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)
«__»_____2020г.

РАЗРАБОТКА ПРОГРАММЫ МОДЕЛИРОВАНИЯ ПЕРЕДАЧИ СООБЩЕНИЙ С АССИМЕТРИЧНЫМ 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 2020

Реферат

Седов М.Д. Разработка программы моделирования передачи сообщений с ассиметричным шифрованием данных: ТПЖА.090301014 ПЗ: Курс. работа / ВятГУ, каф. ЭВМ; рук. Караваева О.В. - Киров, 2020. – ПЗ 41 с, 10 рис., 2 табл., 9 источников.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ, АССИМЕТРИЧНОЕ ШИФРОВАНИЕ, ДЕШИФРОВАНИЕ, ОТКРЫТЫЙ КЛЮЧ, ЗАКРЫТЫЙ КЛЮЧ, ЗАЩИТА ИНФОРМАЦИИ, QT, CMAKE, C++

Цель курсового проекта — повышение качества обучения за счет использования программного комплекса моделирования передачи сообщений с ассиметричным 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 ассиметричным шифрованием данных.

1. **Анализ предметной области**

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

1.1 Ассиметричное шифрование

Концепцию криптографии с открытым ключом была выдвинута Уитфилдом Диффи и Мартином Хеллманов, и независимо Ральфом Мерклом. Их вкладом в криптографию было убеждение, что ключи можно использовать парами – ключ шифрования и ключ дешифрования, и что может быть невозможно получить один ключ из другого. Некоторые из алгоритмов шифрования подходят только для распределения ключей. Другие - наоборот подходят только для цифровых подписей. Алгоритмы, которые хорошо работают как для цифровых подписей, так и для шифрования, являются RSA, ElGamal и Rabin. Они шифруют и дешифруют данные намного медленнее, чем симметричные алгоритмы. Обычно их скорость недостаточна для шифрования больших объемов данных. Гибридные системы обычно позволяют ускорить процесс: для шифрования сообщения используется симметричный алгоритм, а алгоритм с открытым ключом применяется шифрования случайного сеансового ключа.

Алгоритмы криптографии с открытым ключом можно использовать:

1) как самостоятельное средство для защиты передаваемой и хранимой информации,
2) как средство распределения ключей,
3) как средство аутентификации пользователей.

1.1.1 Генерация ключей

Ассиметричное шифрование — система шифрования и/или электронной подписи (ЭП), при которой открытый ключ передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу и используется для проверки ЭП и для шифрования сообщения. Для генерации ЭП и для расшифровки сообщения используется закрытый ключ

Асимметричное шифрование с открытым ключом базируется на следующих принципах:

1) Можно сгенерировать пару очень больших чисел (открытый ключ и закрытый ключ) так, чтобы, зная открытый ключ, нельзя было вычислить закрытый ключ за разумный срок. При этом механизм генерации является общеизвестным.
2) Имеются надёжные методы шифрования, позволяющие зашифровать сообщение открытым ключом так, чтобы расшифровать его можно было только закрытым ключом. Механизм шифрования является общеизвестным.
3) Владелец двух ключей никому не сообщает закрытый ключ, но передает открытый ключ контрагентам или делает его общеизвестным.

Если необходимо передать зашифрованное сообщение владельцу ключей, то отправитель должен получить открытый ключ. Отправитель шифрует свое сообщение открытым ключом получателя и передает его получателю (владельцу ключей) по открытым каналам. При этом расшифровать сообщение не может никто, кроме владельца закрытого ключа.

Односторонняя функция – функция f(x) такая, что по известному значению x легко вычисляется f(x), но вычисление x с известным f(x) невозможно за разумный срок.

Чаще всего криптография с открытым ключом связана с односторонними функциями. Но сами открытые функции из себя мало что представляют. Вместе с ними используется лазейка(секрет), который помогает расшифровать сообщение

без каких-либо проблем. Пусть лазейка равна такому y, что зная f(x) и y, можно вычислить x.

Данный принцип можно рассмотреть на примере конструктора. Если разобрать конструктор, то имея только его, будет довольно сложно собрать обратно. Но если рядом будет находиться инструкция(лазейка), то сборка уже займет гораздо меньше времени.

Применительно к криптографии лазейка это не что иное, как закрытый ключ. Получатель информации формирует открытый ключ и закрытый ключ, затем передает открытый ключ отправителю, а закрытый оставляет у себя. Отправитель шифрует информацию на основе открытого ключа: такую зашифрованную информацию просто расшифровать, лишь имея одновременно и открытый ключ, и закрытый ключ.

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) RSA(Rivest, Shamir, Adleman). Самый распространенный алгоритм, основанный на вычислительной сложности задачи факторизации больших целых чисел. RSA стал первый пригодным как для шифрования, так и для цифровой подписи.
2) Схема Эль-Гамаля(Elgamal). Криптосистема, основанная на трудности вычисления дискретных логарифмов в конечном поле. Включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема Эль-Гамаля лежит в основе бывших стандартов электронной цифровой подписи в США (DSA) и России (ГОСТ Р 34.10-94).
3) Эллиптическая криптография(Elliptic curve cryptosystem). Алгоритмы криптографии, основанные на эллиптических кривых над конечным полем. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи
4) Протокол Диффи — Хеллмана(DH). Протокол, позволяющий двум и более сторонам получить общий секретный ключ, используя незащищенный от прослушивания канал связи. Полученный ключ используется для шифрования дальнейшего обмена с помощью алгоритмов симметричного шифрования. Схема открытого распределения ключей, предложенная Диффи и Хеллманом, произвела настоящую революцию в мире шифрования, так как снимала основную проблему классической криптографии — проблему распределения ключей. В чистом виде алгоритм Диффи — Хеллмана уязвим для модификации данных в канале связи, в том числе для атаки «Man-in-the-middle (человек посередине)», поэтому схемы с его использованием применяют дополнительные методы односторонней или двусторонней аутентификации.

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.

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 ассиметричным 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 открытого и закрытого ключей;
5) функции шифрации данных на основе открытого ключа;
6) функции дешифрации данных на основе закрытого ключа
7) функции, предоставляющие модель передачи сообщений.

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.

2.1.1 Алгоритм обработки поступающих заявок

Алгоритм обработки заявки «Генерация ключей алгоритмом RSA»

1) выбираются два различных (желательно, с описанными здесь свойствами) случайных простых числа p и q заданного размера
2) вычисляется их произведение n=p*q, которое называется модулем;
3) вычисляется значение функции Эйлера от числа n:

$$\varphi(n) = (p-1) * (q-1)$$

4) выбирается целое число e (1<e< $\varphi(n)$, взаимно простое со значением функции $\varphi(n)$;
5) вычисляется число d, мультипликативно обратное к числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее соответствующему сравнению:

$$d * e \equiv 1 \ (mod \ \varphi(n))$$

6) пара (e,n) публикуется в качестве открытого ключа RSA;
7) пара (d,n) играет роль закрытого ключа RSA.

Схема генерации ключей алгоритмом RSA представлена на рисунке 1.

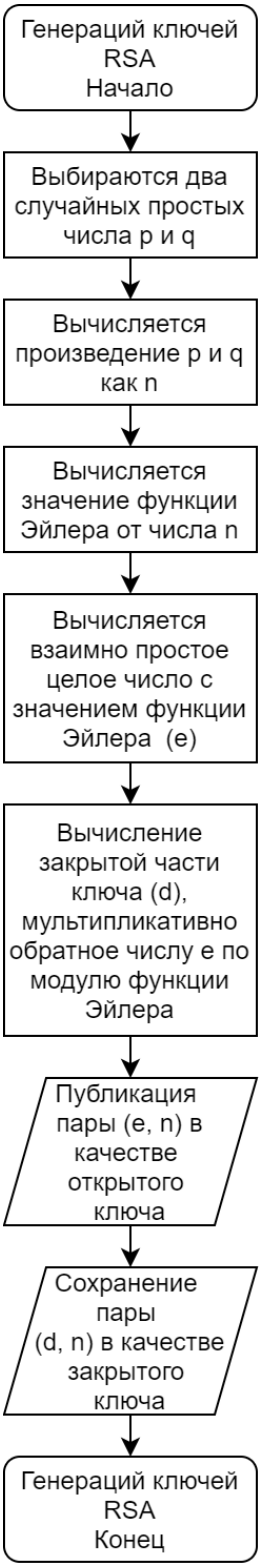


Рисунок 1 — Схема алгоритма генерации ключей RSA

Алгоритм обработки заявки «Генерация ключей алгоритмом Elgamal»

1) Генерируется случайное простое число p;
2) Выбирается целое число g — первообразный корень p;
3) Выбирается случайное целое число, взаимно простое с (p - 1), x такое, что $1<x<p-1$;
4) Вычисляется $y=g^x \bmod p$;
5) Открытым ключом является y, закрытым – x.

Схема генерации ключей алгоритмом Elgamal представлена на рисунке 2.

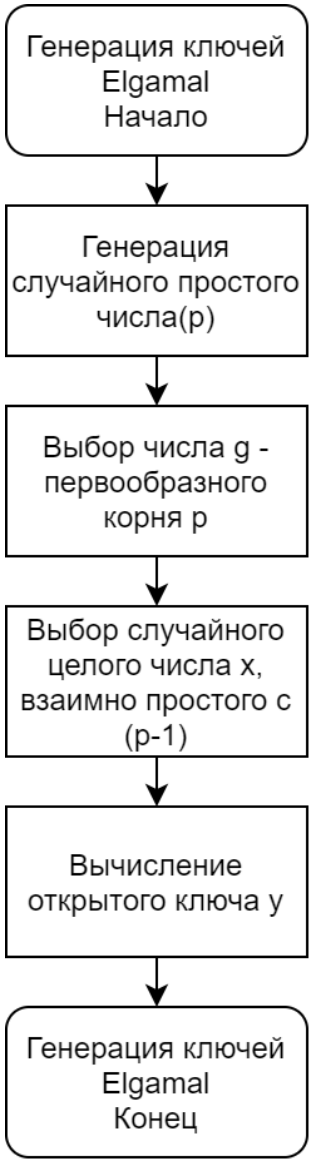


Рисунок 2 — Схема алгоритма генерации ключей Elgamal

Алгоритм обработки заявки «Генерация ключей алгоритмом DH»

1) Генерация случайного натурального числа a — закрытый ключ;
2) Совместно с удалённой стороной устанавливаются открытые параметры p и g (обычно значения p и g генерируются на одной стороне и передаются другой);
3) Вычисляется открытый ключ A, используя преобразование над закрытым ключом;
4) Обмен открытыми ключами с удалённой стороной;
5) Вычисляется общий секретный ключ K, используя открытый ключ удаленной стороны B и свой закрытый ключ a.

Схема генерации ключей алгоритмом DH представлена на рисунке 2.

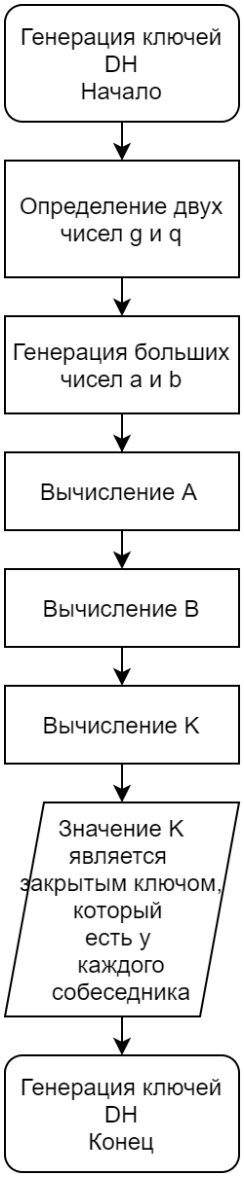


Рисунок 3 — Схема алгоритма генерации ключей DH

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.

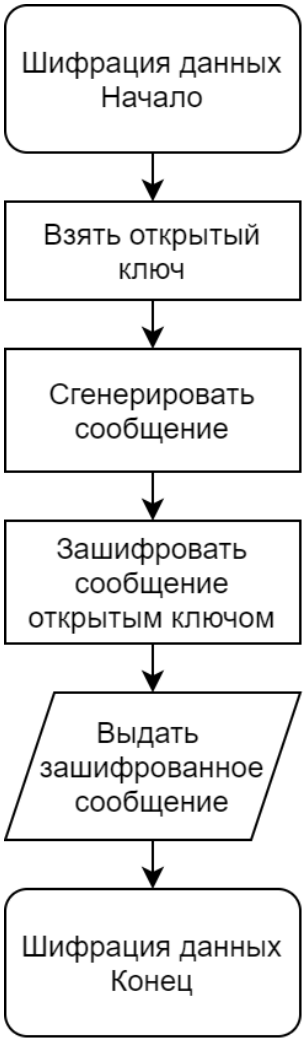


Рисунок 4 — Схема алгоритма шифрации данных

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.

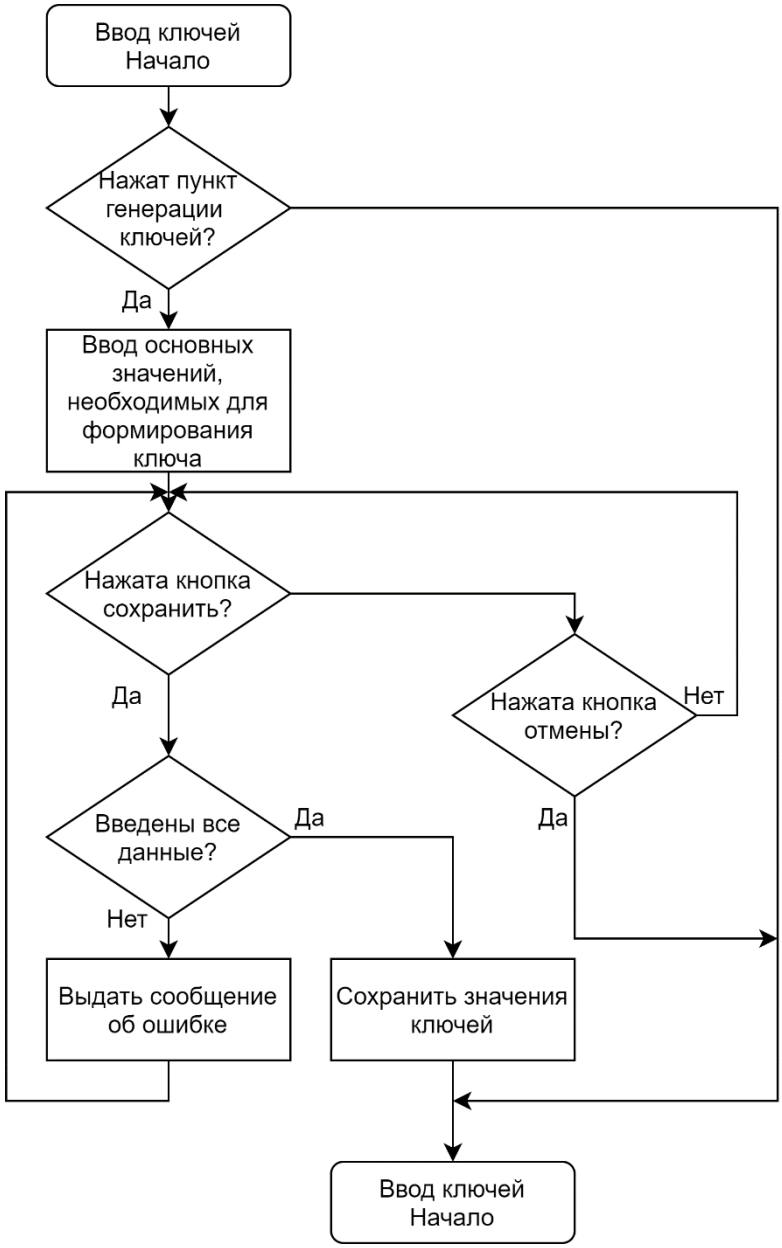


Рисунок 6 — Схема алгоритма ввода данных для генерации ключей

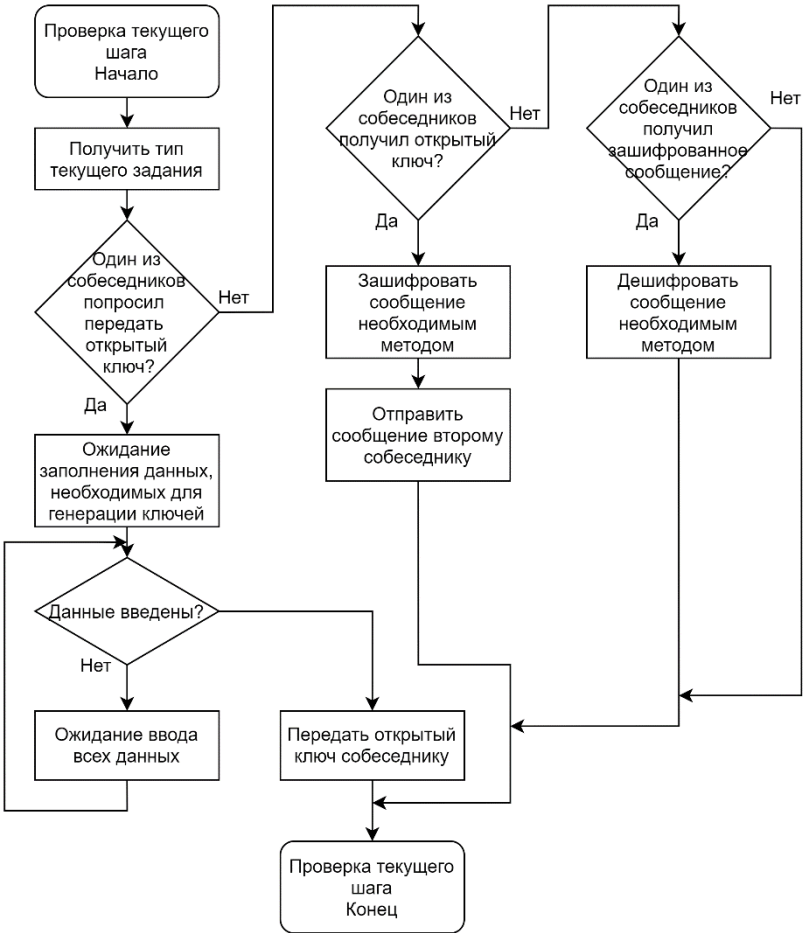


Рисунок 7 — Схема алгоритма выполнения текущего шага лабораторной работы

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.

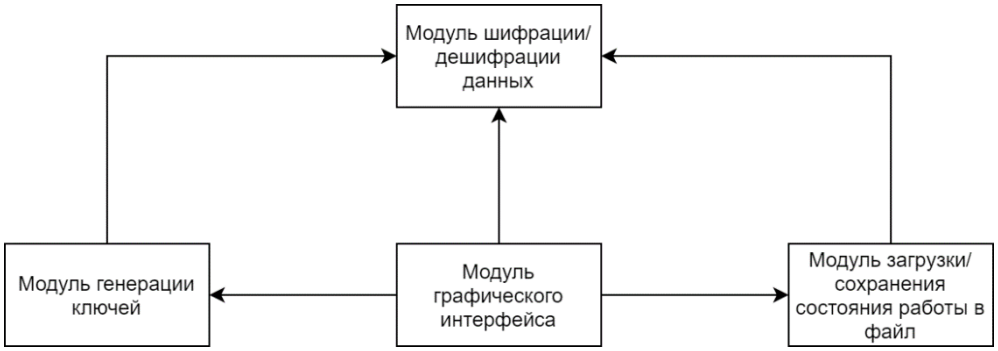


Рисунок 8 — Модульная структура приложения

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.

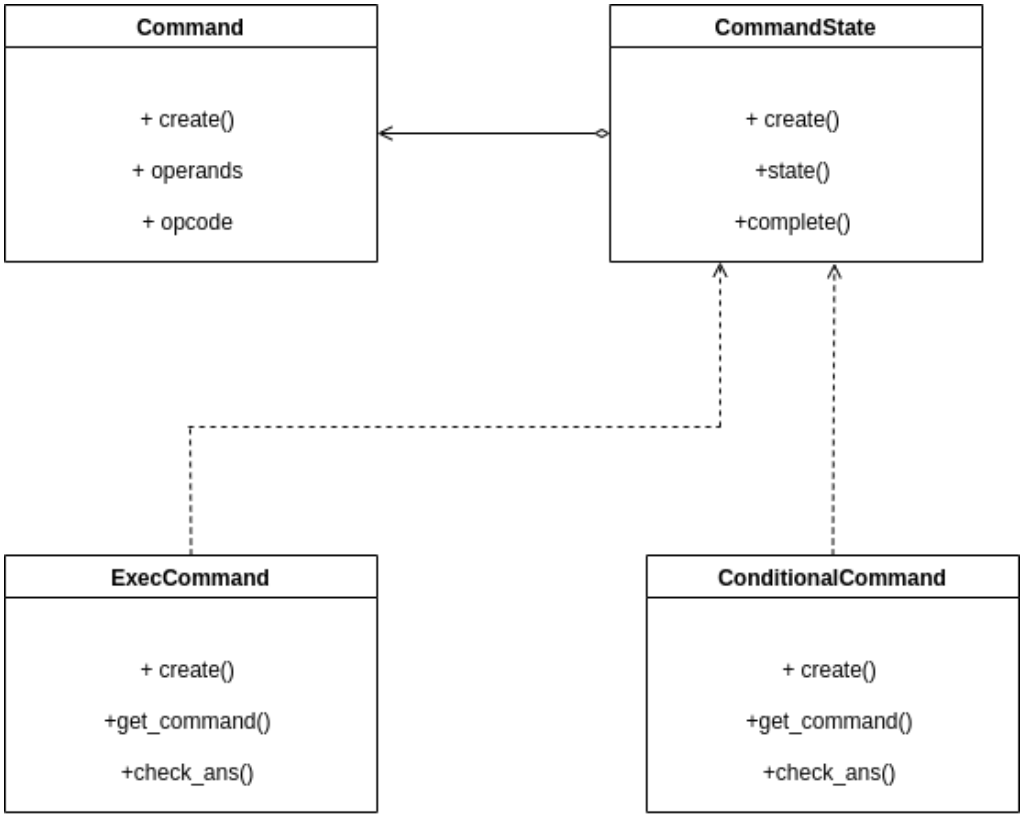


Рисунок 9 — Диаграмма классов модуля обработки команд

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.

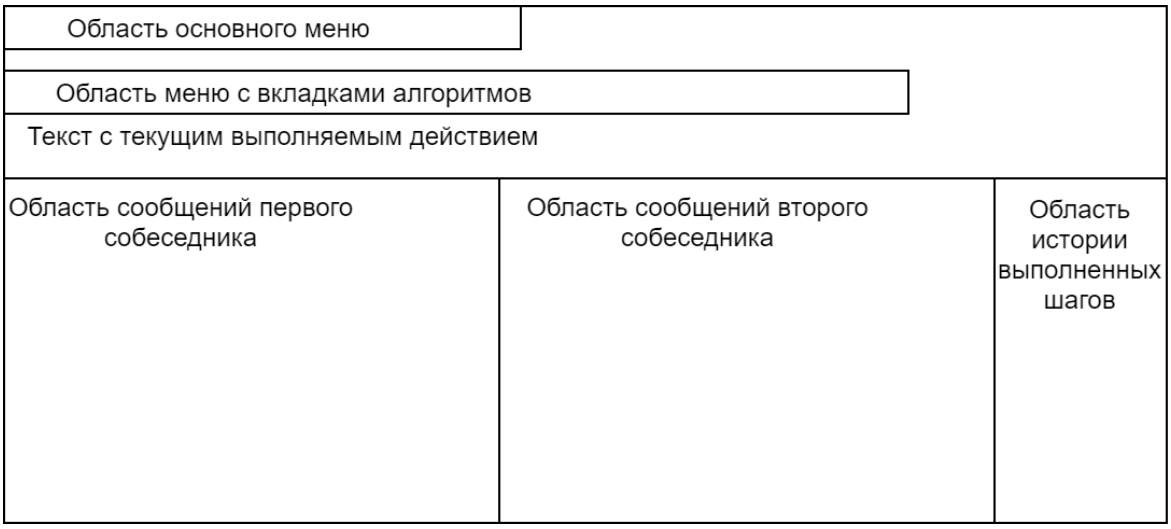


Рисунок 10 — Схема модуля графического интерфейса

.

Заключение

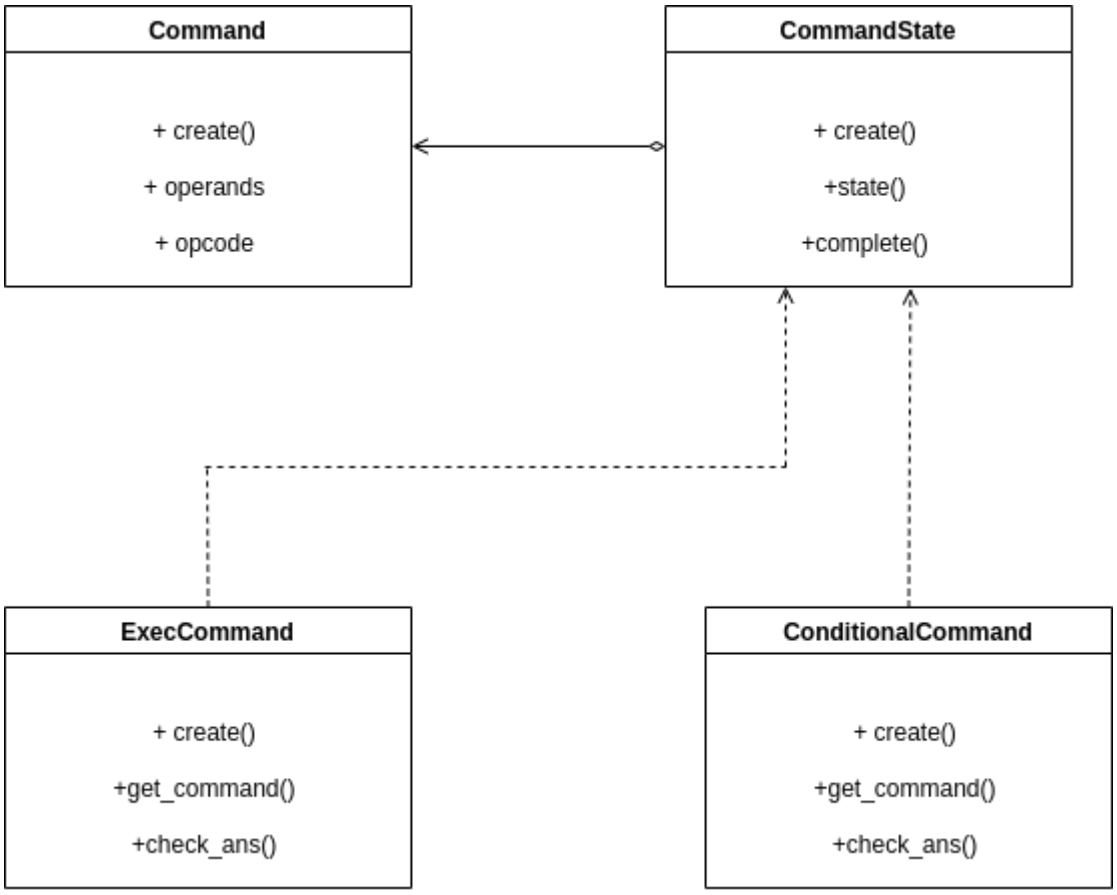
В ходе выполнения курсового проекта было рассмотрены основные алгоритмы ассиметричного шифрования данных, алгоритмы генерации открытого и закрытого ключей. Были рассмотрены плюсы и минусы относительно алгоритмов симметричного шифрования.

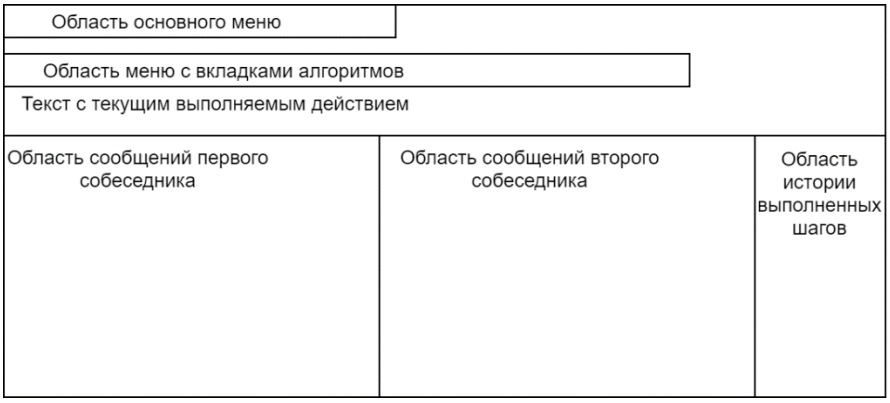
В качестве направления дальнейшего развития можно выбрать разработку алгоритмов симметричного шифрования, а также возможность создания ситуаций перехвата сообщений третьей стороной с последующем изменением.

Приложение А

(обязательное)

Диаграммы классов

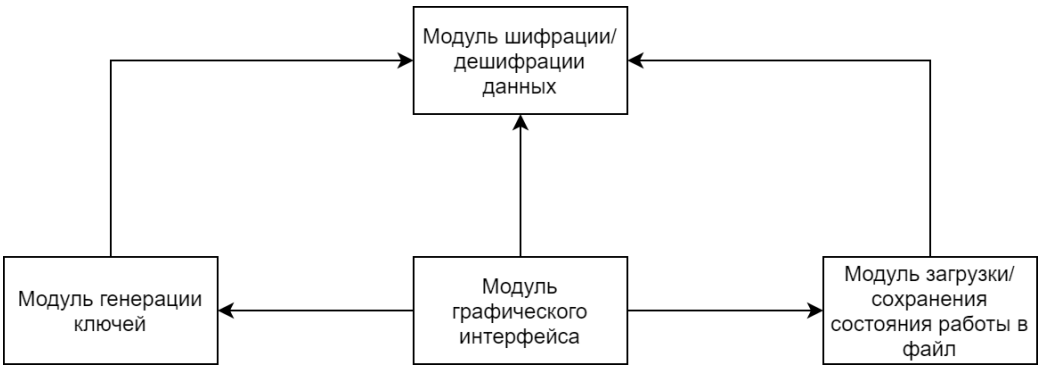



Область основного меню
Область меню с вкладками алгоритмов
Текст с текущим выполняемым действием
Область сообщений первого собеседника
Область сообщений второго собеседника
Область истории выполненных шагов

Приложение Б

(обязательное)

Модульная структура приложения



Приложение В

(справочное)

Список использованных источников и материалов

- https://gcc.gnu.org/
- https://habr.com/ru/post/449552/
- https://ru.wikipedia.org/wiki/RSA
- https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%94%D0%B8%D1%84%D1%84%D0%B8_%E2%80%94_%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0
- https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F
- https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%AD%D0%BB%D1%8C-%D0%93%D0%B0%D0%BC%D0%B0%D0%BB%D1%8F
- 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.