списка подготовленных, а также реализующий генерацию значений, необходимый для конкретного шифрования данных. Все последующие действия определяются либо самим алгоритмом, либо действиями пользователя

.

3.2.4 Модуль графического интерфейса

В данном модуле были разработаны схемы графического интерфейса.

Основная область окна была поделена четырьмя вкладками, каждая из которых отражает реализацию работы алгоритма шифрации данных. Каждая из вкладок содержит два основных текстовых поля, в которых содержится история передачи сообщений между двумя собеседниками. Перед началом отправки сообщений для любого из алгоритмов происходит этап генерации ключей.

На данном этапе пользователю необходимо проделать шаги, характерные для типа алгоритма, после чего на выходе у него будут открытый и закрытый ключи.

Для упрощения, открытый ключ будет виден на экране, а закрытый будет скрыт и находится будет в настройках.

После генерации ключей происходит отправка сообщения. Пользователю необходимо выполнить операции шифрации текста, который после нажатия на кнопку отправляется другому человеку.

При получении сообщения появляется уведомление, после чего необходимо дешифровать сообщение с целью определения исходного сообщения.

Также в правой части окна имеется панель с историей пересланных сообщений, между которым можно перемещаться для просмотра

Последовательность данных действий повторяется несколько раз. К тому же имеется возможность изменить открытый и закрытый ключи, тем самым закрепляется навык владения каждым из алгоритмов шифрования.

Схема графических интерфейсов представлены на рисунке 10.

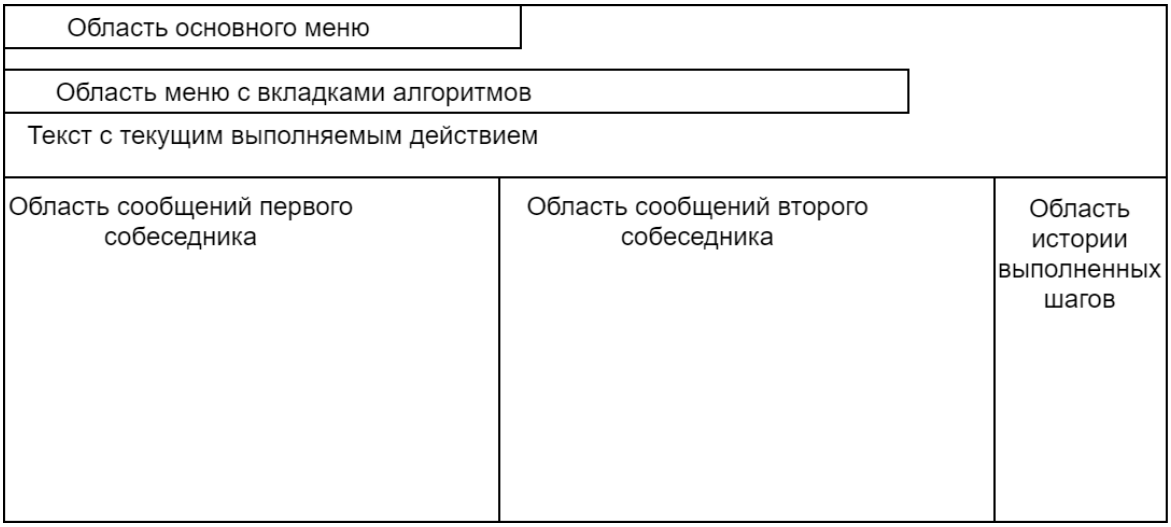



Рисунок 10 — Схема модуля графического интерфейса

.

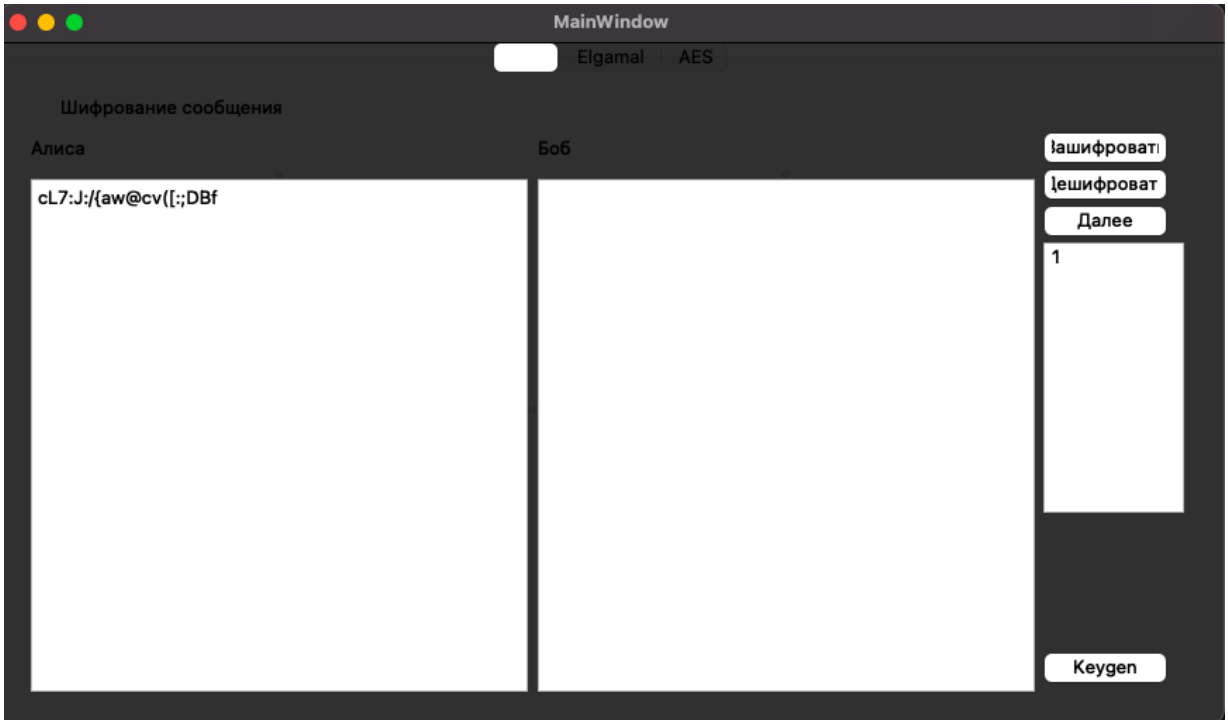



Рисунок 11 – Экранная форма основного окна программы

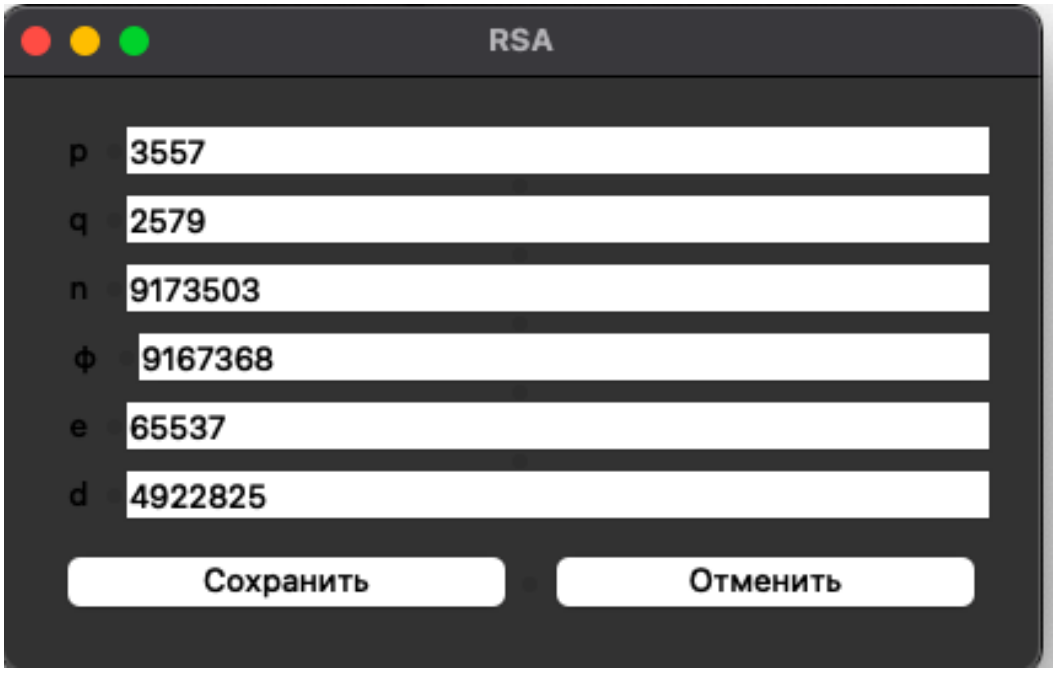



Рисунок 12 – Экранная форма окна с указанием значений, необходимых для генерации ключей

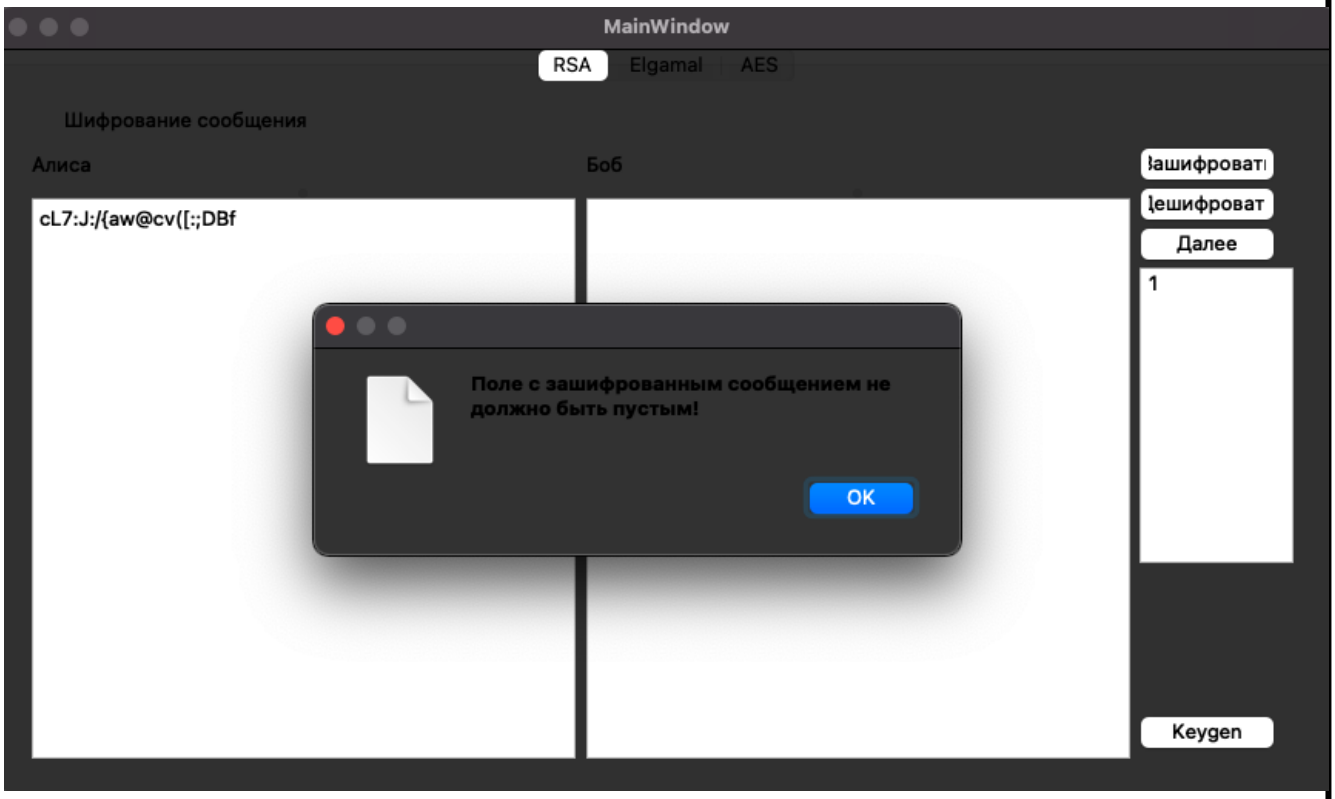



Рисунок 13 – Экранная форма проверки введенного текста с правильным ответом

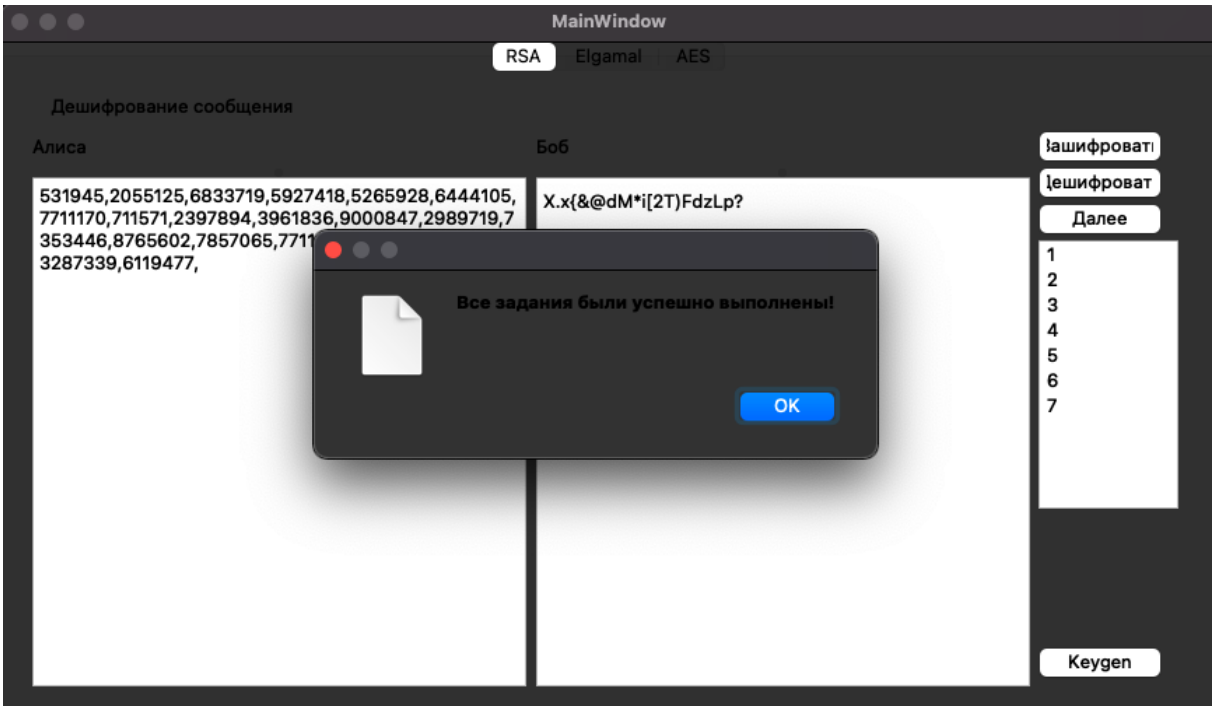



Рисунок 14 –Экранная форма результата выполнения всех шагов

## Заключение

В ходе выполнения курсового проекта было рассмотрены основные алгоритмы симметричного шифрования данных, алгоритмы генерации закрытого ключа, протоколы обмена ключами и передачи сообщений. Были рассмотрены плюсы и минусы относительно алгоритмов асимметричного шифрования.

В качестве направления дальнейшего развития можно указать добавление новых алгоритмов шифрования данных, добавление новых протоколов передачи данных.

## Приложение А

(обязательное)

Листинг программы

```
#include "mainwindow.h"
#include "ui_MainWindow.h"
#include <string>
#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent) :
      QMainWindow(parent), ui(new Ui::MainWindow) {
   ui->setupUi(this);
   ui->textEdit->setReadOnly(true);
   ui->textEdit->setMinimumWidth(200);
   ui->textEdit_2->setMinimumWidth(200);
   ui->textEdit_elgamal->setReadOnly(true);
   this->setCentralWidget(ui->tabWidget);


   gt.push_back(new GeneratorTask());
   gt.push_back(new GeneratorTask());
   gt[0]->setMaxCount(5);
   gt[0]->generateTask();
   gt[1]->setMaxCount(5);
   gt[1]->generateTask();

   rsa = new RSA();
   p_elgamal = new elgamal();

   connect(ui->action_6, SIGNAL(triggered()), this, SLOT(createKeys()));
   connect(ui->pushButton, SIGNAL(clicked(bool)), this, SLOT(slotEncrypt()));
   connect(ui->pushButton_2, SIGNAL(clicked(bool)), this, SLOT(slotDecrypt()));

   connect(ui->encryptElgamal, SIGNAL(clicked(bool)), this, SLOT(slotEncryptElgamal()));
   connect(ui->decryptElgamal, SIGNAL(clicked(bool)), this, SLOT(slotDecryptElgamal()));
   connect(ui->nextCommandRSA, SIGNAL(clicked(bool)), this, SLOT(nextCommandRSA()));
   connect(ui->nextCommandElgamal, SIGNAL(clicked(bool)), this,
SLOT(nextCommandElgamal()));
   connect(ui->generateKeys, SIGNAL(clicked(bool)), this, SLOT(slotGenerateTestKeysRSA()));
   connect(ui->generateKeysElgamal, SIGNAL(clicked(bool)), this,
SLOT(slotGenerateTestKeysElgamal()));

   ui->title->setAlignment(Qt::AlignHCenter | Qt::AlignVCenter);
   ui->title_2->setAlignment(Qt::AlignHCenter | Qt::AlignVCenter);

   ui->title->setText(gt[0]->getCurrentTask());
   ui->title_2->setText(gt[1]->getCurrentTask());
}

MainWindow::~MainWindow() {
   delete ui;
}
```

```
void MainWindow::createKeys() {
    if (ui->tabWidget->currentIndex() == 0) {
        rsa->show();
    } else if (ui->tabWidget->currentIndex() == 1) {
        p_elgamal->show();
    }
}

void MainWindow::nextCommandRSA() {
    int currentTaskType = gt[0]->getCurrentTaskType();
    switch (currentTaskType) {
        case 0: {
            if (rsa->isConfirmed()) {
                gt[0]->nextTask();
                ui->listWidget->addItem(QString::number(gt[0]->getCurrentTaskIndex()));
                if (gt[0]->isCompleted()) {
                    QMessageBox::information(this, "Успешно", "Все задания были успешно
выполнены!");
                    return;
                }
                rsa->lockLineEdits();
                ui->title->setText(gt[0]->getCurrentTask());
                int nextTaskType = gt[0]->getCurrentTaskType();
                QString text = gt[0]->generateRandomString(20);
                if (nextTaskType == 1)
                    ui->textEdit->setText(text);
                else if (nextTaskType == 2)
                    ui->textEdit->setText(rsa->toString(rsa->encrypt(text.toStdString(), {rsa->get_e(), rsa-
>get_n() })));
            } else {
                QMessageBox::information(this, "Ошибка", "Неверно указаны параметры, необходимые
для генерации ключей!");
            }
            break;
        }

        case 1: {
            QString ret = ui->textEdit_2->toPlainText();
            if (!ret.isEmpty()) {
                if (ret == rsa->toString(rsa->encrypt(ui->textEdit->toPlainText().toStdString(), {rsa-
>get_e(), rsa->get_n() }))) {
                    gt[0]->nextTask();
                    ui->listWidget->addItem(QString::number(gt[0]->getCurrentTaskIndex()));
                    if (gt[0]->isCompleted()) {
                        QMessageBox::information(this, "Успешно", "Все задания были успешно
выполнены!");
                        return;
                    }
                    ui->title->setText(gt[0]->getCurrentTask());
                    int nextTaskType = gt[0]->getCurrentTaskType();
                    QString text = gt[0]->generateRandomString(20);
                    if (nextTaskType == 0) {
```

```
                rsa->clearLineEdits();
                ui->textEdit->clear();
                ui->textEdit_2->clear();
            } else if (nextTaskType == 1)
                ui->textEdit->setText(text);
            else
                ui->textEdit->setText(rsa->toString(rsa->encrypt(text.toStdString(), {rsa->get_e(), rsa->get_n() })));
        } else {
            QMessageBox::information(this, "Ошибка", "Неверно зашифрован текст!");
        }
    } else {
        QMessageBox::information(this, "Ошибка", "Поле с зашифрованным сообщением не должно быть пустым!");
    }

    break;
}

case 2: {
    QString ret = ui->textEdit_2->toPlainText();
    if (!ret.isEmpty()) {
        if (ret == rsa->decrypt(rsa->toArray(ui->textEdit->toPlainText()), {rsa->get_d(), rsa->get_n() })) {
            gt[0]->nextTask();
            ui->listWidget->addItem(QString::number(gt[0]->getCurrentTaskIndex()));
            if (gt[0]->isCompleted()) {
                QMessageBox::information(this, "Успешно", "Все задания были успешно выполнены!");
                return;
            }
            ui->title->setText(gt[0]->getCurrentTask());
            int nextTaskType = gt[0]->getCurrentTaskType();
            QString text = gt[0]->generateRandomString(20);
            if (nextTaskType == 0) {
                rsa->clearLineEdits();
                ui->textEdit->clear();
                ui->textEdit_2->clear();
            } else if (nextTaskType == 1)
                ui->textEdit->setText(text);
            else
                ui->textEdit->setText(rsa->toString(rsa->encrypt(text.toStdString(), {rsa->get_e(), rsa->get_n() })));
        } else {
            QMessageBox::information(this, "Ошибка", "Неверно дешифрован текст!");
        }
    } else {
        QMessageBox::information(this, "Ошибка", "Поле с дешифрованным сообщением не должно быть пустым!");
    }

    break;
}
```

```
    }
}

void MainWindow::nextCommandElgamal() {
    qDebug() << "2";
    int currentTaskType = gt[1]->getCurrentTaskType();
    switch (currentTaskType) {
        case 0: {
            if (p_elgamal->isConfirmed()) {
                gt[1]->nextTask();
                ui->listWidget_3->addItem(QString::number(gt[1]->getCurrentTaskIndex()));
                if (gt[1]->isCompleted()) {
                    QMessageBox::information(this, "Успешно", "Все задания были успешно
выполнены!");
                    return;
                }
                p_elgamal->lockLineEdits();
                ui->title_2->setText(gt[1]->getCurrentTask());
                int nextTaskType = gt[1]->getCurrentTaskType();
                QString text = gt[1]->generateRandomString(20);
                if (nextTaskType == 1)
                    ui->textEdit_elgamal->setText(text);
                else if (nextTaskType == 2)
                    ui->textEdit_elgamal->setText(p_elgamal->toString(p_elgamal-
>encrypt(text.toStdString())));
            } else {
                QMessageBox::information(this, "Ошибка", "Неверно указаны параметры, необходимые
для генерации ключей!");
            }
            break;
        }

        case 1: {
            QString ret = ui->textEdit_elgamal_2->toPlainText();
            if (!ret.isEmpty()) {
                if (ret == p_elgamal->toString(p_elgamal->encrypt(ui->textEdit_elgamal-
>toPlainText().toStdString()))) {
                    gt[1]->nextTask();
                    ui->listWidget_3->addItem(QString::number(gt[1]->getCurrentTaskIndex()));
                    if (gt[1]->isCompleted()) {
                        QMessageBox::information(this, "Успешно", "Все задания были успешно
выполнены!");
                        return;
                    }
                    ui->title_2->setText(gt[1]->getCurrentTask());
                    int nextTaskType = gt[1]->getCurrentTaskType();
                    QString text = gt[1]->generateRandomString(20);
                    if (nextTaskType == 0) {
                        p_elgamal->clearLineEdits();
                        ui->textEdit_elgamal->clear();
                        ui->textEdit_elgamal_2->clear();
                    } else if (nextTaskType == 1)
                        ui->textEdit_elgamal->setText(text);
```

```
                else
                    ui->textEdit_elgamal->setText(p_elgamal->toString(p_elgamal-
>encrypt(text.toStdString())));
            } else {
                QMessageBox::information(this, "Ошибка", "Неверно зашифрован текст!");
            }
        } else {
            QMessageBox::information(this, "Ошибка", "Поле с зашифрованным сообщением не
должно быть пустым!");
        }

        break;
    }

    case 2: {
        QString ret = ui->textEdit_elgamal_2->toPlainText();
        if (!ret.isEmpty()) {
            if (ret == p_elgamal->decrypt(p_elgamal->toArray(ui->textEdit_elgamal->toPlainText()))) {
                gt[1]->nextTask();
                ui->listWidget_3->addItem(QString::number(gt[1]->getCurrentTaskIndex()));
                if (gt[1]->isCompleted()) {
                    QMessageBox::information(this, "Успешно", "Все задания были успешно
выполнены!");
                    return;
                }
                ui->title_2->setText(gt[1]->getCurrentTask());
                int nextTaskType = gt[1]->getCurrentTaskType();
                QString text = gt[1]->generateRandomString(20);
                if (nextTaskType == 0) {
                    p_elgamal->clearLineEdits();
                    ui->textEdit_elgamal->clear();
                    ui->textEdit_elgamal_2->clear();
                } else if (nextTaskType == 1)
                    ui->textEdit_elgamal->setText(text);
                else
                     ui->textEdit_elgamal->setText(p_elgamal->toString(p_elgamal-
>encrypt(text.toStdString())));
            } else {
                QMessageBox::information(this, "Ошибка", "Неверно дешифрован текст!");
            }
        } else {
            QMessageBox::information(this, "Ошибка", "Поле с дешифрованным сообщением не
должно быть пустым!");
        }

        break;
    }
  }
}

void MainWindow::slotEncrypt() {
  std::vector<uint64_t> ret = rsa->encrypt(ui->textEdit->toPlainText().toStdString(), {rsa->get_e(),
rsa->get_n() });
```

```
    QString s;
    for (auto it: ret) {
        s.append(QString::number(it) + ",");
    }
    qDebug() << s;
    ui->textEdit_2->setText(s);
}

void MainWindow::slotDecrypt() {
    QStringList sl = ui->textEdit->toPlainText().split(",");
    std::vector<uint64_t> data;
    for (auto it: sl) {
        data.push_back(it.toInt());
    }
    qDebug() << data;
    QString s = rsa->decrypt(data, {rsa->get_d(), rsa->get_n() });
    ui->textEdit_2->setText(s);
}

void MainWindow::slotEncryptElgamal() {
    std::vector<encrypt_data> ret = p_elgamal->encrypt(ui->textEdit_elgamal-
>toPlainText().toStdString());
    QString s;
    for (auto it: ret) {
        s.append(QString::number(it.a) + ";" + QString::number((it.b)) + ",");
    }
    qDebug() << s;
    ui->textEdit_elgamal_2->setText(s);
}

void MainWindow::slotDecryptElgamal() {
    QStringList sl = ui->textEdit_elgamal->toPlainText().split(",");
    std::vector<encrypt_data> data;

    for (auto it: sl) {
        QStringList _sl = it.split(";");
        if (_sl[0] != "") data.push_back({(uint64_t)_sl[0].toInt(), (uint64_t)_sl[1].toInt()});
    }
    QString s = p_elgamal->decrypt(data);
    ui->textEdit_elgamal_2->setText(s);
}

void MainWindow::slotGenerateTestKeysElgamal() {
    p_elgamal->createTemplateKeys();
}

void MainWindow::slotGenerateTestKeysRSA() {
    rsa->createTemplateKeys();
}
```

```
//
// Created by Максим Седов on 05.04.2021.
//

// You may need to build the project (run Qt uic code generator) to get "ui_AES.h" resolved

#include "aes.h"
#include "ui_aes.h"

AES::AES(int keyLen)
{
    this->Nb = 4;
    switch (keyLen)
    {
        case 128:
            this->Nk = 4;
            this->Nr = 10;
            break;
        case 192:
            this->Nk = 6;
            this->Nr = 12;
            break;
        case 256:
            this->Nk = 8;
            this->Nr = 14;
            break;
        default:
            throw "Incorrect key length";
    }

    blockBytesLen = 4 * this->Nb * sizeof(unsigned char);
}

unsigned char * AES::EncryptECB(unsigned char in[], unsigned int inLen, unsigned  char key[],
unsigned int &outLen)
{
    outLen = GetPaddingLength(inLen);
    unsigned char *alignIn  = PaddingNulls(in, inLen, outLen);
    unsigned char *out = new unsigned char[outLen];
    unsigned char *roundKeys = new unsigned char[4 * Nb * (Nr + 1)];
    KeyExpansion(key, roundKeys);
    for (unsigned int i = 0; i < outLen; i+= blockBytesLen)
    {
        EncryptBlock(alignIn + i, out + i, roundKeys);
    }

    delete[] alignIn;
    delete[] roundKeys;

    return out;
}

unsigned char * AES::DecryptECB(unsigned char in[], unsigned int inLen, unsigned  char key[])
```

```
{
    unsigned char *out = new unsigned char[inLen];
    unsigned char *roundKeys = new unsigned char[4 * Nb * (Nr + 1)];
    KeyExpansion(key, roundKeys);
    for (unsigned int i = 0; i < inLen; i+= blockBytesLen)
    {
        DecryptBlock(in + i, out + i, roundKeys);
    }

    delete[] roundKeys;

    return out;
}


unsigned char *AES::EncryptCBC(unsigned char in[], unsigned int inLen, unsigned  char key[],
unsigned char * iv, unsigned int &outLen)
{
    outLen = GetPaddingLength(inLen);
    unsigned char *alignIn  = PaddingNulls(in, inLen, outLen);
    unsigned char *out = new unsigned char[outLen];
    unsigned char *block = new unsigned char[blockBytesLen];
    unsigned char *roundKeys = new unsigned char[4 * Nb * (Nr + 1)];
    KeyExpansion(key, roundKeys);
    memcpy(block, iv, blockBytesLen);
    for (unsigned int i = 0; i < outLen; i+= blockBytesLen)
    {
        XorBlocks(block, alignIn + i, block, blockBytesLen);
        EncryptBlock(block, out + i, roundKeys);
        memcpy(block, out + i, blockBytesLen);
    }

    delete[] block;
    delete[] alignIn;
    delete[] roundKeys;

    return out;
}

unsigned char *AES::DecryptCBC(unsigned char in[], unsigned int inLen, unsigned  char key[],
unsigned char * iv)
{
    unsigned char *out = new unsigned char[inLen];
    unsigned char *block = new unsigned char[blockBytesLen];
    unsigned char *roundKeys = new unsigned char[4 * Nb * (Nr + 1)];
    KeyExpansion(key, roundKeys);
    memcpy(block, iv, blockBytesLen);
    for (unsigned int i = 0; i < inLen; i+= blockBytesLen)
    {
        DecryptBlock(in + i, out + i, roundKeys);
        XorBlocks(block, out + i, out + i, blockBytesLen);
        memcpy(block, in + i, blockBytesLen);
    }
```

```
    delete[] block;
    delete[] roundKeys;

    return out;
}

unsigned char *AES::EncryptCFB(unsigned char in[], unsigned int inLen, unsigned  char key[],
unsigned char * iv, unsigned int &outLen)
{
    outLen = GetPaddingLength(inLen);
    unsigned char *alignIn  = PaddingNulls(in, inLen, outLen);
    unsigned char *out = new unsigned char[outLen];
    unsigned char *block = new unsigned char[blockBytesLen];
    unsigned char *encryptedBlock = new unsigned char[blockBytesLen];
    unsigned char *roundKeys = new unsigned char[4 * Nb * (Nr + 1)];
    KeyExpansion(key, roundKeys);
    memcpy(block, iv, blockBytesLen);
    for (unsigned int i = 0; i < outLen; i+= blockBytesLen)
    {
        EncryptBlock(block, encryptedBlock, roundKeys);
        XorBlocks(alignIn + i, encryptedBlock, out + i, blockBytesLen);
        memcpy(block, out + i, blockBytesLen);
    }

    delete[] block;
    delete[] encryptedBlock;
    delete[] alignIn;
    delete[] roundKeys;

    return out;
}

unsigned char *AES::DecryptCFB(unsigned char in[], unsigned int inLen, unsigned  char key[],
unsigned char * iv)
{
    unsigned char *out = new unsigned char[inLen];
    unsigned char *block = new unsigned char[blockBytesLen];
    unsigned char *encryptedBlock = new unsigned char[blockBytesLen];
    unsigned char *roundKeys = new unsigned char[4 * Nb * (Nr + 1)];
    KeyExpansion(key, roundKeys);
    memcpy(block, iv, blockBytesLen);
    for (unsigned int i = 0; i < inLen; i+= blockBytesLen)
    {
        EncryptBlock(block, encryptedBlock, roundKeys);
        XorBlocks(in + i, encryptedBlock, out + i, blockBytesLen);
        memcpy(block, in + i, blockBytesLen);
    }

    delete[] block;
    delete[] encryptedBlock;
    delete[] roundKeys;
```

```
    return out;
}

unsigned char * AES::PaddingNulls(unsigned char in[], unsigned int inLen, unsigned int alignLen)
{
    unsigned char *alignIn = new unsigned char[alignLen];
    memcpy(alignIn, in, inLen);
    memset(alignIn + inLen, 0x00, alignLen - inLen);
    return alignIn;
}

unsigned int AES::GetPaddingLength(unsigned int len)
{
    unsigned int lengthWithPadding =  (len / blockBytesLen);
    if (len % blockBytesLen) {
        lengthWithPadding++;
    }

    lengthWithPadding *=  blockBytesLen;

    return lengthWithPadding;
}

void AES::EncryptBlock(unsigned char in[], unsigned char out[], unsigned  char *roundKeys)
{
    unsigned char **state = new unsigned char *[4];
    state[0] = new unsigned  char[4 * Nb];
    int i, j, round;
    for (i = 0; i < 4; i++)
    {
        state[i] = state[0] + Nb * i;
    }


    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < Nb; j++)
        {
            state[i][j] = in[i + 4 * j];
        }
    }

    AddRoundKey(state, roundKeys);

    for (round = 1; round <= Nr - 1; round++)
    {
        SubBytes(state);
        ShiftRows(state);
        MixColumns(state);
        AddRoundKey(state, roundKeys + round * 4 * Nb);
    }

    SubBytes(state);
```

```
    ShiftRows(state);
    AddRoundKey(state, roundKeys + Nr * 4 * Nb);

    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < Nb; j++)
        {
            out[i + 4 * j] = state[i][j];
        }
    }

    delete[] state[0];
    delete[] state;
}

void AES::DecryptBlock(unsigned char in[], unsigned char out[], unsigned  char *roundKeys)
{
    unsigned char **state = new unsigned char *[4];
    state[0] = new unsigned  char[4 * Nb];
    int i, j, round;
    for (i = 0; i < 4; i++)
    {
        state[i] = state[0] + Nb * i;
    }


    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < Nb; j++) {
            state[i][j] = in[i + 4 * j];
        }
    }

    AddRoundKey(state, roundKeys + Nr * 4 * Nb);

    for (round = Nr - 1; round >= 1; round--)
    {
        InvSubBytes(state);
        InvShiftRows(state);
        AddRoundKey(state, roundKeys + round * 4 * Nb);
        InvMixColumns(state);
    }

    InvSubBytes(state);
    InvShiftRows(state);
    AddRoundKey(state, roundKeys);

    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < Nb; j++) {
            out[i + 4 * j] = state[i][j];
        }
    }
```

```
    delete[] state[0];
    delete[] state;
}


void AES::SubBytes(unsigned char **state)
{
    int i, j;
    unsigned char t;
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < Nb; j++)
        {
            t = state[i][j];
            state[i][j] = sbox[t / 16][t % 16];
        }
    }

}

void AES::ShiftRow(unsigned char **state, int i, int n)    // shift row i on n positions
{
    unsigned char *tmp = new unsigned char[Nb];
    for (int j = 0; j < Nb; j++) {
        tmp[j] = state[i][(j + n) % Nb];
    }
    memcpy(state[i], tmp, Nb * sizeof(unsigned char));

    delete[] tmp;
}

void AES::ShiftRows(unsigned char **state)
{
    ShiftRow(state, 1, 1);
    ShiftRow(state, 2, 2);
    ShiftRow(state, 3, 3);
}

unsigned char AES::xtime(unsigned char b)    // multiply on x
{
    return (b << 1) ^ (((b >> 7) & 1) * 0x1b);
}



/* Implementation taken from
https://en.wikipedia.org/wiki/Rijndael_mix_columns#Implementation_example */
void AES::MixSingleColumn(unsigned char *r)
{
    unsigned char a[4];
    unsigned char b[4];
    unsigned char c;
```

```
    unsigned char h;
    /* The array 'a' is simply a copy of the input array 'r'
    * The array 'b' is each element of the array 'a' multiplied by 2
    * in Rijndael's Galois field
    * a[n] ^ b[n] is element n multiplied by 3 in Rijndael's Galois field */
    for(c=0;c<4;c++)
    {
        a[c] = r[c];
        /* h is 0xff if the high bit of r[c] is set, 0 otherwise */
        h = (unsigned char)((signed char)r[c] >> 7); /* arithmetic right shift, thus shifting in either zeros
or ones */
        b[c] = r[c] << 1; /* implicitly removes high bit because b[c] is an 8-bit char, so we xor by 0x1b
and not 0x11b in the next line */
        b[c] ^= 0x1B & h; /* Rijndael's Galois field */
    }
    r[0] = b[0] ^ a[3] ^ a[2] ^ b[1] ^ a[1]; /* 2 * a0 + a3 + a2 + 3 * a1 */
    r[1] = b[1] ^ a[0] ^ a[3] ^ b[2] ^ a[2]; /* 2 * a1 + a0 + a3 + 3 * a2 */
    r[2] = b[2] ^ a[1] ^ a[0] ^ b[3] ^ a[3]; /* 2 * a2 + a1 + a0 + 3 * a3 */
    r[3] = b[3] ^ a[2] ^ a[1] ^ b[0] ^ a[0]; /* 2 * a3 + a2 + a1 + 3 * a0 */
}

/* Performs the mix columns step. Theory from:
https://en.wikipedia.org/wiki/Advanced_Encryption_Standard#The_MixColumns_step */
void AES::MixColumns(unsigned char** state)
{
    unsigned char *temp = new unsigned char[4];

    for(int i = 0; i < 4; ++i)
    {
        for(int j = 0; j < 4; ++j)
        {
            temp[j] = state[j][i]; //place the current state column in temp
        }
        MixSingleColumn(temp); //mix it using the wiki implementation
        for(int j = 0; j < 4; ++j)
        {
            state[j][i] = temp[j]; //when the column is mixed, place it back into the state
        }
    }
    delete[] temp;
}

void AES::AddRoundKey(unsigned char **state, unsigned char *key)
{
    int i, j;
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < Nb; j++)
        {
            state[i][j] = state[i][j] ^ key[i + 4 * j];
        }
    }
}
```

```
void AES::SubWord(unsigned char *a)
{
   int i;
   for (i = 0; i < 4; i++)
   {
      a[i] = sbox[a[i] / 16][a[i] % 16];
   }
}

void AES::RotWord(unsigned char *a)
{
   unsigned char c = a[0];
   a[0] = a[1];
   a[1] = a[2];
   a[2] = a[3];
   a[3] = c;
}

void AES::XorWords(unsigned char *a, unsigned char *b, unsigned char *c)
{
   int i;
   for (i = 0; i < 4; i++)
   {
      c[i] = a[i] ^ b[i];
   }
}

void AES::Rcon(unsigned char * a, int n)
{
   int i;
   unsigned char c = 1;
   for (i = 0; i < n - 1; i++)
   {
      c = xtime(c);
   }

   a[0] = c;
   a[1] = a[2] = a[3] = 0;
}

void AES::KeyExpansion(unsigned char key[], unsigned char w[])
{
   unsigned char *temp = new unsigned char[4];
   unsigned char *rcon = new unsigned char[4];

   int i = 0;
   while (i < 4 * Nk)
   {
      w[i] = key[i];
      i++;
   }
```

```
    i = 4 * Nk;
    while (i < 4 * Nb * (Nr + 1))
    {
        temp[0] = w[i - 4 + 0];
        temp[1] = w[i - 4 + 1];
        temp[2] = w[i - 4 + 2];
        temp[3] = w[i - 4 + 3];

        if (i / 4 % Nk == 0)
        {
            RotWord(temp);
            SubWord(temp);
            Rcon(rcon, i / (Nk * 4));
            XorWords(temp, rcon, temp);
        }
        else if (Nk > 6 && i / 4 % Nk == 4)
        {
            SubWord(temp);
        }

        w[i + 0] = w[i - 4 * Nk] ^ temp[0];
        w[i + 1] = w[i + 1 - 4 * Nk] ^ temp[1];
        w[i + 2] = w[i + 2 - 4 * Nk] ^ temp[2];
        w[i + 3] = w[i + 3 - 4 * Nk] ^ temp[3];
        i += 4;
    }

    delete []rcon;
    delete []temp;

}


void AES::InvSubBytes(unsigned char **state)
{
    int i, j;
    unsigned char t;
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < Nb; j++)
        {
            t = state[i][j];
            state[i][j] = inv_sbox[t / 16][t % 16];
        }
    }
}


unsigned char AES::mul_bytes(unsigned char a, unsigned char b) // multiplication a and b in galois
field
{
    unsigned char p = 0;
    unsigned char high_bit_mask = 0x80;
```

```
    unsigned char high_bit = 0;
    unsigned char modulo = 0x1B; /* x^8 + x^4 + x^3 + x + 1 */


    for (int i = 0; i < 8; i++) {
        if (b & 1) {
            p ^= a;
        }

        high_bit = a & high_bit_mask;
        a <<= 1;
        if (high_bit) {
            a ^= modulo;
        }
        b >>= 1;
    }

    return p;
}


void AES::InvMixColumns(unsigned char **state)
{
    unsigned char s[4], s1[4];
    int i, j;

    for (j = 0; j < Nb; j++)
    {
        for (i = 0; i < 4; i++)
        {
            s[i] = state[i][j];
        }
        s1[0] = mul_bytes(0x0e, s[0]) ^ mul_bytes(0x0b, s[1]) ^ mul_bytes(0x0d, s[2]) ^
mul_bytes(0x09, s[3]);
        s1[1] = mul_bytes(0x09, s[0]) ^ mul_bytes(0x0e, s[1]) ^ mul_bytes(0x0b, s[2]) ^
mul_bytes(0x0d, s[3]);
        s1[2] = mul_bytes(0x0d, s[0]) ^ mul_bytes(0x09, s[1]) ^ mul_bytes(0x0e, s[2]) ^
mul_bytes(0x0b, s[3]);
        s1[3] = mul_bytes(0x0b, s[0]) ^ mul_bytes(0x0d, s[1]) ^ mul_bytes(0x09, s[2]) ^
mul_bytes(0x0e, s[3]);

        for (i = 0; i < 4; i++)
        {
            state[i][j] = s1[i];
        }
    }
}

void AES::InvShiftRows(unsigned char **state)
{
    ShiftRow(state, 1, Nb - 1);
    ShiftRow(state, 2, Nb - 2);
    ShiftRow(state, 3, Nb - 3);
```

```
}

void AES::XorBlocks(unsigned char *a, unsigned char * b, unsigned char *c, unsigned int len)
{
   for (unsigned int i = 0; i < len; i++)
   {
      c[i] = a[i] ^ b[i];
   }
}

void AES::printHexArray (unsigned char a[], unsigned int n)
{
   for (unsigned int i = 0; i < n; i++) {
      printf("%02x ", a[i]);
   }
}


AES::~AES() {
   delete ui;
}

#include "elgamal.h"
#include "ui_elgamal.h"

elgamal::elgamal(QWidget *parent) :
   QWidget(parent),
   ui(new Ui::elgamal)
{
   ui->setupUi(this);
}

elgamal::~elgamal()
{
   delete ui;
}

void elgamal::gen_keys(uint64_t p)
{
   this->g = 123;
   this->p = p;
   this->x = 8;
   this->k = rand() % (this->p - 2) + 1;
   this->y = this->binpow(g, x, p);
}

std::vector<encrypt_data> elgamal::encrypt(std::string text) {
   std::vector<encrypt_data> ret(text.length());

   for (uint i = 0; i < text.length(); i++) {
      uint64_t a = binpow(this->g, k, this->p);
      uint64_t b = mul(binpow(this->y, k, this->p), (int) text[i], this->p);
      ret[i] = {a, b};
```

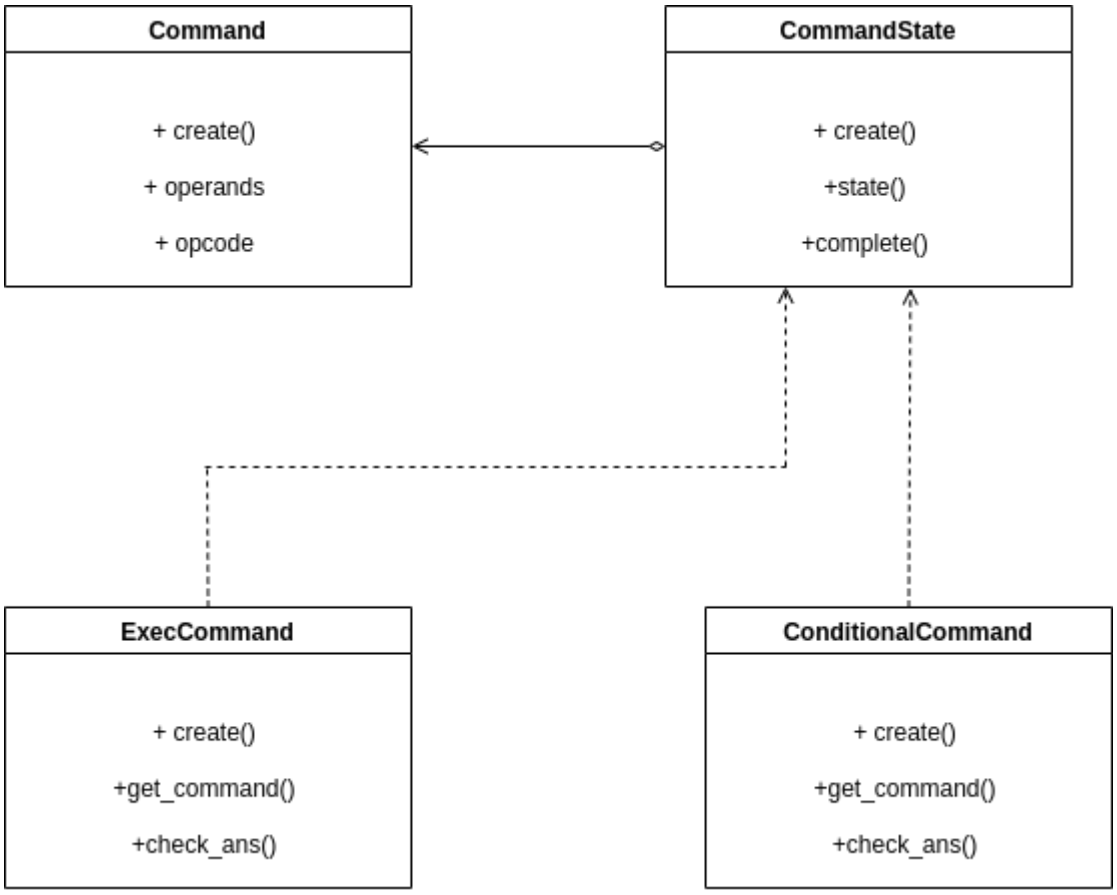
```
    }
    return ret;
}

QString elgamal::decrypt(std::vector<encrypt_data> data) {
    QString ret;
    for (uint i = 0; i < data.size(); i++) {
        ret.append((char)this->mul(data[i].b, this->binpow(data[i].a, this->p - 1 - this->x, this->p), this->p));
    }
    return ret;
}

void elgamal::lockLineEdits() {
    ui->lineEdit->setReadOnly(true);
    ui->lineEdit_2->setReadOnly(true);
    ui->lineEdit_3->setReadOnly(true);
    ui->lineEdit_4->setReadOnly(true);
    ui->lineEdit_5->setReadOnly(true);
}

void elgamal::clearLineEdits() {
    ui->lineEdit->clear();
    ui->lineEdit_2->clear();
    ui->lineEdit_3->clear();
    ui->lineEdit_4->clear();
    ui->lineEdit_5->clear();
}

void elgamal::createTemplateKeys() {
    this->gen_keys(593);
    ui->lineEdit->setText(QString::number(this->p));
    ui->lineEdit_2->setText(QString::number(this->g));
    ui->lineEdit_3->setText(QString::number(this->x));
    ui->lineEdit_4->setText(QString::number(this->k));
    ui->lineEdit_5->setText(QString::number(this->y));
}

bool elgamal::isConfirmed() {
    if (!ui->lineEdit->text().isEmpty() &&
        !ui->lineEdit_2->text().isEmpty() &&
        !ui->lineEdit_3->text().isEmpty() &&
        !ui->lineEdit_4->text().isEmpty() &&
        !ui->lineEdit_5->text().isEmpty()) return true;
    return false;
}

QString elgamal::toString(std::vector<encrypt_data> data) {
    QString s;
    for (auto it: data) {
        s.append(QString::number(it.a) + ";" + QString::number((it.b)) + ",");
    }
    return s;
```

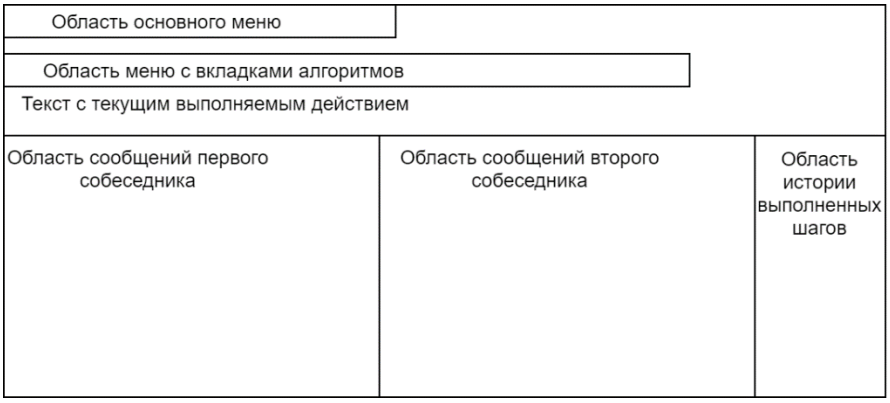
```
}

std::vector<encrypt_data> elgamal::toArray(QString s) {
    QStringList sl = s.split(",");
    std::vector<encrypt_data> data;

    for (auto it: sl) {
        QStringList _sl = it.split(";");
        if (_sl[0] != "") data.push_back({(uint64_t)_sl[0].toInt(), (uint64_t)_sl[1].toInt()});
    }
    return data;
}

uint64_t elgamal::mul(uint64_t a, uint64_t b, uint64_t n) {
    uint64_t sum=0;

    for(uint i = 0; i < b; i++){
        sum += a;
        if(sum>=n){
            sum -= n;
        }
    }
    return sum;
}

int64_t elgamal::gcdex(int64_t a, int64_t b, int64_t &x, int64_t &y) {
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }
    int64_t x1, y1;
    int64_t d = gcdex(b % a, a, x1, y1);
    x = y1 - (b / a) * x1;
    y = x1;
    return d;

}

int64_t elgamal::invmod(int64_t a, int64_t m) {
    int64_t x, y;
    gcdex(a, m, x, y);
    x = (x % m + m) % m;
    return x;
}
```

**Приложение Б**

(обязательное)

Диаграммы классов

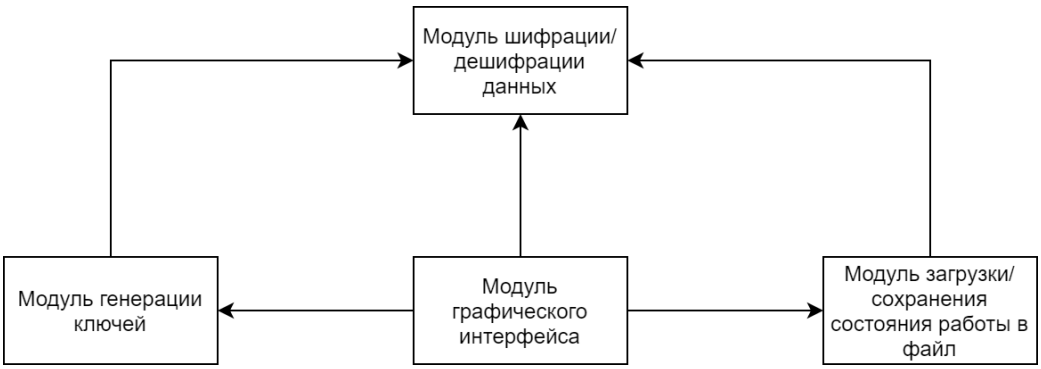



Область основного меню
Область меню с вкладками алгоритмов
Текст с текущим выполняемым действием
Область сообщений первого собеседника
Область сообщений второго собеседника
Область истории выполненных шагов

Приложение В

(обязательное)

Модульная структура приложения



Приложение Г

(справочное)

Список использованных источников и материалов


- GNU GCC [Электронный ресурс]. - https://gcc.gnu.org/
- Асимметричное шифрование на практике [Электронный ресурс]. - https://habr.com/ru/post/449552/
- Алгоритм RSA [Электронный ресурс]. - https://ru.wikipedia.org/wiki/RSA
- Асимметричное шифрование [Электронный ресурс]. - https://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F16.pdf

Приложение Г

(обязательное)

Библиографический список

1. ГОСТ 19.701–90 Единая система программной документации. М.: Изд-во стандартов, 1990.

2. Брюс Шнайер Прикладная криптография: Изд-во Диалектика, 2016. – 610 с.