МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему

ПО приложения для передачи конфиденциальных данных по протоколу UDP

Студент _____ Грушевский Александр Анатольевич
(подпись, дата)

Руководитель _____ Гурин Е.И.
(подпись, дата)

Нормоконтролёр _____ Кучин А.В.
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой _____ Митрохин М.А.
(подпись)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Гурин Е.И.
(подпись)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____ Митрохин М.А.
«30» апреля 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Грушевский Александр Анатольевич гр. 16ВВ1 факультета вычислительной техники.

2. Тема работы: «ПО приложения для передачи конфиденциальных данных по протоколу UDP».

Тема утверждена приказом ПГУ №0335а/о от "30" апреля 2020 г.

3. Руководитель бакалаврской работы д.т.н., профессор каф. ВТ ПГУ Гурин Е.И.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение, предназначенное для передачи и приёма имеющих ценность данных по протоколу UDP. Необходимо предусмотреть использование ассиметричного шифрования с открытым ключом. Приложение должно иметь удобный пользовательский интерфейс, должно задавать параметры работы (IP-адрес и порт получателя, собственный интерфейс и порт), должно выводить данные, принятые приложением в текстовом виде с указанием адреса отправителя.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области, выбрать архитектуру приложения, определить язык программирования, разработать протокол взаимодействия, выбрать алгоритм ассиметричного шифрования, определить способы ввода текстовых сообщений пользователем.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области, выбор языка программирования	10	30.04.2020 – 04.05.2020	
2.	Выбор архитектуры приложения, разработка протокола взаимодействия	10	04.05.2020 – 07.05.2020	
3.	Разработка пользовательского интерфейса, определение способа ввода текстовых сообщение пользователем	20	07.05.2020 – 14.05.2020	
4.	Выбор алгоритма шифрования и поиск подходящих библиотек	10	14.05.2020 – 18.05.2020	
5.	Разработка приложения	20	18.05.2020 – 25.05.2020	
6.	Отладка и тестирование	10	25.05.2020 – 28.05.2020	
7.	Оформление выпускной работы	20	28.05.2020 – 03.06.2020	

Дата выдачи задания "30" апреля 2020 г.

Руководитель бакалаврской работы _____ Гурин Е.И.
(подпись, дата)

Задание к исполнению принял студент _____ Грушевский А.А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Бакалаврская работа содержит 93 листа, 24 рисунка, 2 таблицы, 16 источников, 2 приложения.

UDP, QT, ШИФРОВАНИЕ, ПРИЁМНИК, ОТПРАВИТЕЛЬ, КРОССПЛАТФОРМЕННОЕ ПРИЛОЖЕНИЕ, RSA, ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ.

Цель работы — разработать приложение, предназначенное для передачи и приёма имеющих ценность как для отправителя, так и для получателя, сообщений по протоколу UDP.

Объектом разработки является приложение, состоящее из программы с удобным для пользователя графическим интерфейсом, которая способна принимать входящие сообщения, выполнять их обработку и отправлять по заданному адресу, используя протокол транспортного уровня UDP.

В результате проведённой работы реализован проект, полностью удовлетворяющий поставленной задаче, а именно, создано приложение, осуществляющее приём и передачу конфиденциальных сообщений по протоколу UDP.

Данное приложение рекомендуется использовать для обмена конфиденциальными данными в локальной вычислительной сети между различными рабочими станциями. В случае перехвата передаваемой по открытым каналам дейтаграммы злоумышленником, ему не удастся извлечь передаваемое сообщение поскольку оно зашифровано, и не удастся его успешно подменить из-за использования алгоритма цифровой подписи, благодаря которой принимающая сообщение сторона узнает о подмене.

Перечень принятых сокращений

Сетевая модель OSI (The Open Systems Interconnection model) – сетевая модель стека сетевых протоколов;

TCP (Transmission Control Protocol) – протокол управления передачей;

IP (Internet Protocol) – межсетевой протокол;

UDP (User Datagram Protocol) – протокол пользовательских дейтаграмм;

DSA (Digital Signature Algorithm) – алгоритм цифровой подписи;

ЭЦП – электронная цифровая подпись;

ГОСТ – региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества Независимых Государств;

RSA (Rivest, Shamir, Adleman) – аббревиатура от фамилий;

PGP (Pretty Good Privacy) – очень надежная криптографическая программа, которая позволяет пользователям анонимно обмениваться информацией в электронном виде;

API (Application Programming Interface) – программный интерфейс приложения;

GUI (Graphical user interface) – графический пользовательский интерфейс;

ООП – объектно-ориентированное программирование;

IDE (Integrated Development Environment) – интегрированная среда разработки;

ЛВС – локальная вычислительная сеть.

Содержание

РЕФЕРАТ....4
Перечень принятых сокращений....5
Содержание....6
Введение....8
1. Анализ предметной области....10
1.1. Анализ протоколов транспортного уровня....10
1.1.1. Протокол TCP....11
1.1.2. Протокол UDP....12
1.2. Анализ методов шифрования данных....13
1.2.1. Симметричное шифрование....13
1.2.2. Асимметричное шифрование....14
1.3. Виды асимметричных шифров....18
1.3.1. Алгоритм цифровой подписи DSA....18
1.3.2. Схема Эль-Гамаля....18
1.3.3. ГОСТ Р 34.10-2018....20
1.3.4. Асимметричный шифр RSA....22
1.4. Выбор платформы для разработки и языка программирования...25
2. Реализация....26
2.1. Описание работы приложения....26
2.2. Описание библиотеки Qt-Secret....32
2.3. Описание функций....33
3. Инструкция пользователя....38
3.1. Главное окно программы....38
3.2. Меню приложения....40
3.3. Пункт меню «Настройки»....42
3.4. Пункт меню «О программе»....42
3.5. Обработка исключительных случаев....43
4. Безопасность жизнедеятельности....44

Заключение.....45
Список используемых источников.....46
Приложение А. Листинги программы.....48
Файл «mainwindow.h».....48
Файл «mainwindow.cpp».....56
Файл «settings.h».....76
Файл «settings.cpp».....78
Файл «about.h».....81
Файл «about.cpp».....82
Файл «main.cpp».....83
Приложение В. Презентация.....85

Введение

Пользователями информации сейчас являются миллионы людей, которым для её обмена доступны самые различные варианты технологических устройств. В настоящее время круг людей, которые занимаются обработкой информации, вырос до фантастических размеров, а скорость обмена информации стала необычайно высока. На наших глазах устанавливается информационное общество, в котором акцент внимания и значимости смещается с традиционных видов ресурсов (финансовые, материальные, энергетические и пр.) на информационные, которые хоть всегда и существовали, но обычно не рассматривались как что-то значимое [1 - 3].

Исходя из этого одной из важнейших проблем современности является проблема защиты информации. Несколько десятков лет назад данная проблема могла быть эффективно решена с помощью организационных мер и отдельных программно-аппаратных средств разграничения доступа. Однако с появлением мощных персональных компьютеров и развитием информационных сетей проблема защиты информации существенно обострилась. В этом деле проблемы могут принимать разный характер. Данные могут перехватываться, тогда целостность информации сохраняется, но конфиденциальность уже нарушается. В случае модификации данных источник сообщения изменяется или полностью заменяется другим и отсылается получателю. Так же проблема замена авторства информации может иметь очень серьёзные последствия. Одной из важных проблем является социально-этическая. Всё больше применяются методы криптографический защиты данных, что является противоречием между желаниями пользователей и специальных служб. Одни хотят защитить свою информацию, а другие хотят иметь возможность доступа к ней с целью пресечения незаконных действий [4, 5].

Наряду с проблемами защиты информации появлялись и пути их решений. К организационным мерам по защите данных можно отнести: ограничение физического доступа в помещения, в которых осуществляются различные

операции с важной информацией, использование сейфов для её хранения, допуск к ней только проверенных должностных лиц. Для передачи конфиденциальной информации в сетях используются криптографические коды.

В данной работе будет рассмотрена реализация приложения, которое обеспечивает доставку и приём текстовых сообщений, имеющих ценность как для его отправителя, так и для стороны получателя. Данные будут передаваться по открытым каналам связи, но содержимое сообщения при этом будет надёжно защищено благодаря использованию алгоритмов асимметричного шифрования с открытым ключом.

1. Анализ предметной области

В современной научно-технической литературе существует большое количество работ, посвящённых как передаче данных по сети, так и криптографии. Для каждой проблемы всегда имеется несколько путей её решения, из которых нужно выбрать самый оптимальный. Данный раздел содержит обзор протоколов транспортного уровня TCP и UDP: их главные положительные качества и недостатки. Этот раздел так же описывает методы шифрования данных (симметричный и асимметричный) и виды шифров (DSA, RSA и прочие). Кроме того, в каждом рассматриваемом блоке обосновывается выбор того или иного технологического решения выполнения поставленной задачи, который является наиболее подходящим для решения проблемы.

1.1. Анализ протоколов транспортного уровня

Транспортный уровень является четвёртым уровнем сетевой модели OSI, предназначенным для доставки данных. Здесь неважно, какие данные передаются, откуда и куда, то есть, он предоставляет сам механизм передачи [1].

Модель OSI можно сопоставить со стеком протоколов TCP/IP, который широко используется для описания сетей на базе TCP/IP. Другими словами, он представляет собой основу Интернета.

Обе модели можно сопоставить (Рисунок 1). Синим цветом – модель OSI, жёлтым – стек протоколов TCP/IP [2].

Рисунок 1 – Сравнение моделей OSI и TCP/IP.

Основное предназначение транспортного уровня – транспортировка сообщений между приложениями узла источника и узла назначения. Приложение узла источника формирует сообщение и передаёт его на приложение узла назначения независимо от маршрута (независимо от протоколов сетевого уровня) и от среды передачи (независимо от протоколов канального и физического уровней). Транспортный уровень реализует управление потоком информации от источника до устройства назначения. Протоколы данного уровня предназначаются для взаимодействия типа точка-точка.

Транспортный уровень включает в себя два основных протокола: TCP и UDP. Далее рассмотрим их поподробнее.

1.1.1. Протокол TCP

Transmission Control Protocol (TCP) – протокол управления передачей, один из основных протоколов передачи данных Интернета, предназначенный для управления передачей данных.

TCP предусматривает необходимость предварительного установления соединения, благодаря чему гарантирует уверенность в целостности получаемых данных. В случае потери данных или их искажения он выполняет повторный запрос. Помимо этого, протокол TCP отслеживает дублирование пакетов и в случае обнаружения уничтожает дубликаты пакетов. Поэтому он используется при передаче файлов, где потеря одного пакета может привести к искажению всего файла. Для обеспечения надёжности данные разбиваются на сегменты оптимальной длинны, которые приёмник собирает в правильном порядке. При пересылке сегмента использует таймер для ожидания подтверждения от принимающей стороны (квитанция). Если по истечении определенного времени квитанция не приходит, выполняется повторная передача сегмента. После получения сегмента данных принимающая сторона проверяет контрольную сумму сегмента [1].

Можно делать вывод, что главный плюс протокола TCP – обеспечение надёжной доставки информации без потерь. Из минусов можно выделить: отсутствие широковещательной рассылки, медленный «старт» (необходимость в установлении соединения), работает медленнее UDP.

1.1.2. Протокол UDP

User Datagram Protocol (UDP) – протокол пользовательских дейтаграмм, один из ключевых элементов набора сетевых протоколов для Интернета. Он имеет короткий служебный заголовок, формат которого включает в себя 4 поля: порты отправителя и получателя, длина (содержит длину всей дейтаграммы UDP), контрольная сумма (используется для корректной доставки дейтаграммы) (Таблица 1) [2].

Таблица 1 – Форма заголовка UDP.

Порт отправителя (16 бит)	Порт получателя (16 бит)
Длина (16 бит)	Контрольная сумма (16 бит)

Главное преимущество UDP в том, что протокол обеспечивает более высокую скорость работы по сравнению с TCP, так как у него нет накладных расходов на установку и на разрыв соединения. Ошибки в современных сетях происходят достаточно редко и сетевые приложения способны самостоятельно исправлять такие, редко возникающие ошибки. Так же UDP позволяет вести широковещательную рассылку (в рамках локальной сети). Область применения протокола — это системы, которые работают в режиме запрос-ответ и обмениваются между собой короткими сообщениями, так как максимальная длина сообщения может составлять 65528 байт.

Для реализации поставленной задачи используется протокол UDP транспортного уровня модели OSI и стека протоколов TCP/IP. Обуславливается это тем, что он обеспечивает наиболее быструю передачу данных по сравнению

с протоколом TCP, а также отправляет данные напрямую по заданным параметрам без установления соединения.

1.2. Анализ методов шифрования данных

Шифрование используется человечеством с того самого момента, как появилась первая секретная информация, т. е. такая, доступ к которой должен быть ограничен. Есть два основных типа шифрования: симметричное (или закрытый ключ) и асимметричное (открытый ключ).

1.2.1. Симметричное шифрование

Симметричное шифрование носит такое название из-за того, что в симметричных криптосистемах используется один и тот же ключ и для шифрования, и для расшифрования. Оно являлось единственным существовавшим способом сокрытия данных до изобретения схемы асимметричного шифрования.

Оставаясь в рамках симметричной системы (так она названа оттого, что для шифрования и дешифрования подходит один и тот же ключ), необходимо иметь надежный канал связи для передачи секретного ключа. Ключ алгоритма должен сохраняться в тайне обеими сторонами, которые должны выбрать алгоритм шифрования до начала обмена сообщениями. Без заранее обговорённых приёмником и передатчиком алгоритма и ключа информация будет составлять собой лишь набор символов, не имеющих смысла. К классическим примерам симметричных криптографических алгоритмов можно отнести следующие: простая перестановка (сообщение записывается в таблицу по столбцам, после считывается по строкам), одиночная перестановка по ключу (похож на предыдущий, но колонки переставляются по ключевому слову), двойная перестановка (зашифрованный текст шифруется повторно, но с другими длинами строк и столбцов), перестановка «Магический квадрат» (квадратные таблицы со вписанными в их клетки последовательными

натуральными числами от 1, которые дают в сумме по каждому столбцу, каждой строке и диагонали одно и то же число [3 - 5].

Рассмотренный метод шифрования в реализации данной работы не подходит, поскольку не рассматривается вариант предварительного обсуждения пользователями используемого ключа и алгоритма шифрования.

1.2.2. Асимметричное шифрование

Огромной проблемой криптографии прошлых веков являлась проблема передачи ключей. Ведь в современных реалиях человек не будет осуществлять поездку до физического местоположения сервера с почтой, чтобы подтвердить авторизацию. Решение проблемы пришло в 76 году прошлого столетия, когда Диффи и Хеллман предложили свой «метод экспоненциального обмена ключей». С этого момента началось развитие асимметричных криптосистем [3 - 5].

Асимметричный алгоритм предполагает под собой наличие двух ключей – публичного и приватного. Сообщение шифруется публичным ключом, а расшифровывается приватным. Именно такую концепцию и сформулировал Диффи. Суть алгоритма в том, что принимающая сторона перед приёмкой сообщения генерирует пару ключей на основе модульной арифметики. Отправитель перед отправкой получает публичный ключ и шифрует им сообщение, после чего данное сообщение можно будет расшифровать только приватным ключом, который хранится в секрете на стороне приёмника, как показано на рисунке 2 [6, 7].

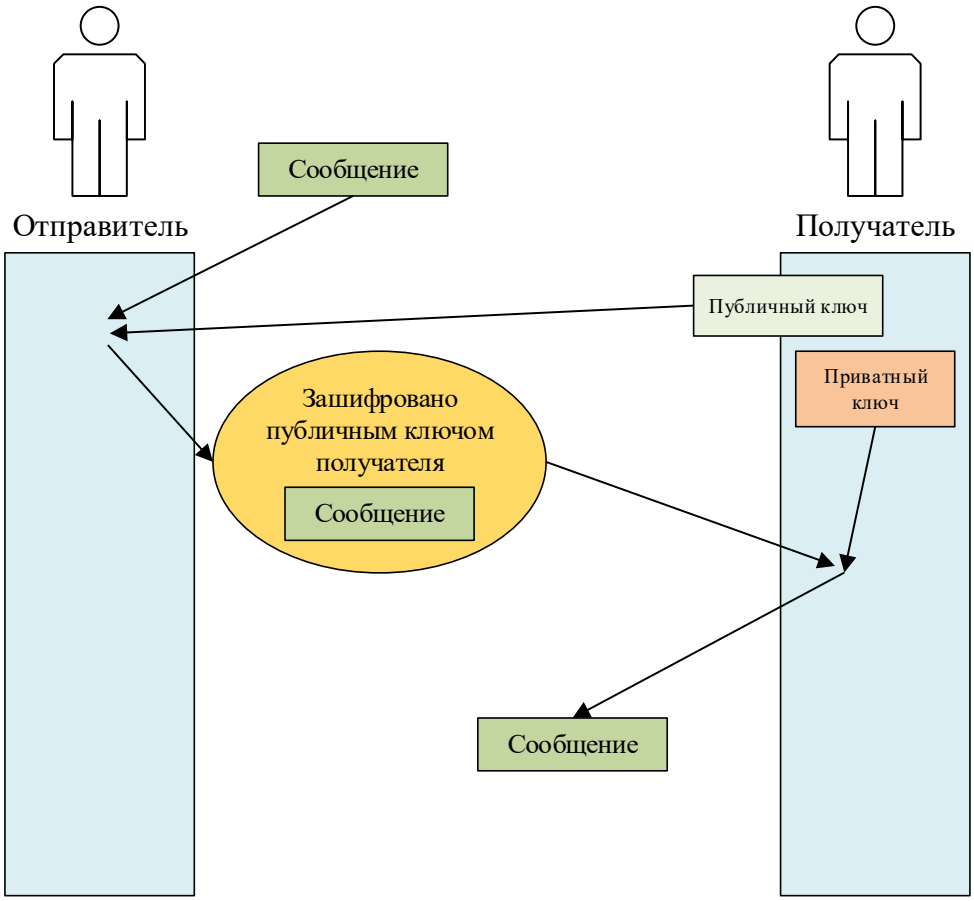


Рисунок 2 - Схема работы асимметричного алгоритма.

Однако у представленного алгоритма есть изъян. Проблема заключается в том, что получатель не может быть уверен в том, кто на самом деле является отправителем сообщения, т.к. публичным ключом может без проблем завладеть злоумышленник и представиться чужим именем.

Данная проблема решается использованием цифровой подписи. ЭЦП (электронная цифровая подпись) в самом примитивном случае можно считать результатом хеш-функции, то есть с высокой степенью вероятности её результат не повторяется для разных исходных данных. А кроме того, что её результат короче исходных данных, так ещё по нему нельзя восстановить исходную информацию.

Отправитель должен подтвердить авторство письма, зашифровав его своим приватным ключом, однако тогда его легко можно будет расшифровать кому угодно с использованием его же публичного ключа. Чтобы этого избежать,

нужно применить основное шифрование на подписанное сообщение (Рисунок 3).

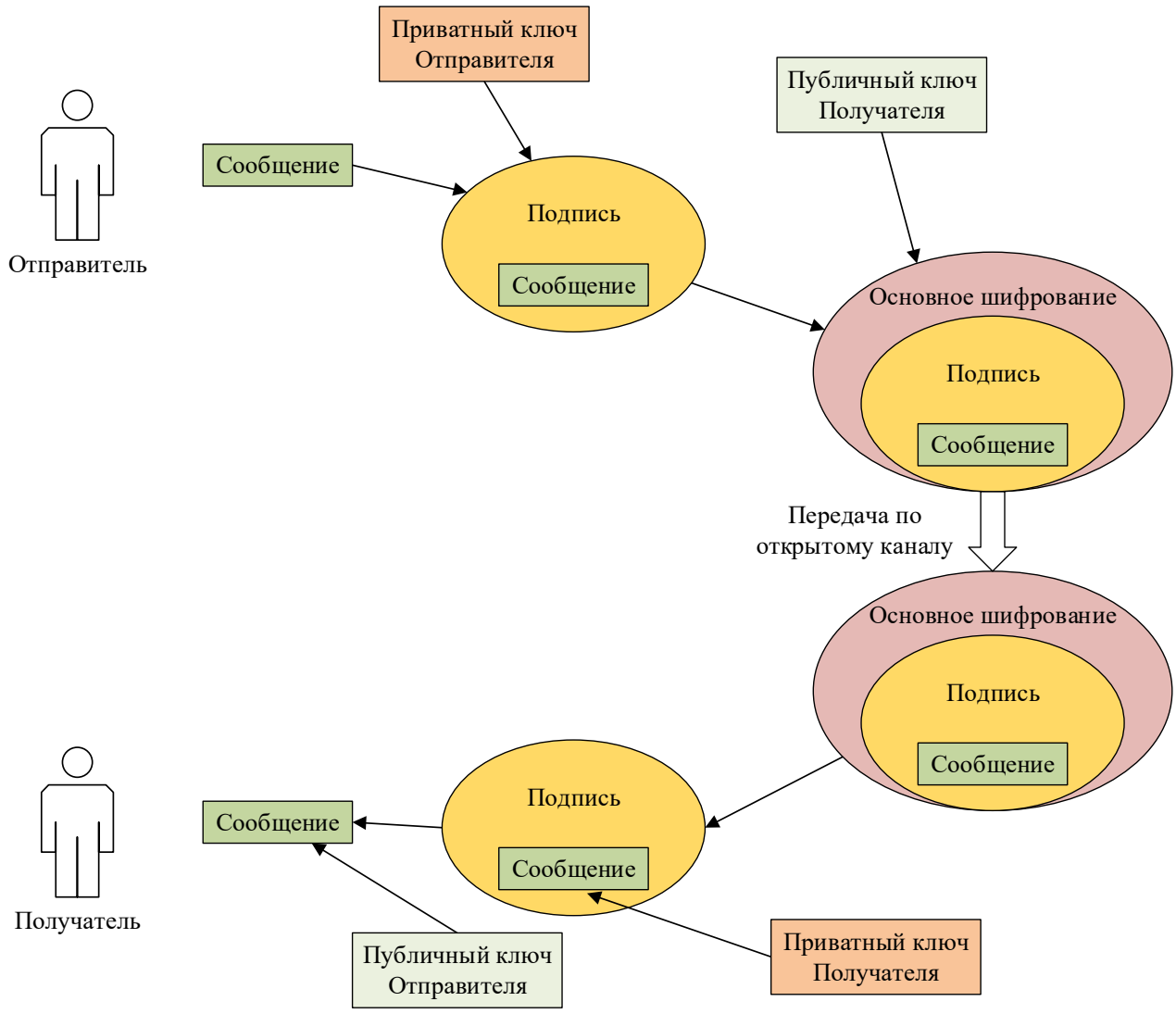


Рисунок 3 – Схема работы алгоритма цифровой подписи.

Суть алгоритма в следующем:

1. Отправитель шифрует сообщение своим приватным ключом (подписывает его).

2. Отправитель шифрует подписанное сообщение публичным ключом получателя, который он получил заранее.

3. Отправитель посылает сообщение.

4. Получатель принимает и расшифровывает шифротекст своим приватным ключом.

5. Получатель проверяет подписанное сообщение публичным ключом отправителя.

Такой подход и называется цифровой подписью, которая выполняет две основные функции – аутентификация отправителя и подтверждение целостности документа. То есть, если текст расшифровался публичным ключом отправителя, то автор действительно тот, за кого он себя выдаёт, а также документ пришёл в неизменном виде.

Впрочем, для того чтобы удостоверить отправителя, не обязательно шифровать весь текст целиком. Можно зашифровать приватным ключом хеш от исходного текста и полученную строку отправлять приёмнику вместе с текстом (Рисунок 4). Получателю лишь останется получить хеш сообщения и сравнить его с подписью (Рисунок 5) [6, 7].

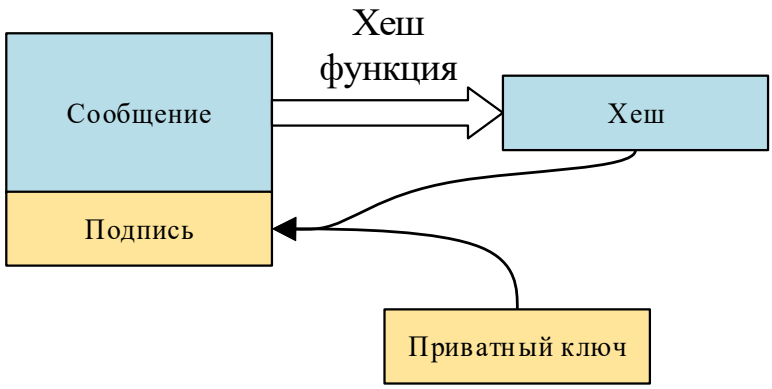


Рисунок 4 - Генерация подписи.

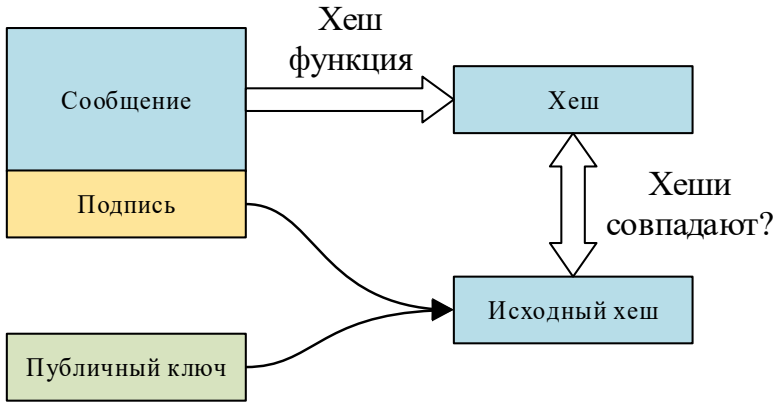


Рисунок 5 – Проверка подписи.

В данной работе для шифрования сообщения была использована цифровая подпись, поскольку она представляет из себя наиболее современное и технологичное решение по защите передаваемых по открытым каналам данных.

1.3. Виды асимметричных шифров

Так же существует множество видов асимметричных шифров, рассмотрим некоторые из них поподробнее.

1.3.1. Алгоритм цифровой подписи DSA

Digital Signature Algorithm (DSA) – алгоритм цифровой подписи, криптографический алгоритм с использованием открытого ключа для создания электронной подписи, но не для шифрования. Подпись создаётся секретно, но может быть проверена публично. Иначе говоря, только один субъект может создать подпись сообщения, но любой может проверить её корректность.

Этот алгоритм предложен и запатентован в августе 1991 года Национальным институтом стандартов и технологий (США) и основан на вычислительной сложности взятия логарифмов в конечных полях [3, 4, 5]. Примерная иллюстрация работы DSA приводилась ранее (Рисунок 4, Рисунок 5).

1.3.2. Схема Эль-Гамаля

Elgamal – криптосистема с открытым ключом, которая основана на трудности вычисления дискретных логарифмов в конечном поле. Она была предложена Тахером Эль-Гамалем в 1984 году, он усовершенствовал систему Диффи-Хеллмана и получил два алгоритма, которые использовались для шифрования и обеспечения аутентификации. Лежит в основе бывших стандартов ЭЦП в США (Алгоритм цифровой подписи) и России (ГОСТ Р 34.10-94). Работа в режиме шифрования представлена ниже. Шифрование по схеме Эль-Гамаля не следует путать с алгоритмом цифровой подписи по схеме Эль-Гамаля [8].

Наглядно схему шифрования по Эль-Гамалю можно рассмотреть на иллюстрации (Рисунок 6).

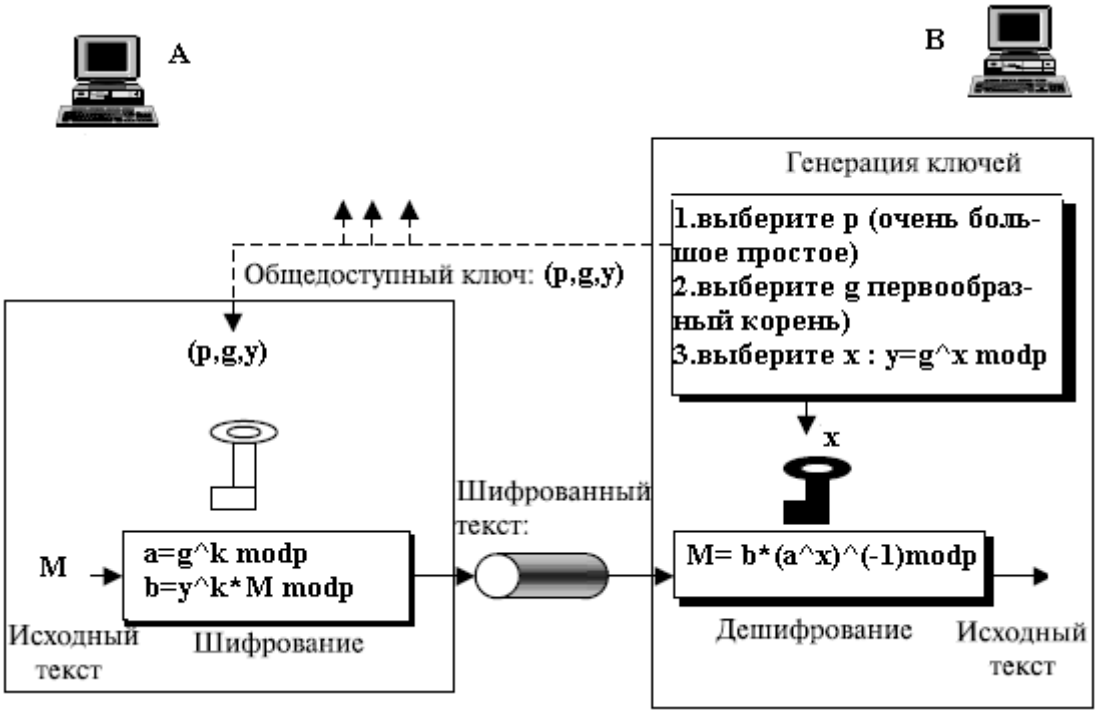


Рисунок 6 – Шифрование по схеме Эль-Гамаля.

Схему цифровой подписи по Эль-Гамалю можно рассмотреть на иллюстрации (Рисунок 7).

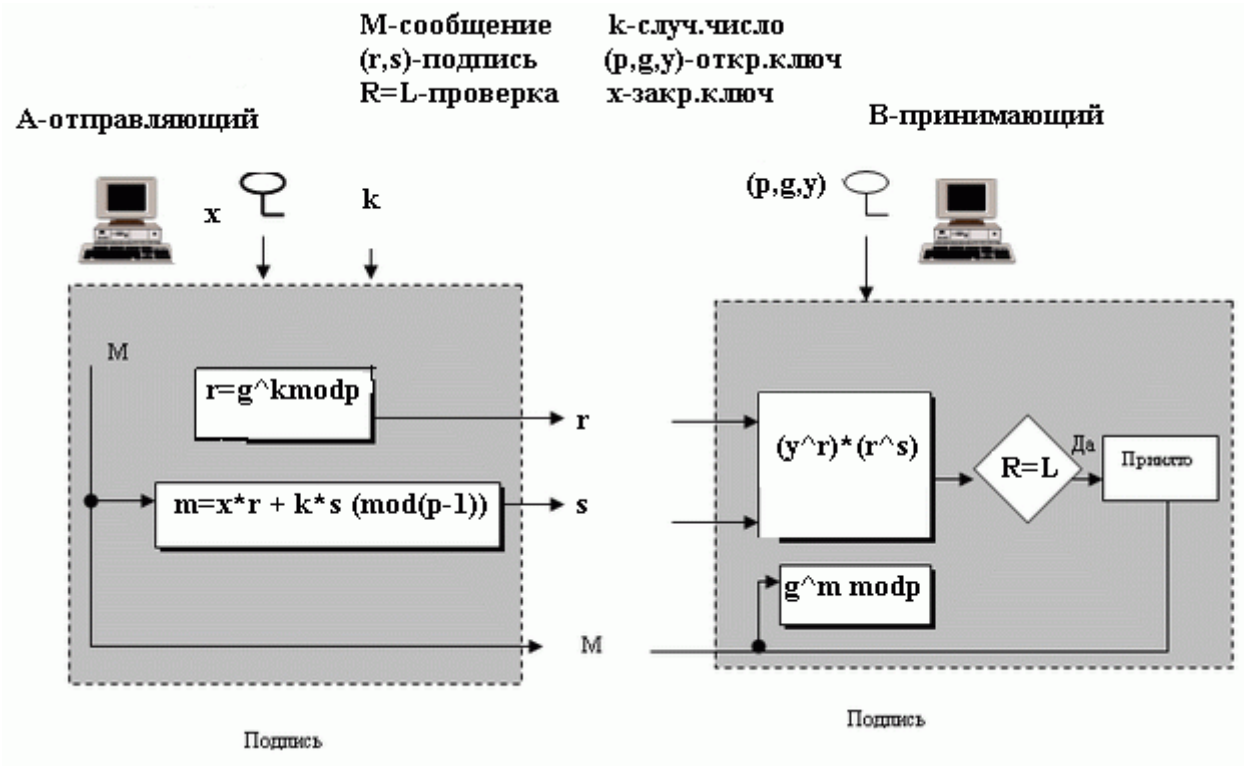


Рисунок 7 – Цифровая подпись по схеме Эль-Гамаля.

1.3.3. ГОСТ Р 34.10-2018

«ГОСТ 34.10-2018. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи» – действующий межгосударственный криптографический стандарт, описывающий алгоритмы формирования и проверки электронной цифровой подписи реализуемой с использованием операций в группе точек эллиптической кривой, определенной над конечным простым полем [8].

Стандарт разработан на основе национального стандарта Российской Федерации ГОСТ Р 34.10-2012 и введён в действие недавно, с 1 июня 2019 года.

Ниже наглядно представлены блок-схемы формирования и проверки цифровой подписи (Рисунок 8, Рисунок 9).

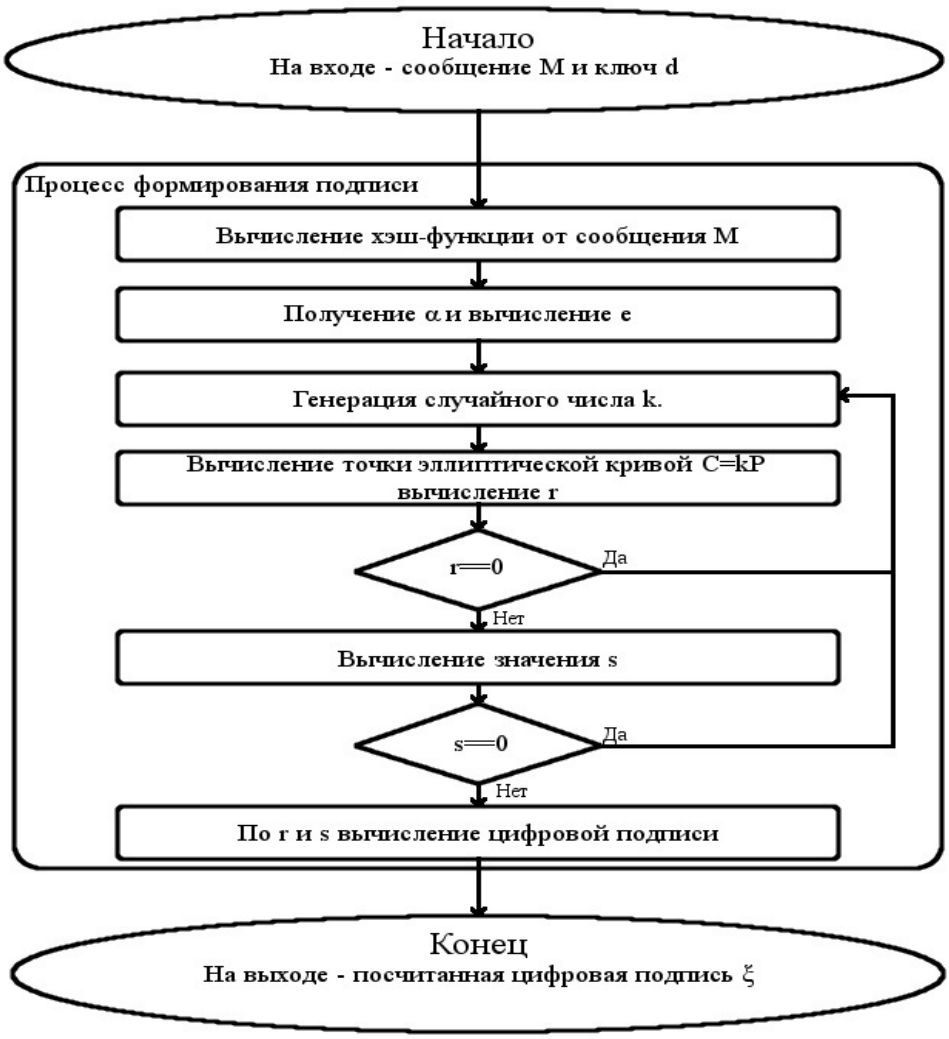


Рисунок 8 – Алгоритм формирования подписи ГОСТ 34.10-2018.

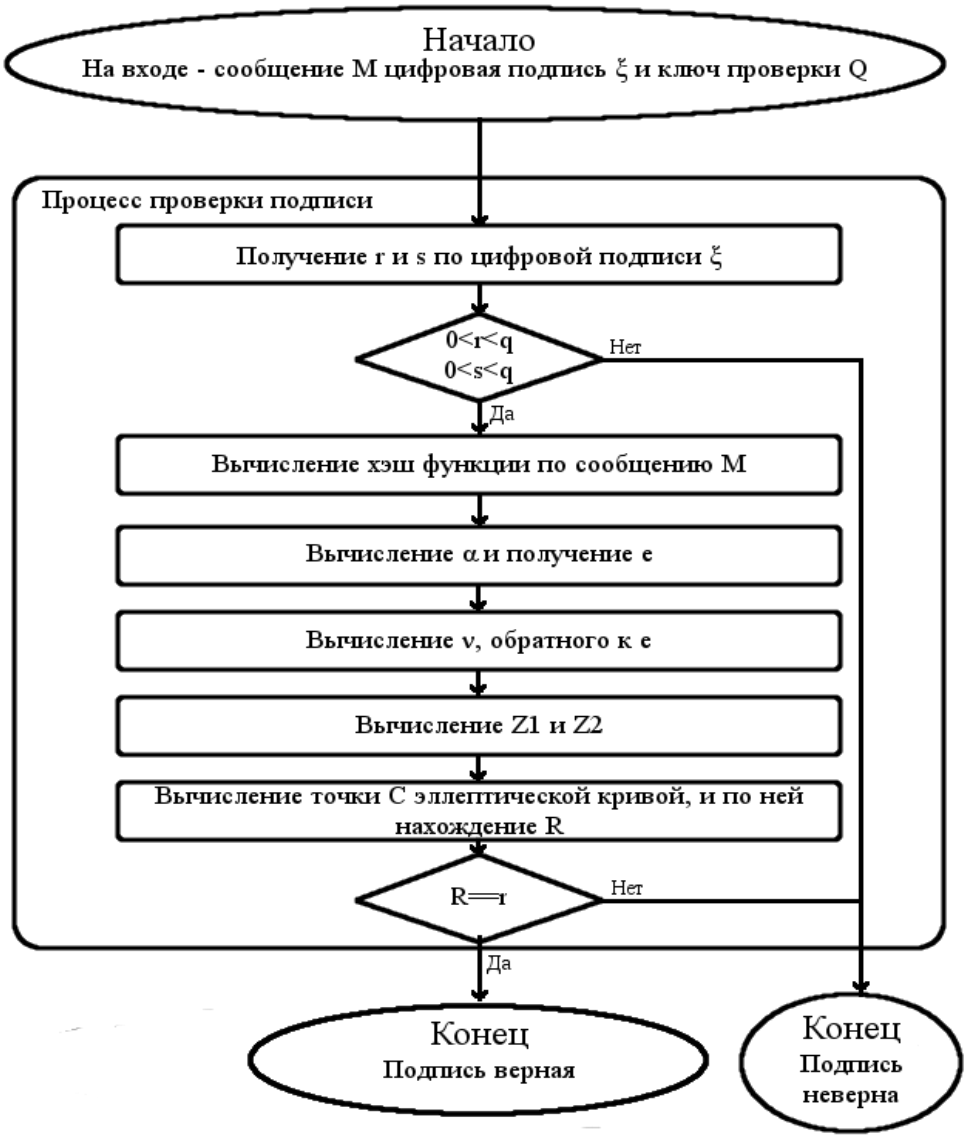


Рисунок 9 – Алгоритм проверки подписи ГОСТ 34.10-2018.

1.3.4. Асимметричный шифр RSA

RSA (аббревиатура от фамилий Rivest, Shamir, Adleman) – криптографический алгоритм с открытым ключом, который основывается на вычислительной сложности задачи факторизации больших целых чисел. Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи. Алгоритм используется в большом числе криптографических приложений [3, 4, 5, 8].

Главное достоинство алгоритма RSA – имея открытый ключ и зная алгоритм шифрования невозможно повторить закодированное сообщение, на

базе алгоритма RSA работает программа шифрования PGP, реализуются хэш-функции (электронно-цифровая подпись).

Открытый и закрытый ключи каждого из участников обмена данными в криптосистеме RSA образуют «согласованную пару» в том смысле, что они являются взаимно обратными.

Алгоритм создания публичного и приватного ключей:

1) Выбираются два различных случайных простых числа p и q заданного размера.

2) Вычисляется их произведение $n = p \cdot q$, которое называется модулем.

3) Вычисляется значение функции Эйлера от числа n:
$\varphi(n) = (p - 1) \cdot (q - 1)$.

4) Выбирается целое число e $(1 < e < \varphi(n))$, взаимно простое со значением функции $\varphi(n)$. Число e называется открытой экспонентой.

5) Вычисляется число d, мультипликативно обратное числу e по модулю $\varphi(n)$, то есть число, удовлетворяющее сравнению:
$d \cdot e \equiv 1 \pmod{\varphi(n)}$. Число d называется секретной экспонентой, обычно оно вычисляется по расширенному алгоритму Евклида.

6) Пара (e, n) – открытый ключ RSA.

7) Пара (d, n) – закрытый ключ RSA.

Алгоритм шифрования (сообщениями являются целые числа в интервале от 0 до $n - 1$, т.е. $m \in Z_n$:

1) Получить открытый ключ адресата (e, n).

2) Взять сообщение m.

3) Зашифровать сообщение по формуле: $c = E(m) = m^e \bmod n$.

Алгоритм расшифрования:

1) Принять зашифрованный текс *c*.
2) Взять свой закрытый ключ *(d, n)*.
3) Расшифровать сообщение по формуле: $m = D(c) = c^d \bmod n$.

Более наглядно рассмотреть алгоритм шифрования и расшифрования RSA можно по иллюстрации (Рисунок 10).

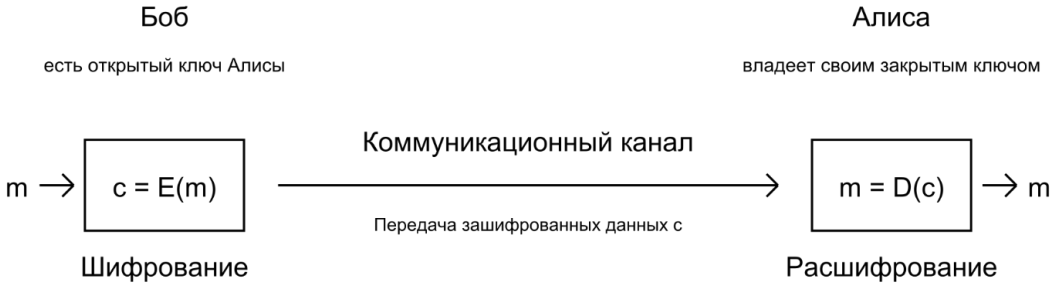


Рисунок 10 – Алгоритм шифрования и расшифрования RSA.

Как уже описывалось ранее, система RSA может быть использована как для шифрования, так и для цифровой подписи.

Алгоритм цифровой подписи RSA:

1) Взять сообщение *m*.
2) Создать цифровую подпись *s* с помощью своего секретного ключа *{d, n}* по формуле $s = S_A(m) = m^d \bmod n$.
3) Передать пару *{m, s}*, состоящую из сообщения и подписи.

Алгоритм проверки цифровой подписи RSA:

1) Принять пару *{m, s}*.
2) Взять открытый ключ отправителя *{e, n}*.
3) Вычислить прообраз сообщения из подписи: $m' = P_A(s) = s^e \bmod n$.
4) Если *m* и *m'* равны, то подпись верна.

Более наглядно рассмотреть алгоритм формирования и проверки цифровой подписи RSA можно по иллюстрации (Рисунок 11).

Рисунок 11 – Алгоритм формирования и проверки цифровой подписи RSA.

Для выполнения поставленной задачи будет использован именно алгоритм RSA, поскольку он является одним их самых распространённых и надёжных методов асимметричного шифрования.

1.4. Выбор платформы для разработки и языка программирования

Учитывая требования, необходимые для реализации данного приложения, для его написания был выбран Qt – кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++ [9, 10].

Qt зародился из желания разработчиков C++ иметь удобный инструментарий разработки GUI (графических интерфейсов программ). Наряду с этим, он включает в себя классы для работы с сетью, мультимедиа и множество других немаловажных и востребованных вещей. Первый релиз библиотеки вышел в 1995 году, с тех пор Qt лишь расширялся за пределы разработки только интерфейсов. Сейчас его можно рассматривать, как полноценный инструментарий классов на все случаи жизни, что отлично подойдёт для реализации поставленной задачи. Разработка пользовательского интерфейса программы не будет являться сложной задачей и появится возможность сконцентрировать своё внимание на других вещах.

Язык программирования C++ содержит средства для создания программ практически любого назначения, от низкоуровневых до сложных программных комплексов совершенно разного назначения. Он поддерживает возможность использования разных технологий и стилей программирования, включая ООП,

обобщённое и метапрограммирование. Использование Qt наделяет разработчика возможностью удобной настройки межпроцессорного взаимодействия, предоставляет ему широкий спектр возможностей (контейнеры, сроки, алгоритмы, GUI, и прочее). Так же фреймворк имеет хорошую документацию и неплохую IDE, переносимость на уровне исходного кода, низкий порог вхождения. Но что не мало важно – Qt активно развивается и прогрессирует.

2. Реализация

В данной главе описана реализация заданного приложения. Особое внимание уделяется выделению о описанию различных сущностей, и их взаимодействию между собой. При этом освещены основные особенности реализации, связанные с программированием на языке C++ платформы Qt. Целью данного раздела является объяснение принципа работы и внутреннее устройство проектируемой программы.

2.1. Описание работы приложения

При старте приложения автоматически выбирается первый подходящий сетевой интерфейс и свободный порт текущей машины, затем происходит связывание с ними сокета, после этого сразу можно принимать входящие сообщения благодаря последующей установке функции приёма дейтаграмм.

Программа после нажатия кнопки «Отправить» посылает введённый текст на другой экземпляр приложения в данной ЛВС по заданным в группе «Параметры получателя» IP-адресу и порту. В случае, если введённые параметры получателя будут некорректными или пользователь не введёт текст сообщения, программа выдаст соответствующее сообщение.

При получении сообщения, в поле принятых сообщений отобразятся параметры отправителя, время получение сообщения и сам текст. Так же в поле принятых сообщений можно отображать собственные отправленные сообщения, поставив соответствующую галочку в пункте меню «Сообщения».

Непосредственно перед отправкой сообщения отправитель посылает запрос приёмнику на получение открытого ключа, передавая ему вместе с тем свой публичный ключ. Затем, когда отправитель получит публичный ключ получателя, он может подписать и зашифровать сообщение, в последствии отправив его. Получатель уже примет сообщение и корректно расшифрует, и проверит подпись полученным публичным ключом.

Сменить собственные локальные адрес и порт пользователь может в пункте «Настройки», выбрав один из доступных адресов и указав нужный порт. После сохранения настроек сокет произойдёт повторная привязка сокета к новым параметрам.

Для работы с содержимым текстовых файлов используется класс QFile. При извлечении содержимого файла (нажатии кнопки «Прикрепить…» и выбора файла) оно считывается в поток и помещается в поле для ввода сообщения. Для сохранения принятой информации необходимо выделить нужный текст, нажать кнопку «Сохранить…» и выбрать файл, в который запишется выбранный текст.

UML-диаграмма классов реализованной программы прилагается (Рисунок 12).

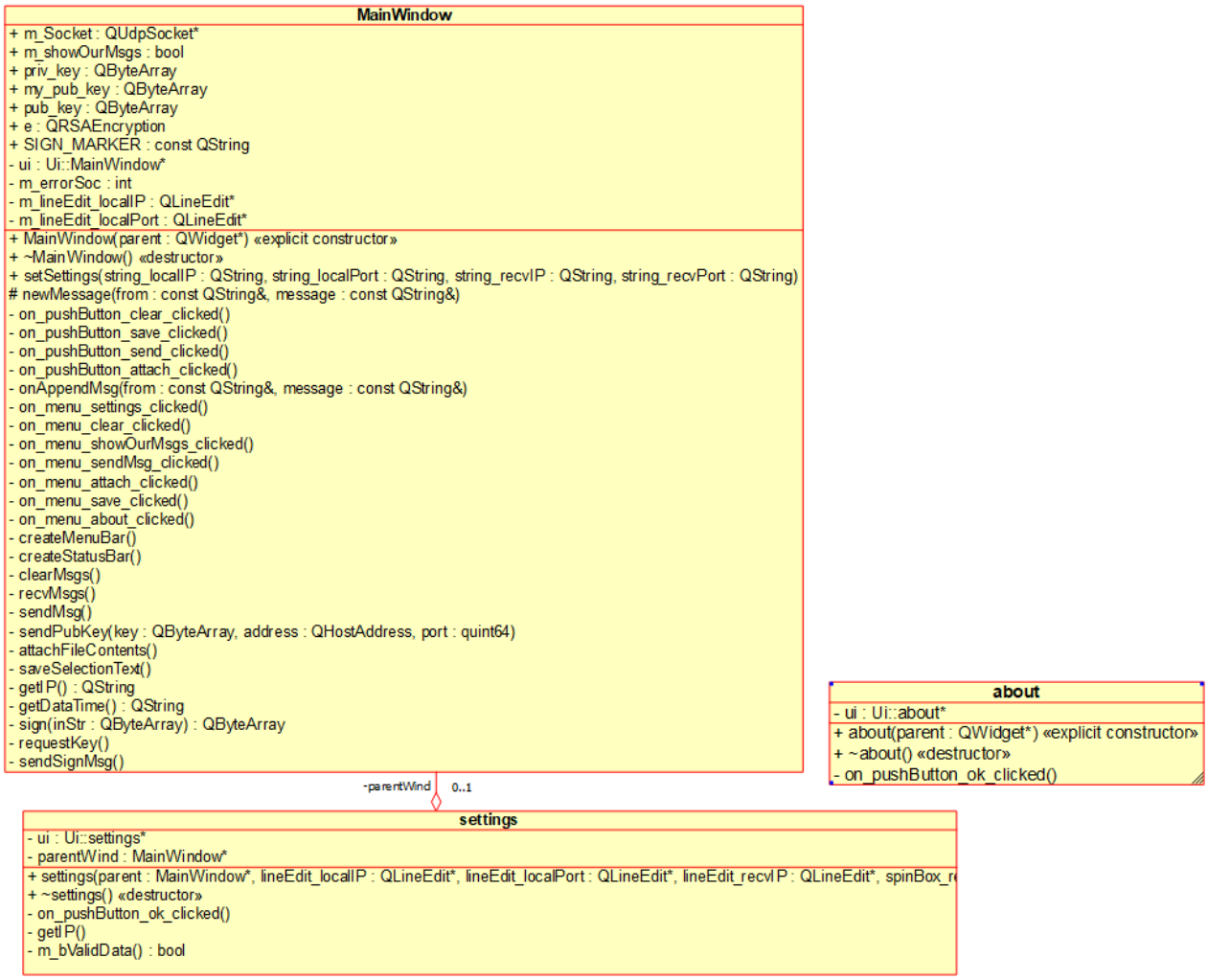


Рисунок 12 – UML-диаграмма классов.

На основе описания работы приложения можно оставить UML-диаграмму последовательности (Рисунок 13). Она визуализирует действия, описанные выше.

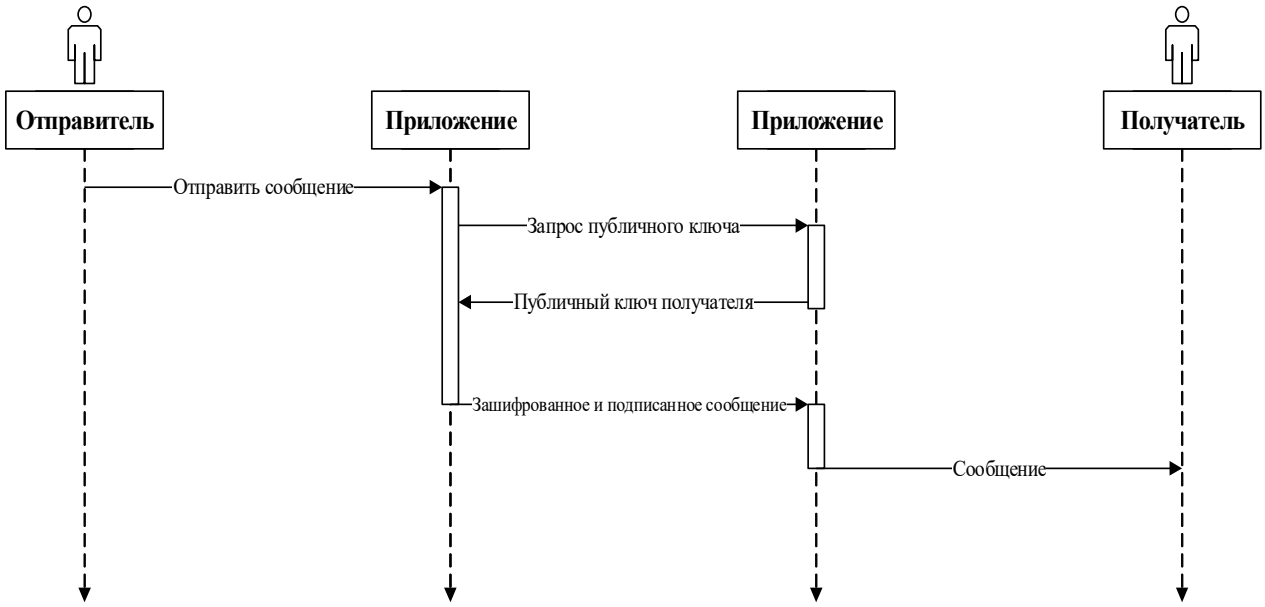


Рисунок 13 – UML-диаграмма последовательности.

UML-диаграмма вариантов использования так же представлена (Рисунок 14). Она отражает возможности пользователя.

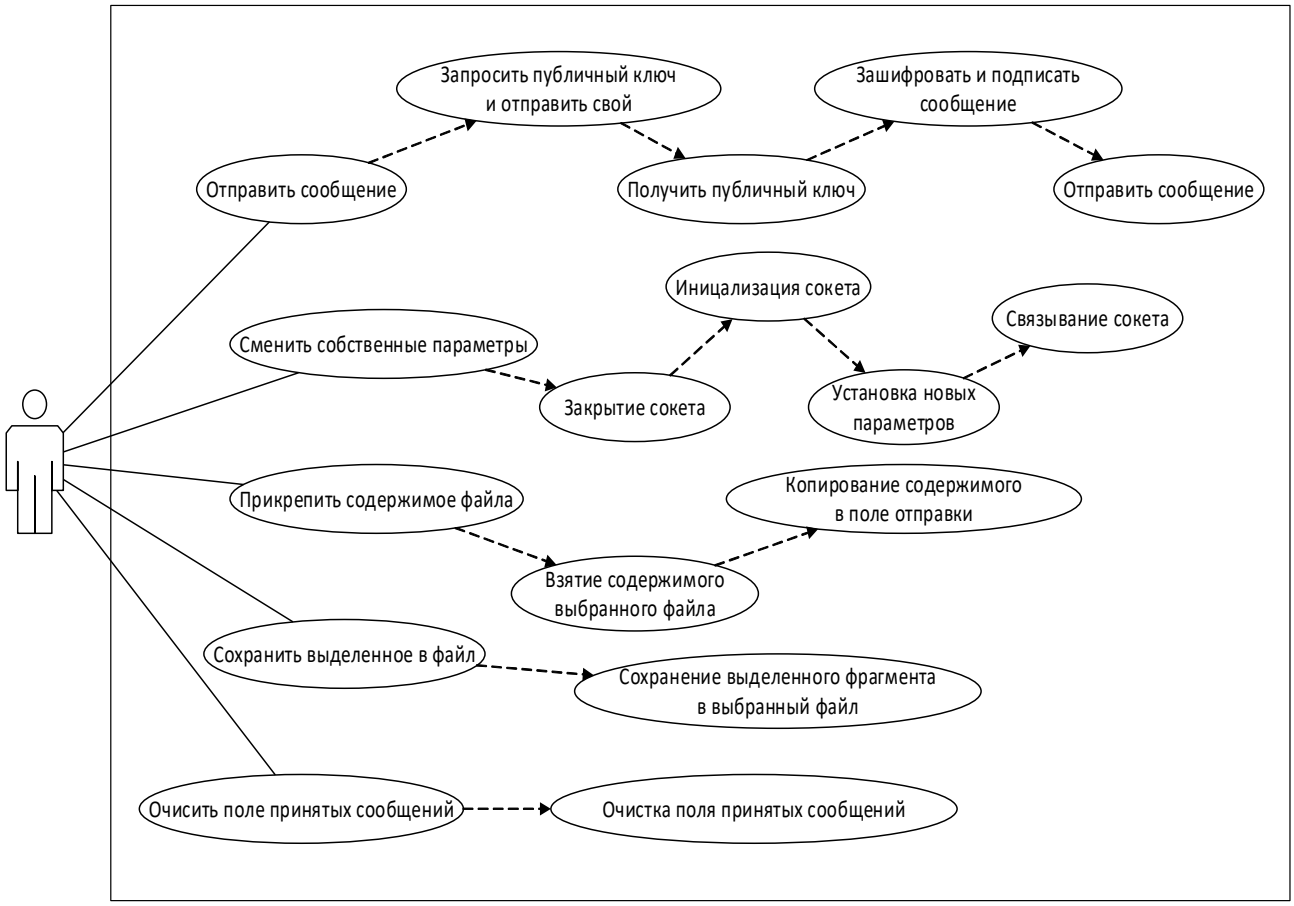


Рисунок 14 – UML-диаграмма вариантов использования.

Диаграмма деятельности программы тоже имеет место быть (Рисунок 15). Она отражает жизненный цикл программы.

Из диаграммы деятельности можно выделить следующие состояния программы:

1. Состояние ожидания входящей дейтаграммы.
2. Состояние ожидания отправки сообщения пользователем.
3. Состояние подписи и шифрации сообщения.
4. Состояние ожидания ключа.
5. Состояние дешифрации и проверки подписи.
6. Состояние отправки ключа.
7. Состояние отправки сообщения.
8. Состояние вывода сообщения.

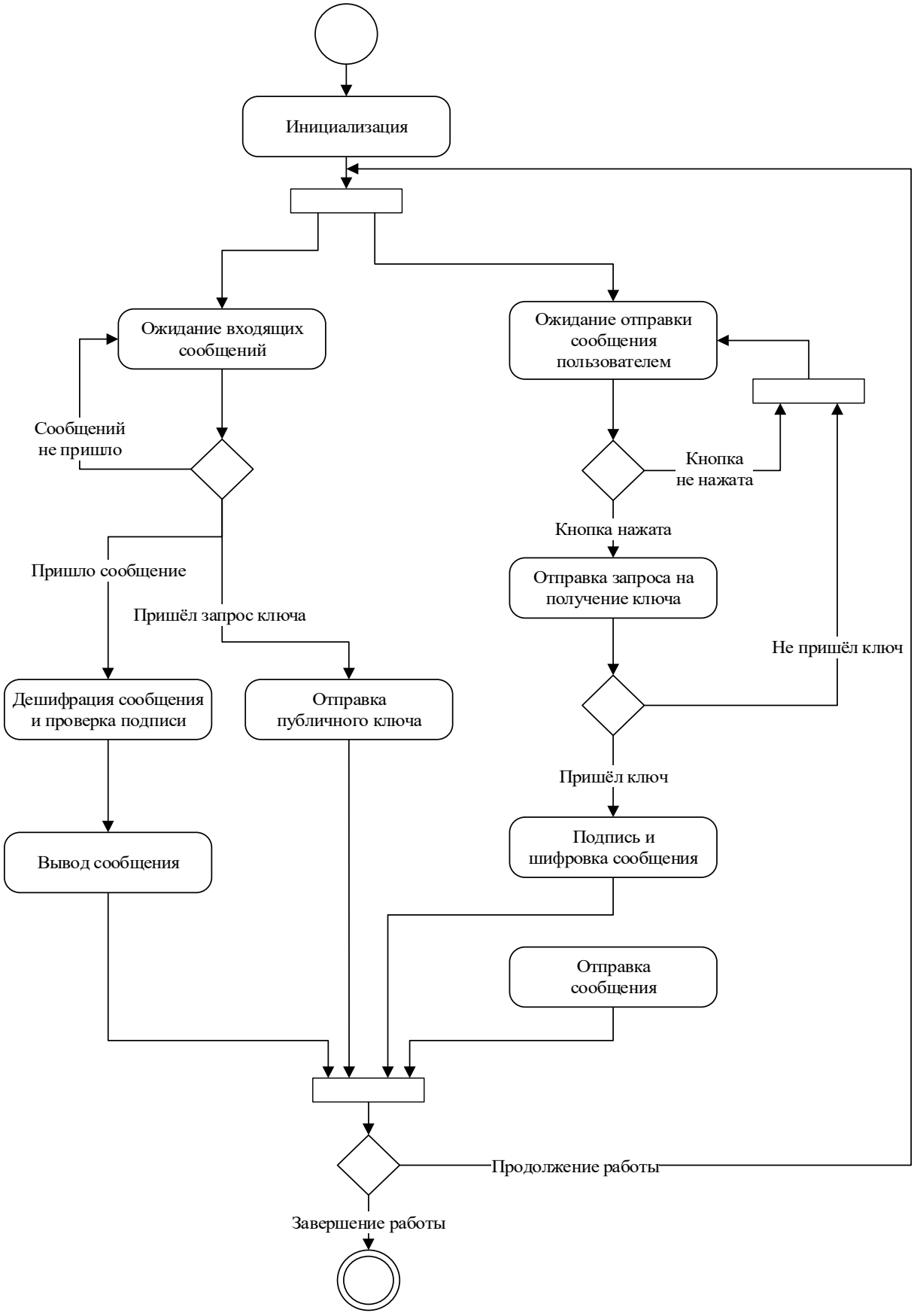


Рисунок 15 – UML-диаграмма деятельности.

2.2. Описание библиотеки Qt-Secret

Функции подписывания сообщения, его шифрования, расшифрования и проверки подписи в приложении реализованы с использованием библиотеки Qt-Secret. Она представляет из себя быструю библиотеку шифрования с поддержкой алгоритмов RSA и AES. Написана она группой QuasarApp на Qt/qmake, цель которой – обеспечить базовые возможности шифрования, которых как раз не хватает в базовом Qt [13].

Необходимый для решения задачи алгоритм RSA поддерживает несколько размеров – от 64 до 8192 бит. Поддерживается шифрование и дешифрование данных, подпись и проверка подписи сообщения.

Так же на старте программы средствами библиотеки генерируется пара ключей для шифрования. Для этого используется функция *generatePairKey()*, в первый параметр которой запишется публичный ключ, во второй – приватный, а третий параметр – режим шифрования).

Функция *signMessage()* подписывает сообщение, которое передаётся в неё первым параметром и возвращает в массиве байт с сообщением и хеш-функцией, которая является подписью. Вторым параметром указывается свой приватный ключ, а третьим – режим шифрования.

Функция *encode()* шифрует сообщение, которое передаётся первым параметром. Вторым параметром указывается публичный ключ получателя, а третьим – режим шифрования. Возвращает уже зашифрованный массив байт.

Для расшифровки сообщения принимающая сторона сначала использует функцию *decode()*, передавая в первый параметр зашифрованное сообщение, а во второй – свой приватный ключ, третьим параметром является режим шифрования. Возвращает функция сообщение вместе с подписью.

Для проверки подписи используется функция *checkSignMessage()*, которая возвращает истину в случае, если проверка проведена успешно. Первым параметров в неё передаётся сообщение с подписью (расшифрованный массив байт), вторым – публичный ключ отправителя, а третьим – режим шифрования.

Далее следует отрезать ненужные байты с подписью и получить исходное сообщение.

Исходя из описанного выше моно сделать вывод, что библиотека Qt-Secret отлично подходит для решения поставленной задачи и использование её в данной работе является абсолютно обоснованным.

2.3. Описание функций

В приложении реализован ряд функций, необходимый для решения поставленной задачи.

Публичная функция *setSettings()* необходима для уставноки в главном окне новых сетевых параметров после изменения их в окне настроек. Она содержит четыре параметра с новым локальным адресом и портом и с новым портом и адресом получателя. Так же в ней происходит закрытие сокета со старыми параметрами и инициализация сокета с новыми параметрами.

Приватный слот *onAppendMsg()* принимает параметрами данные отправителя и сообщение, которое он отправил, и помещает их в окно принятых сообщений. В этом же методе происходит разукрашивание частей сообщений в нужные цвета, операции с полосами прокрутки и переносы курсоров для отораженния последнего полученного сообщения.

При программированияя пользовательского графического интерфейса, когда изменяется один из виджетов, часто имеется необходимость, чтобы другие виджеты были об этом уведомлены. В Qt для реализации этого используются сигналы и слоты Сигнал выполняется в том случае, если просиходит определённое событие. Хоть виджеты в Qt и имеют множество предопределённых сигналов, но всегда можно унаследоваться от виджета и определить для них обсвтвенные сигналы. Слот – это функция, которая вызывается в ответ на определённый сигнал. Также виджеты имеют и множество предопределённых слотов, но добавление собственных слотов очень распространено.

В реализованной программе определён сигнал *newMessage()*, срабатывание которого вызывает функцию *onAppendMsg()*, рассмотренную выше. Связывание их происходит функцией *connect()*.

Остальные приватные слоты в приложении отвечают за обработку нажатия кнопок. К ним относятся:

- *on_pushButton_clear_clicked()* – обработка нажатия кнопки «Очистить»;
- *on_pushButton_save_clicked()* – обработка нажатия кнопки «Сохранить…»;
- *on_pushButton_send_clicked()* – обработка нажатия кнопки «Отправить»;
- *on_pushButton_attach_clicked()* – обработка нажатия кнопки «Прикрепить…».

Для создания строки меню используется функция *createMenuBar()*. Она необходима из-за невозможности создания панели меню в дизайнере форм. В ней создаются объекты класса *QAction* и к ним устанавливаются сигналы *&QAction::triggered* и соответствующие функции, которые необходимо вызвать в том или ином событии. К событиям относятся:

- *action_setting* – «Настройки»;
- *action_clear* – «Очистить поле с сообщениями»;
- *action_showOurMsgs* – «Отображать собственные сообщения»;
- *action_sendMsg* – «Отправить сообщение»;
- *action_attach* – «Прикрепить содержимое файла…»;
- *action_save* – «Сохранить выделенный фрагмент в файл…»;
- *action_about* – «О программе».

К вызываемым функциям относятся:

- *on_menu_settings_clicked()* – обработка нажатия пункта меню «Настройки» (создание и отображение окна настроек);

- *on_menu_clear_clicked()* – обработка нажатия подпункта «Очистить поле с сообщениями» (вызов функции очистки поля принятых сообщений);
- *on_menu_showOurMsgs_clicked()* – обработка нажатия подпункта «Отображать собственные сообщения» (если отображение было включено – отключение отображения, иначе – включение);
- *on_menu_sendMsg_clicked()* – обрабокта нажатия подпункта «Отправить собщение» (вызов функции отправки сообщения);
- *on_menu_attach_clicked()* – обработка нажатия подпункта «Прикрепить содержимое файла…» (вызов функции прикрепления содержимого файла);
- *on_menu_save_clicked()* – обработка нажатия подпункта «Сохранить выделенный фрагмент в файл…» (вызов функции сохранения содержимого файла);
- *on_menu_about_clicked()* – обработка подпункта «О программе» (создание и отображение окна «О программе»).

Для создания строки состояния используется функция *createStatusBar()*. Её необходимость также обуславливается невозможностью создания строки состояния в дизайнере форм. Она содержит два объекта класа *QLabel* и два объекта класса *QLineEdit*, которые для пользователя представляют из себя инфрмацию о собственном IP-адресе и порте, которая доступна только для чтения, но не для редактирования.

Функция *clearMsgs()* производит очистку поля принятых сообщений, занося в это поле пустую строку.

Функция *recvMsgs()* используется в роли слота, который вызывается в случае готовности слушающего сокета принимать входящие дейтаграммы. Данный слот устаналвивается при старте программы в конструкторе главного класса функцией *connect()*. Также *recvMsg()* переустанавливается последовательным вызовоы функций *disconnect()* и *connect()* в случае изменения сетевых параметров в окне настроек.

Функция *sendMsg()* используется для отправки сообщений. Сначала проверяется, заполнены ли все нужные поля, далее идёт проверка на размер сообщения. Если сообщение слишком длинное, то лишнее обрезается. После этого вызывается функция запроса публичного ключа получателя *requestKey()*. Получателю отправляется дейтаграмма, первый байт которой содержит код «0», который означает, что пришёл запрос на получение ключа, а оставшееся тело дейтаграммы – публичный ключ отправителя. Отправка дейтаграммы выполняется с помощью метода *writeDatagram()*.

Поподробнее теперь рассмотрим функцию приёма сообщений *recvMsg()*. В ней принимается дейтаграмма, извлекается сообщение и данные отправителя. Чтение ейтаграммы осуществляется с использованием метода *readDatagram()*. Далее проверяется тип сообщения. В приложении существует три типа сообщений:

- запрос на получение ключа (первый байт дейтаграммы – ‘0’) – помимо кода содержит в себе публичный ключ отправителя;
- пришёл публичный ключ (первый байт дейтаграммы – ‘1’) – дейтаграмма помимо кода содерит сам ключ;
- пришло сообщение (первый байт дейтаграммы – ‘2’) – дейтаграмма содержит в себе само сообщение.

В случае получения запроса на ключ, полученный ключ сохраняется и вызывается функция *sendPubKey()*, которая отправляет публичный ключ тому, кто его запросил.

В случае получения публичного ключа, в переменную публичного ключа получателя заносится полученный ключ и вызывается функция непосредственной отправки сообщения *sendSignMsg()*. В ней происходит шифрация и подпись отправляемого сообщения функцией *sign()* и последующая отправка уже зашифрованного и подписаннного текста.

В случае, если функция обнаружила дейтаграмму с сообщением, происходит дешифрация сообщения и проверка электронной цифровой

подписи. После успешного прохождения проверки формируется строка с параметрами отправителя и вызывается сигнал *newMessage()*, который активирует вызов функции *onAppendMessage()*.

Типы передаваемых дейтаграмм наглядно представлены ниже (Таблица 2).

Таблица 2 – Типы дейтаграмм.

Дейтаграмма	
Тип дейтаграммы (первый байт)	Тело дейтаграммы
µ0¶	Публичный ключ отправителя
µ1¶	Публичный ключ получателя
µ2¶	Сообщение

Для того, чтобы поместить содержимое некоторого файла в поле для отправки сообщений используется функция *attachFileContents()*. В случае её вызова перед пользователем появляется диалоговое окно с ожиданием выбора файла. После чего содержимое выбранного файла считается в поток и сохранится в строковую переменную, а затем отобразится в поле для отправки.

Для сохранения выделенного фрагмента в файл используется функция *saveSelectionText()*. В ней создаётся объект класса курсора, сохраняется выделенное методом *selectedText()*, заменяются символы переноса. Далее полученный текст преобразовывается в массив байт. Если массив байт не пустой, то вызывается диалоговое окно в котором пользователю необходимо выбрать файл, в который перезапишется полученный фрагмент.

Для отпределения собственного IP-адресов используется функция *getIP()*, которая перебирает список адресов текущей машины и возвращает в строковом формате первый подходящий по фильтру протокола IPv4.

Текущие дата и время в программе определяются функцией *getDataTime()*, в которой возвращается системное время локальной машины.

3. Инструкция пользователя

3.1. Главное окно программы

После запуска приложения перед пользователем появляется главное окно (Рисунок 16).

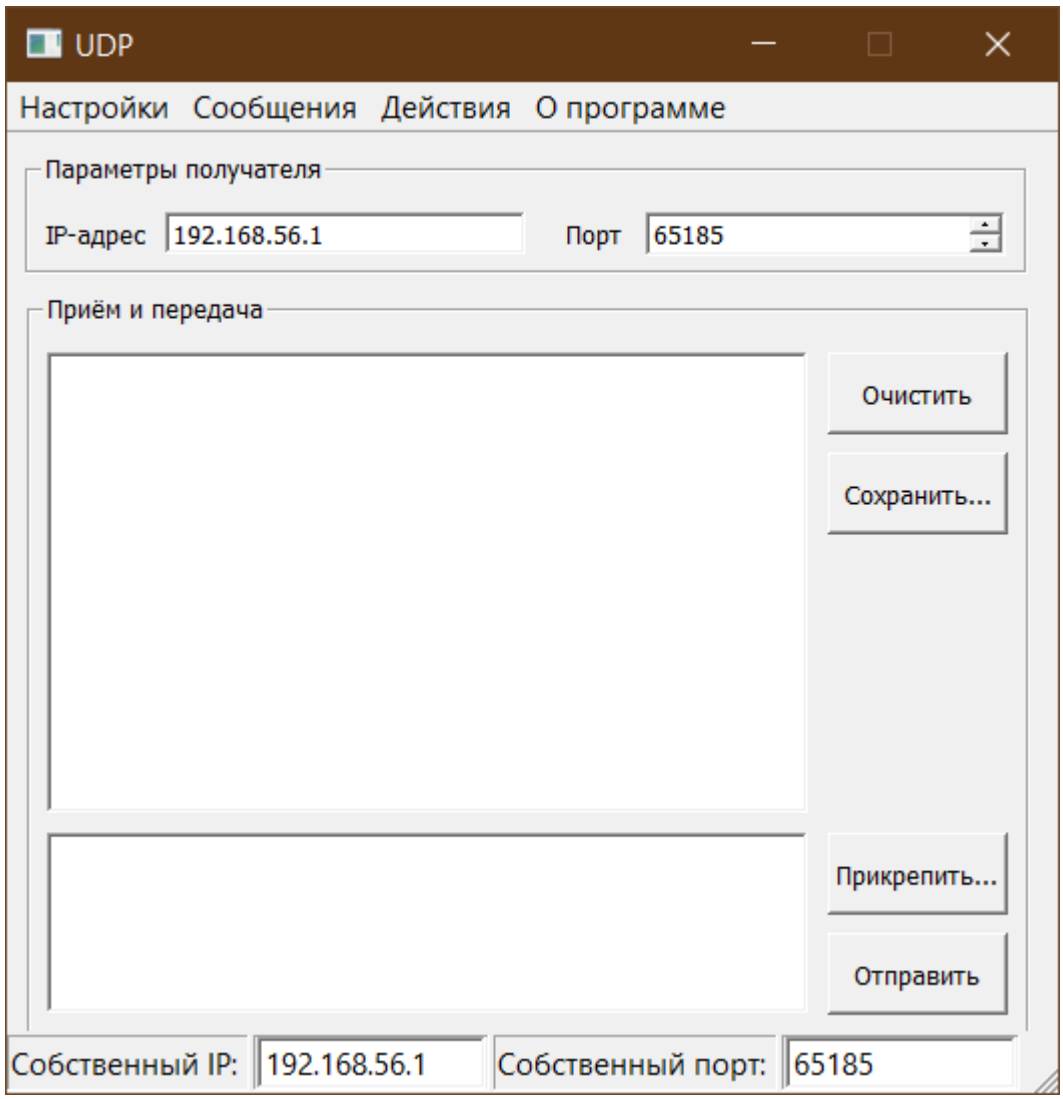


Рисунок 16 – Главное окно программы.

Главное окно программы имеет 3 основные части:

- группа «Параметры получателя»;
- группа «Приём и передача»;
- строка состояния.

Группа «Параметры получателя» включает в себя 2 поля – поле для ввода IP-адреса получателя и поле для ввода порта получателя. При старте

программы эти поля заполняются в соответствии с собственными параметрами IP-адреса и порта. В последствии пользователь может указать здесь необходимые параметры.

Группу «Приём и передача» можно так же разделить на две части – для приёма (верхняя) и для передачи (нижняя). Часть для приёма включает в себя большое поле, недоступное пользователю для редактирования, в котором отображаются принятые сообщения. В случае, если пользователь хочет, чтобы его сообщения после отправки также отображались в это поле, необходимо включить соответствующую настройку в пункте меню «Сообщения» (Рисунок 17). Очистить поле принятых сообщений можно нажатием кнопки «Очистить». А для сохранения принятого сообщения в текстовый файл необходимо в поле приёмы выделить необходимый фрагмент и нажать кнопку «Сохранить…», затем выбрать файл для сохранения.

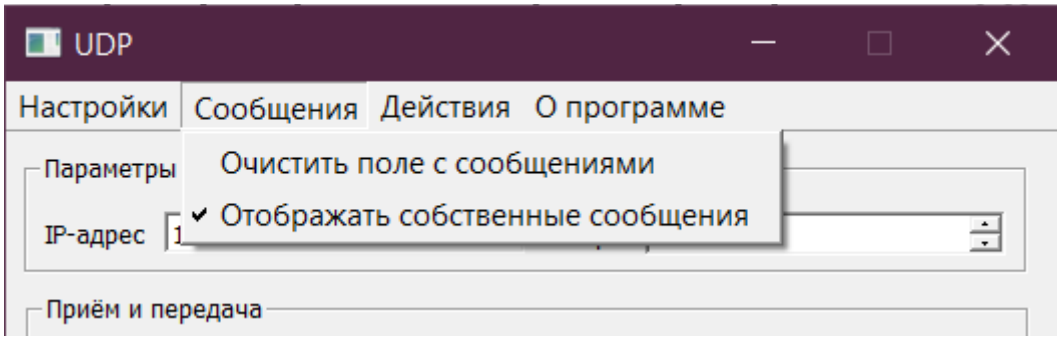


Рисунок 17 – Включение отображения собственных сообщений.

Часть передачи в группе «Приём и передача» включает в себя поле для отправки сообщений, соответственно, после нажатия кнопки «Отправить» происходит непосредственно отправка введённого текста по заданным параметрам. Также пользователь имеет возможность поместить в это поле содержимое текстового файла, нажав кнопку «Прикрепить…» и выбрав необходимый файл.

Иллюстрация обмена сообщениями между двух экземпляров программы прилагается (Рисунок 18).

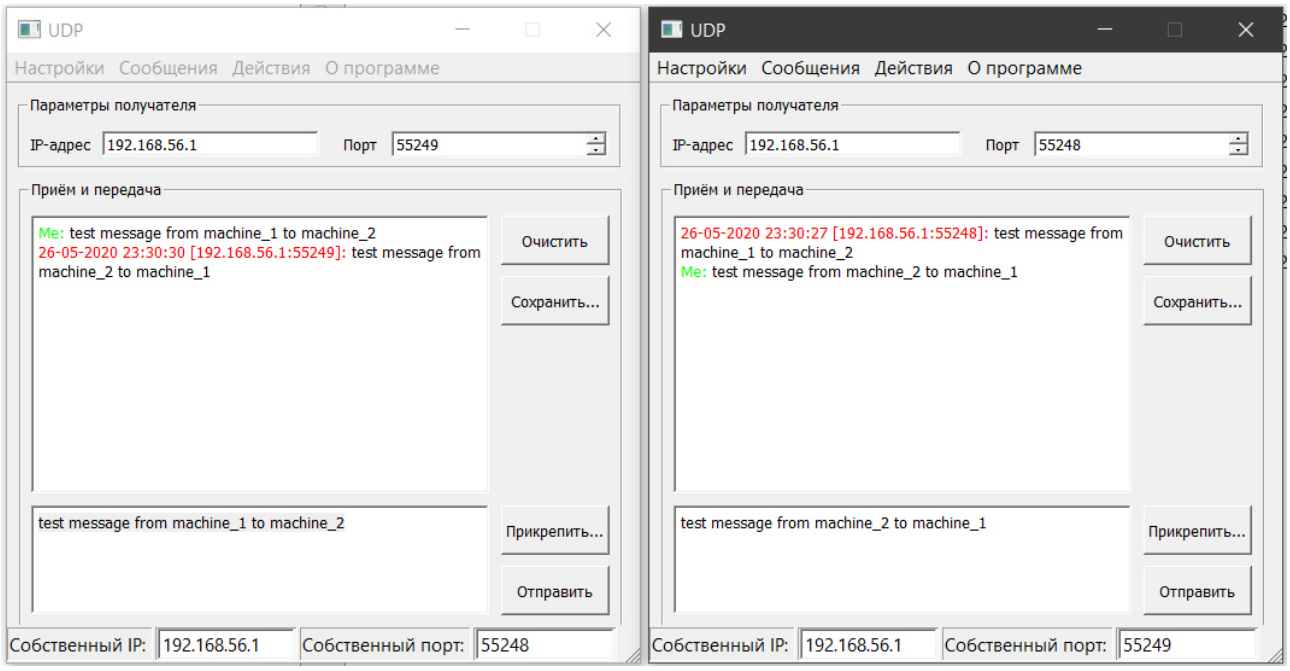


Рисунок 18 – Обмен сообщениями между программами.

Строка состояния находится в нижней части главного окна и отображает собственные IP-адрес пользователя и его порт. Эти параметры генерируются автоматически при запуске программы.

3.2. Меню приложения

Меню приложения содержит 4 пункта (Рисунок 16):

- Настройки;
- Сообщения;
- Действия;
- О программе.

Пункт «Сообщения» содержит 2 подпункта – «Очистить поле с сообщениями» и активируемые «Отображать собственные сообщения». Первый, соответственно, дублирует кнопку «Очистить» главного окна программы. Второй же, как уже описывалось ранее, включает и выключает отображение собственных сообщений в поле приёма сообщений. Собственные сообщения обозначаются пометкой «Me: » зелёного цвета, а принятые сообщения содержат параметры отправителя и время получения сообщения и окрашиваются в красный цвет (Рисунок 20).

Пункт «Действия» дублирует 3 действия главного окна – «Отправить сообщение», «Прикрепить содержимое файла…», «Сохранить выделенный фрагмент в файл…» (Рисунок 19).

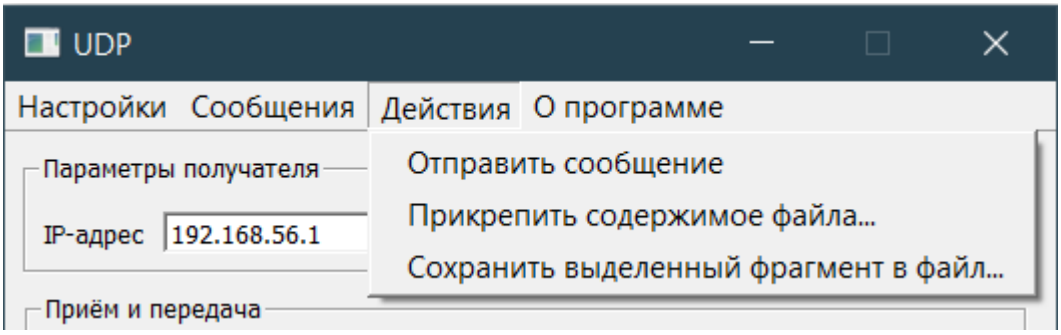


Рисунок 19 – Пункт меню «Действия».

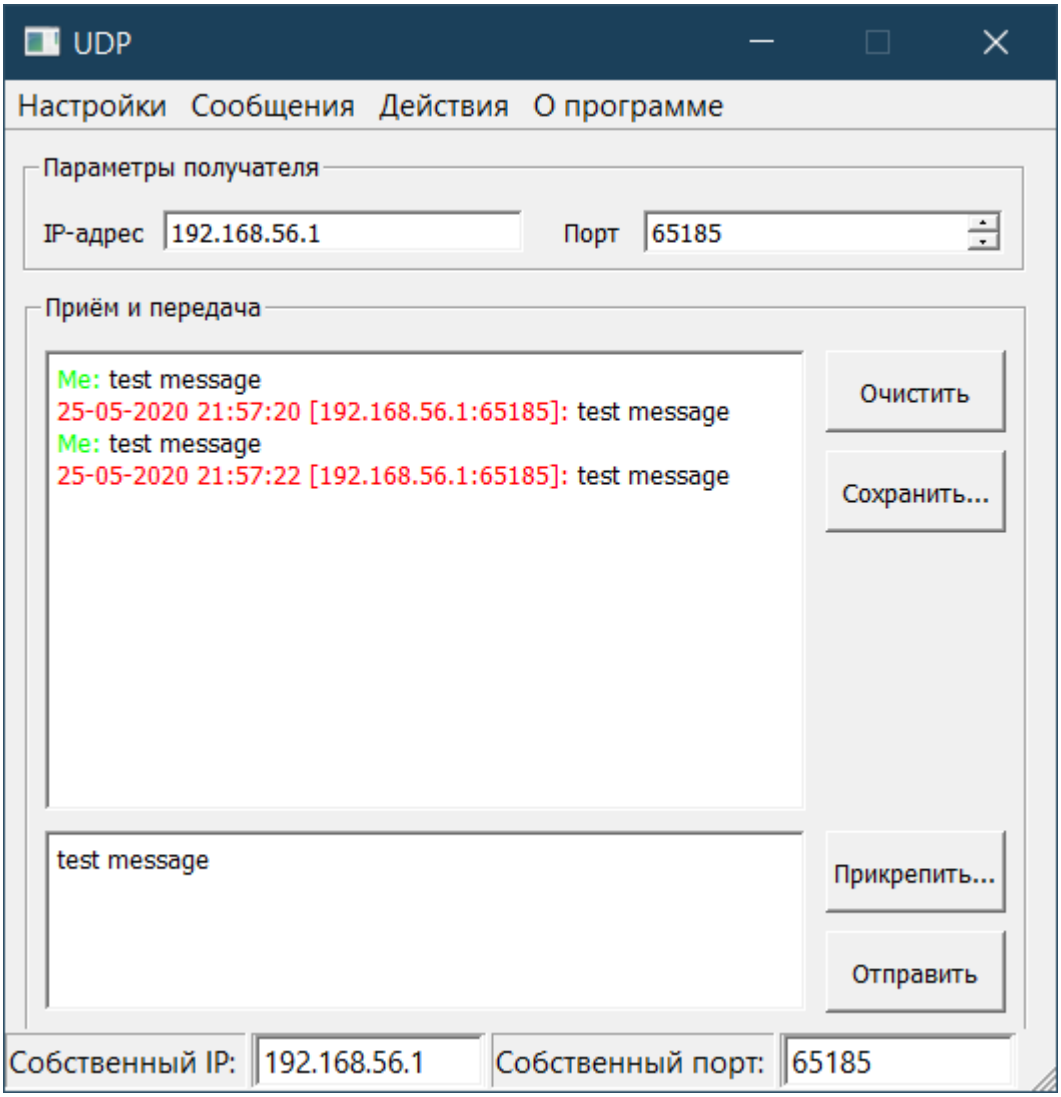


Рисунок 20 – Отправка и получение сообщений.

3.3. Пункт меню «Настройки»

При нажатии на пункт меню «Настройки» перед пользователем открывается окно настроек, в котором он может изменить как IP-адрес и порт получателя, так и собственные параметры (Рисунок 21).

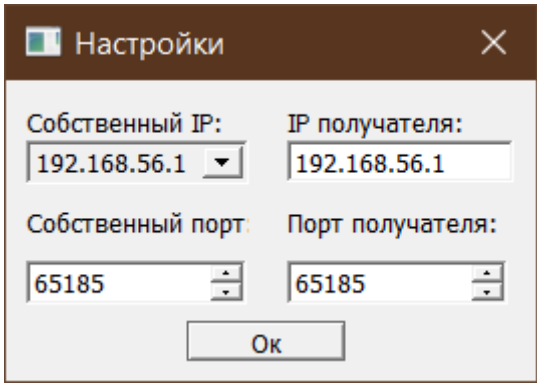


Рисунок 21 – Окно настроек.

Собственный IP в данном случае можно выбрать из выпадающего списка. После изменения параметров необходимо нажать кнопку «Ок», тогда произойдёт обновление параметров. В случае, если введённый собственный порт уже занят, то ему будет присвоено свободное значение автоматически.

3.4. Пункт меню «О программе»

При нажатии на пункт меню «О программе» пользователю открывается новое окно с информацией о программе и разработчике (Рисунок 22).

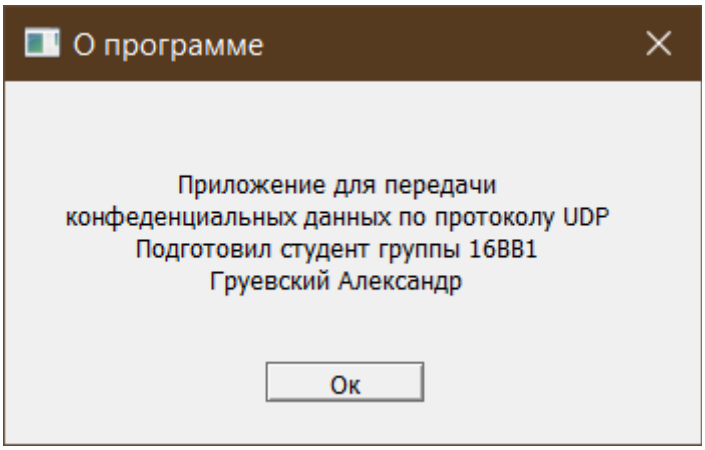


Рисунок 22 – Окно «О программе».

3.5. Обработка исключительных случаев

В случае, если пользователь захочет послать сообщение, не заполнив поле для отправки или указав некорректные IP-адрес и порт получателя, то перед ним появится соответствующее уведомление (Рисунок 23).

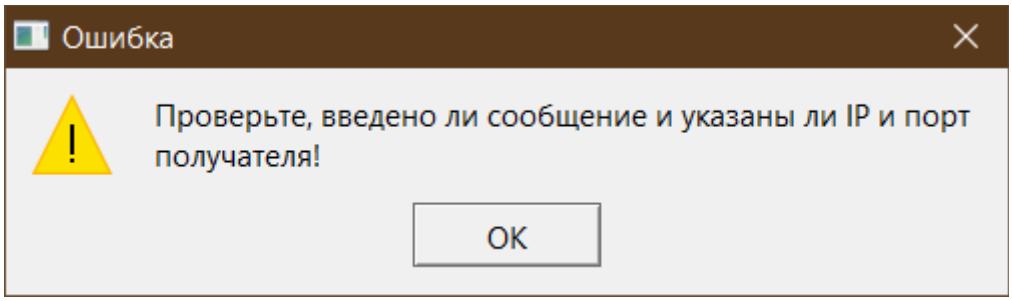


Рисунок 23 – Не введено сообщение, IP-адрес или порт.

В случае, если пользователь захочет сохранить данные в файл, не выделив нужного фрагмента, он получит соответствующее уведомление (Рисунок 24).

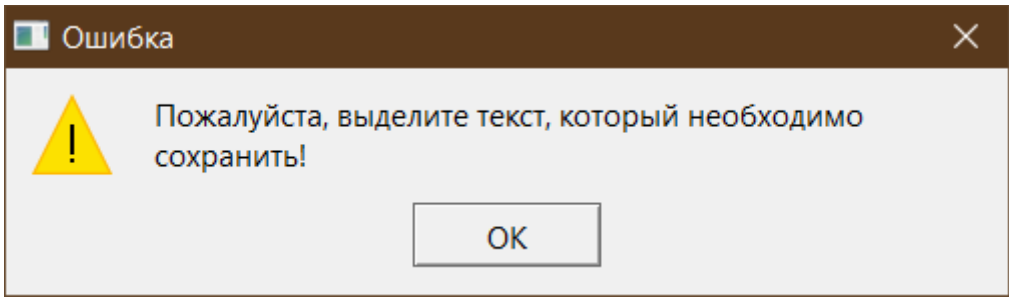


Рисунок 24 – Не выделен фрагмент для сохранения.

4. Безопасность жизнедеятельности

Современный человек много своего времени проводит за компьютером. Для правильного взаимодействия пользователя и программного обеспечения необходимо соблюдать рекомендации по обеспечению общих санитарно-технических норм к помещению и рабочим местам.

Помещение, в котором планируется производиться взаимодействие с разработанным продуктом, а точнее его размеры – площадь и объём, должны соответствовать количеству полностью оснащённых рабочих мест сотрудников. Помимо этого, необходимо обязательно учитывать соответствующие нормы освещения, температуры, чистоты воздуха, обеспечивать изоляцию от производственных шумов и т.п. На одного работающего сотрудника, основываясь на санитарные нормы труда СН 245-71, должно быть выделено помещение объёмом не менее 15 $м^3$, площадью не менее 4,5 $м^2$.

В ГОСТ 12.2.032 78 прописаны основные требования к организации рабочего место и его оборудованию. Высота рабочей поверхности стола для сотрудника должна регулироваться в пределах 680-800 мм; при отсутствии таковой возможности высота рабочей поверхности стола должна составлять 725 мм. Параметры рабочего стола должны быть следующими: ширина соответствует 800, 1200 или 1400 мм, глубина – 800 или 1000 мм при нерегулируемой высоте, которая равна 725 мм. Кроме того, пространство для ног высотой в рабочей области должно быть не менее 600 мм, а шириной – не менее 500 мм, глубиной не уровне колен – не менее 450 мм, и на уровне вытянутых ног – не менее 650 мм.

Рабочий стул (кресло) должен быть подъёмно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также по расстоянию спинки до переднего края сиденья.

Заключение

В ходе выполнения данной выпускной квалификационной работы были изучены способы шифрования данных, способы сетевых взаимодействий на уровне транспортного уровня модели OSI и стека протоколов TCP/IP, также были получены навыки в области программирования на языке C++ на платформе Qt, навыки в разработке пользовательского интерфейса.

Результатом решения поставленной задачи можно считать разработанное и протестированное приложение для обмена конфиденциальными данными с использованием протокола UDP. Разработанное программное обеспечение позволяет пользователю без страха передавать важную информацию по открытым каналам связи благодаря использованию алгоритмов RSA для шифрования и подписывания данных.

В процессе тестирования приложения были выявлены различные проблемы. При изменении собственных параметров IP-адреса и порта функция приёма сообщений перестала отлавливать входящие дейтаграммы. Однако это решилось закрытием сокета, новой его инициализацией и последующей переустановкой функций приёма входящих сообщений.

В дальнейшем программа может быть улучшена путём добавления выбора способа шифрования и кнопки обновления сгенерированных ключей шифрования. Так же можно расширить работы с файлами или добавить режим установки соединения, используя ещё и протокол TCP.

Список используемых источников

1. Олифер В., Олифер Н. Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов. 5-е изд. — СПб.: Питер, 2016. — 992 с.: ил. — (Серия «Учебник для вузов»).
2. Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. — СПб.: Питер, 2012. — 960 с.: ил.
3. Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В. Основы криптографии Учебное пособие, 2-е изд., испр. и доп. — М. Гелиос АРВ, 2002. — 480 с., ил.
4. Нильс Фергюсон, Брюс Шнайер. Практическая криптография.: Пер. с англ. — М.: Издательский дом “Вильямс”, 2004. — 432 с.: ил. — Парал. тит. англ.
5. Введение в криптографию / Под общ. ред. В. В. Ященко. — 4-е изд., доп. М.: МЦНМО, 2012. — 348 с.
6. Дмитрий Амиров. Статья «Асимметричное шифрование. Как это работает?» – URL: https://intsystem.org/security/asymmetric-encryption-how-it-work/.
7. Дмитрий Амиров. Статья «Цифровая подпись» - URL: https://intsystem.org/security/digital-signature/.
8. Гатченко Н.А., Исаев А.С., Яковлев А.Д. «Криптографическая защита информации» – СПб: НИУ ИТМО, 2012. – 142 с.
9. Бланшет Ж., Саммерфилд М. Qt 4: программирование GUI на C++. Пер. с англ. 2-е изд., доп. — М.: КУДИЦ-ПРЕСС, 2008. — 736 с.
10. Боровский А.Н. Qt4.7+. Практическое программирование на C++. – СПб.: БХВ-Петербург, 2012. — 496 с.: ил. — (Профессиональное программирование).
11. Qt, справочная документация – URL: https://doc.qt.io/.
12. Qt, справочная документация на русском языке – URL: http://doc.crossplatform.ru.

13. Библиотека Qt-Secret – URL: https://github.com/QuasarApp/Qt-Secret.

14. Форум программистов и сисадминов Киберфорум – URL: https://www.cyberforum.ru/.

15. Система вопросов и ответов о программировании Stack Overflow – URL: https://stackoverflow.com/.

16. Система вопросов и ответов о программировании Stack Overflow на русском языке – URL: https://ru.stackoverflow.com/.

Приложение А. Листинги программы

Файл «mainwindow.h»

```
/*_____________ mainwindow.h ____________________________________ */

/**
    \file       mainwindow.h
    \brief      Приложение для работы по протоколу UDP.
    \author     Грушевский А.А.
    \version    1.0
    \date       17.05.2020
    \note       Программа реализует возможность приёма и передачи
                данных по протоколу UDP.
*/

/* _____________________________________________________________ */

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

/* _____________________________________________________________ */

/// Максимальная длина сообщения
#define MAX_LEN 32751


#define _CRT_SECURE_NO_WARNINGS

/* _____________________________________________________________ */

#include <QtConcurrent/QtConcurrentRun>
#include <QtNetwork/QNetworkInterface>
#include <QtNetwork/QHostAddress>
#include <qtextstream.h>
#include <QMainWindow>
#include <QMessageBox>
#include <QFileDialog>
#include <QUdpSocket>
#include <QTextCodec>
#include <QScrollBar>
#include <QTextTable>
#include <QUdpSocket>
#include <QException>
#include <QLineEdit>
#include <QComboBox>
#include <QLineEdit>
#include <QIODevice>
#include <QAction>
```

```
#include <QThread>
#include <QAction>
#include <qfile.h>
#include <QDebug>
#include <QMenu>
#include <QFile>
#include <qrsaencryption.h> // Подключаем библиотеку Qt-Secret

/* ____________________________________________________________ */

namespace Ui {
class MainWindow;
}

/* ____________________________________________________________ */

class MainWindow : public QMainWindow
{

    Q_OBJECT

public:

    explicit MainWindow(QWidget *parent = nullptr);

    ~MainWindow();

    /// Сокет
    QUdpSocket *m_Socket;

    /// Флаг отображения собственных сообщений
    bool m_showOurMsgs;

    /// Ключи
    QByteArray priv_key, my_pub_key, pub_key;

    /// Шифровщик
    QRSAEncryption e;

    /// Маркер подписи
    const QString SIGN_MARKER = "-SIGN-";

    /* ____________________________________________________________ */

    /**
        \fn         void setSettings(QString string_localIP,
                    QString string_localPort, QString
                    string_recvIP, QString string_recvPort)
        \brief      Функция установки локальных параметров и
                    параметров получателя после изменения настроек.
        \param      string_localIP - локальный IP-адрес.
```

```
        \param      string_localPort - локальный порт.
        \param      string_recvIP - IP-адрес получателя.
        \param      string_recvPort - порт получателя.
        \note       Устанавливает новые параметры после закрытия
                    окна настроек.
    */

    void setSettings(QString string_localIP,
                     QString string_localPort,
                     QString string_recvIP,
                     QString string_recvPort);

    /* _____________________________________________________________ */


signals:

    /* _____________________________________________________________ */

    /**
        \fn         void newMessage(const QString &from, const
                    QString &message)
        \brief      Сигнал с новым сообщением и данными
                    отправителя.
        \param      &from - данные отправителя.
        \param      &message - сообщение.
        \note       Используется в качестве сигнала.
    */

    void newMessage(const QString &from, const QString &message);

    /* _____________________________________________________________ */

private slots:

    /* _____________________________________________________________ */

    /**
       \fn          void on_pushButton_clear_clicked()
       \brief       Кнопка "Очистить".
       \note        Вызывает функцию очистки поля с принятыми
                    сообщениями.
    */

    void on_pushButton_clear_clicked();

    /* _____________________________________________________________ */

    /**
       \fn          void on_pushButton_save_clicked()
       \brief       Кнопка "Сохранить...".
```

```
       \note        Вызывает функцию сохранения в файл выделенного
                    фрагмента.
    */

    void on_pushButton_save_clicked();

    /* ____________________________________________________________ */

    /**
       \fn          void on_pushButton_send_clicked()
       \brief       Кнопка "Отправить".
       \note        Вызывает функцию отправки сообщения по заданным
                    параметрам.
    */

    void on_pushButton_send_clicked();

    /* ____________________________________________________________ */

    /**
       \fn          void on_pushButton_attach_clicked()
       \brief       Кнопка "Прикрепить...".
       \note        Вызывает функцию прикрепления содержимого файла
                    в поле для отправки.
    */

    void on_pushButton_attach_clicked();

    /* ____________________________________________________________ */

    /**
        \fn         void onAppendMsg(const QString &from,
                    const QString &message)
        \brief      Функция отображения принятого сообщения.
        \param      &from - отправитель сообщения.
        \param      &message - полученное сообщение.
        \note       Отображает полученное сообщения в поле для
                    принятых данных.
    */

    void onAppendMsg(const QString &from, const QString &message);

    /* ____________________________________________________________ */

private:

    Ui::MainWindow *ui;

        /// Флаг для обнаружения ошибок, связаных с сетевым
взаимодействием
    int m_errorSoc;
```

```
        /// Поле для отображения локального IP-адреса в строке
состояния
    QLineEdit *m_lineEdit_localIP;

    /// Поле для отображения локального порта в строке состояния
    QLineEdit *m_lineEdit_localPort;

    /* _____________________________________________________________ */

    /**
       \fn          void on_menu_settings_clicked()
       \brief       Меню "Настройки".
       \note        Открывает окно с настройками.
    */

    void on_menu_settings_clicked();

    /* _____________________________________________________________ */

    /**
       \fn          void on_menu_clear_clicked()
       \brief       Меню "Сообщения" -> "Очистить поле с
                    сообщениями".
       \note        Вызывает функцию очистки поля с принятыми
                    сообщениями.
    */

    void on_menu_clear_clicked();

    /* _____________________________________________________________ */

    /**
       \fn          void on_menu_showOurMsgs_clicked()
       \brief       Меню "Сообщения" -> "Отображать собственные
                    сообщения".
       \note        Включение/отключение отображения собственных
                    сообщений.
    */

    void on_menu_showOurMsgs_clicked();

    /* _____________________________________________________________ */

    /**
       \fn          void on_menu_sendMsg_clicked()
       \brief       Меню "Действия" -> "Отправить сообщение".
       \note        Вызывает функцию отправки сообщения.
    */

    void on_menu_sendMsg_clicked();
```

```
/* _____________________________________________________________ */

/**
   \fn          void on_menu_attach_clicked()
   \brief       Меню "Сообщения" -> "Прикрепить содержимое
                файла...".
   \note        Вызывает функцию прикрепления содержимого файла
                в поле отправки.
*/

void on_menu_attach_clicked();

/* _____________________________________________________________ */

/**
   \fn          void on_menu_save_clicked()
   \brief       Меню "Сообщения" ->
                "Сохранить выделенный фрагмент ы файл...".
   \note        Вызывает функцию сохранения выделенного
                фрагмента текста в файл.
*/

void on_menu_save_clicked();

/* _____________________________________________________________ */

/**
   \fn          void on_menu_about_clicked()
   \brief       Меню "О программе".
   \note        Открывает окно "О программе".
*/

void on_menu_about_clicked();

/* _____________________________________________________________ */


/**
   \fn          void createMenuBar()
   \brief       Функция создания строки меню.
   \note        Необходима из-за невозможности создания
                подобной в дизайнере
*/

void createMenuBar();

/* _____________________________________________________________ */

/**
   \fn          void createStatusBar()
```

```
       \brief       Функция создания строки состояния.
       \note        Необходима из-за невозможности создания
                    подобной в дизайнере
    */

    void createStatusBar();

    /* _____________________________________________________________ */

    /**
        \fn         void clearMsgs()
        \brief      Функция очистки принятых сообщений.
        \note       Очищает поле для входящих сообщений.
    */

    void clearMsgs();

    /* _____________________________________________________________ */

    /**
        \fn         void recvMsgs()
        \brief      Функция приёма сообщений.
        \note       Ожидает входящие датаграмы.
    */

    void recvMsgs();

    /* _____________________________________________________________ */

    /**
        \fn         void sendMsg()
        \brief      Функция отправки сообщений.
        \note       Отправляет датаграмы по заданным параметрам.
    */

    void sendMsg();

    /* _____________________________________________________________ */

    /**
        \fn         void sendPubKey(QByteArray inStr,
                    QHostAddress address, quint64 port)
        \brief      Функция отправки ключа.
        \param      key - ключ.
        \param      address - адрес получателя.
        \param      port - порт получателя.
        \note       Отправляет ключ по заданным параметрам.
    */

     void sendPubKey(QByteArray key, QHostAddress address, quint64
port);
```

```
/* ____________________________________________________________ */

/**
   \fn          void attachFileContents()
   \brief       Меню "Сообщения" -> "Прикрепить содержимое
                файла...".
   \note        Вызывает функцию прикрепления содержимого файла
                в поле отправки.
*/

void attachFileContents();

/* ____________________________________________________________ */

/**
   \fn          void saveSelectionText()
   \brief       Меню "Сообщения" ->
                "Сохранить выделенный фрагмент ы файл...".
   \note        Вызывает функцию сохранения выделенного
                фрагмента текста в файл.
*/

void saveSelectionText();

/* ____________________________________________________________ */

/**
    \fn         QString getIP()
    \brief      Функция получения локального IP-адреса.
    \note       Выявляет IP-адреса локальной машины.
*/

QString getIP();

/* ____________________________________________________________ */

/**
    \fn         QString getDataTime()

    \brief      Функция получения текущих даты и времени.
    \note       Необходима для установки времени получения
                сообщения.
*/

QString getDataTime();

/* ____________________________________________________________ */

/**
    \fn         QByteArray sign(QByteArray inStr)
```

```
        \brief      Функция подписи и шифрования сообщения.
        \param      inStr - сообщение.
        \note       Подписывает и шифрует сообщение.
    */

    QByteArray sign(QByteArray inStr);

    /* _____________________________________________________________ */

    /**
        \fn         requestKey()
        \brief      Функция запроса ключа.
        \note       Отправляет датаграмму с запросом ключа.
    */

    void requestKey();

    /* _____________________________________________________________ */

    /**
        \fn         sendSignMsg()
        \brief      Отправка закодированного сообщения.
        \note       Отправляет закодированное сообщение.
    */

    void sendSignMsg();

    /* _____________________________________________________________ */

};

/* _____________________________________________________________ */

#endif /// MAINWINDOW_H

/* __________ end of mainwindow.h _____________________________ */
```

Файл «mainwindow.cpp»

```
/*_________________ mainwindow.cpp _____________________________ */

/**
    \file       mainwindow.cpp
    \brief      Приложение для работы по протоколу UDP.
    \author     Грушевский А.А.
    \version    1.0
    \date       17.05.2020
    \note       Программа реализует возможность приёма и передачи
                данных протоколу UDP.
*/
```

```
/* ____________________________________________________________ */

#include "ui_mainwindow.h"
#include "mainwindow.h"
#include "settings.h"
#include "about.h"

/* ____________________________________________________________ */

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    /// Задание фиксированных размеров окна
    QWidget::setFixedSize(525, 505);

    /// Выбор кодировки
    QTextCodec* codec = QTextCodec::codecForName("UTF-8");
    /// Установка кодировки
    QTextCodec::setCodecForLocale(codec);

    /// Инициализация графического интерфейса
    ui->setupUi(this);
    /// Инициализация строки меню
    createMenuBar();
    /// Инициализация строки состояния
    createStatusBar();

    /// Инициализация сокета
    m_Socket = new QUdpSocket(this);
    /// Определение локального IP-адреса
        QHostAddress  local_address(MainWindow::m_lineEdit_localIP-
>text());
    /// Связывание сокета с локальным IP-адресом
    m_errorSoc = m_Socket->bind(local_address, 0);

    /// Если возникли ошибки при связывании
    if (m_errorSoc == -1)
    {
        /// Вызов диалогового окна, уведомляющего об ошибке
        QMessageBox::warning(this,
                             "Ошибка",
                             "Не удалось связаться с сокетом!");
        /// Закрытие приложения
        exit(0);
    }

    /// Определение выделенного локального порта
           m_lineEdit_localPort->setText(QString().setNum(m_Socket-
>localPort()));
    /// Отображение выделенного порта в поле порта приёмника
```

```
                ui->spinBox_recvPort->setValue(m_lineEdit_localPort-
>text().toInt());

    /// Отключение отображения собственных сообщений
    m_showOurMsgs = false;

    /// Вспомогательная строка для установки регулярного выражения
       QString  ipRange  =  "(?:[1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|
25[0-5])";
    /// Создание маски для поля ввода IP-адреса получателя
    QRegExp ipRegex     ("^" + ipRange
                      + "\\." + ipRange
                      + "\\." + ipRange
                      + "\\." + ipRange
                      + "$");
    /// Применение маски на валидатор
     QRegExpValidator *ipValidator = new QRegExpValidator(ipRegex,
this);
     /// Установка валидатора с заданной маской на поле ввода IP-
адреса
    ui->lineEdit_recvIP->setValidator(ipValidator);

    /// Генерация пары ключей
                        e.generatePairKey(my_pub_key,      priv_key,
QRSAEncryption::Rsa::RSA_64);

    /// Установка функции приёма сообщений
              connect(m_Socket,    &QUdpSocket::readyRead,    this,
&MainWindow::recvMsgs);
    /// Уставновка функции обработки принятых сообщений
    connect(this,
            SIGNAL(newMessage(QString, QString)),
            this,
            SLOT(onAppendMsg(QString, QString)));
}

/* ____________________________________________________________________ */

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

/* ____________________________________________________________________ */

/**
   \fn         void MainWindow::on_menu_settings_clicked()
   \brief      Меню "Настройки".
   \note       Открывает окно с настройками.
*/
```

```
void MainWindow::on_menu_settings_clicked()
{
    /// Создание окна настроек
    settings *s = new settings(this,
                               m_lineEdit_localIP,
                               m_lineEdit_localPort,
                               ui->lineEdit_recvIP,
                               ui->spinBox_recvPort);
    /// Отображение окна настроек
    s->exec();
} /// on_menu_settings_clicked

/* ____________________________________________________________ */

/**
   \fn         void MainWindow::on_menu_clear_clicked()
   \brief      Меню "Сообщения" -> "Очистить поле с сообщениями".
   \note       Вызывает функцию очистки поля с принятыми
               сообщениями.
*/

void MainWindow::on_menu_clear_clicked()
{
    /// Вызов функции очистки сообщений
    clearMsgs();
} /// on_menu_clear_clicked

/* ____________________________________________________________ */

/**
   \fn         void MainWindow::on_menu_showOurMsgs_clicked()
   \brief      Меню "Сообщения" -> "Отображать собственные
               сообщения".
   \note       Включение/отключение отображения собственных
               сообщений.
*/

void MainWindow::on_menu_showOurMsgs_clicked()
{
    /// Если отображение было включено
    if (m_showOurMsgs == true)
    {
        /// Отключение отображения
        m_showOurMsgs = false;
    }
    else
    {
        /// Включение отображения
        m_showOurMsgs = true;
    }
} /// on_menu_showOurMsgs_clicked
```

```
/* _____________________________________________________________ */

/**
   \fn         void MainWindow::on_menu_sendMsg_clicked()
   \brief      Меню "Действия" -> "Отправить сообщение".
   \note       Вызывает функцию отправки сообщения.
*/

void MainWindow::on_menu_sendMsg_clicked()
{
    /// Вызов функции отправки сообщения
    sendMsg();
} /// on_menu_sendMsg_clicked

/* _____________________________________________________________ */

/**
   \fn         void MainWindow::on_menu_attach_clicked()
   \brief      Меню "Сообщения" -> "Прикрепить содержимое
               файла...".
   \note       Вызывает функцию прикрепления содержимого файла в
               поле отправки.
*/

void MainWindow::on_menu_attach_clicked()
{
    /// Вызов функции прикрепления содержимого файла
    attachFileContents();
} /// on_menu_attach_clicked

/* _____________________________________________________________ */

/**
   \fn         void MainWindow::on_menu_save_clicked()
   \brief      Меню "Сообщения" -> "Сохранить выделенный фрагмент ы
               файл...".
   \note       Вызывает функцию сохранения выделенного фрагмента
               текста в файл.
*/

void MainWindow::on_menu_save_clicked()
{
     /// Вызов функции сохранения выделенного фрагмента текста в
файл
    saveSelectionText();
} /// on_menu_save_clicked

/* _____________________________________________________________ */

/**
```

```
   \fn         void MainWindow::on_menu_about_clicked()
   \brief      Меню "О программе".
   \note       Открывает окно "О программе".
*/

void MainWindow::on_menu_about_clicked()
{
    /// Создания окна "О программе"
    about *a = new about();
    /// Открытие окна "О программе"
    a->exec();
} /// on_menu_about_clicked

/* ____________________________________________________________ */

/**
   \fn         void MainWindow::on_pushButton_clear_clicked()
   \brief      Кнопка "Очистить".
   \note       Вызывает функцию очистки поля с принятыми
               сообщениями.
*/

void MainWindow::on_pushButton_clear_clicked()
{
    /// Вызов функции очистки поля приёма данных
    clearMsgs();
} /// on_pushButton_clear_clicked

/* ____________________________________________________________ */

/**
   \fn         void MainWindow::on_pushButton_send_clicked()
   \brief      Кнопка "Отправить".
   \note       Вызывает функцию отправки сообщения по заданным
               параметрам.
*/

void MainWindow::on_pushButton_send_clicked()
{
    /// Вызов функции отправки сообщения
    sendMsg();
} /// on_pushButton_send_clicked

/* ____________________________________________________________ */

/**
   \fn         void MainWindow::on_pushButton_save_clicked()
   \brief      Кнопка "Сохранить...".
   \note       Вызывает функцию сохранения в файл выделенного
               фрагмента.
*/
```

```
void MainWindow::on_pushButton_save_clicked()
{
    /// Вызов функции сохранения выделенного фаргмента в файл
    saveSelectionText();
} /// on_pushButton_save_clicked

/* ____________________________________________________________ */

/**
   \fn         void MainWindow::on_pushButton_attach_clicked()
   \brief      Кнопка "Прикрепить...".
   \note       Вызывает функцию прикрепления содержимого файла
               в поле для отправки.
*/

void MainWindow::on_pushButton_attach_clicked()
{
    /// Вызов функции прикрепления содержимого файла
    attachFileContents();
} /// on_pushButton_attach_clicked

/* ____________________________________________________________ */

/**
   \fn         void MainWindow::createMenuBar()
   \brief      Функция создания строки меню.
   \note       Необходима из-за невозможности создания подобной в
               дизайнере
*/

void MainWindow::createMenuBar()
{
    /// "Настройки"
     QTextCodec::setCodecForLocale(QTextCodec::codecForName("Windows-1251"));
               QAction   *action_setting   =   this->ui->menuBar->addAction("Настройки");
    connect(action_setting,
            &QAction::triggered,
            this,
            &MainWindow::on_menu_settings_clicked);

    /// "Сообщения"
    QMenu *menu_msgs = new QMenu("Сообщения");
    ui->menuBar->addMenu(menu_msgs);

        /// "Очистить поле с сообщениями"
        QAction *action_clear = new QAction(
                    "Очистить поле с сообщениями",
                    menu_msgs);
```

```
        connect(action_clear,
                &QAction::triggered,
                this,
                &MainWindow::on_menu_clear_clicked);
        menu_msgs->addAction(action_clear);
        /// "Отображать собственные сообщения"
        QAction *action_showOurMsgs = new QAction(
                    "Отображать собственные сообщения",
                    menu_msgs);
        action_showOurMsgs->setCheckable(true);
        connect(action_showOurMsgs,
                &QAction::triggered,
                this,
                &MainWindow::on_menu_showOurMsgs_clicked);
        menu_msgs->addAction(action_showOurMsgs);

    /// "Действия"
    QMenu *menu_actions = new QMenu("Действия");
    ui->menuBar->addMenu(menu_actions);

        /// "Отправить сообщение"
        QAction *action_sendMsg = new QAction(
                    "Отправить сообщение",
                    menu_actions);
        connect(action_sendMsg,
                &QAction::triggered,
                this,
                &MainWindow::on_menu_sendMsg_clicked);
        menu_actions->addAction(action_sendMsg);
        /// "Прикрепить содержимое файла..."
        QAction *action_attach = new QAction(
                    "Прикрепить содержимое файла...",
                    menu_actions);
        connect(action_attach,
                &QAction::triggered,
                this,
                &MainWindow::on_menu_attach_clicked);
        menu_actions->addAction(action_attach);
        /// "Сохранить выделенный фрагмент в файл..."
        QAction *action_save = new QAction(
                    "Сохранить выделенный фрагмент в файл...",
                    menu_actions);
        connect(action_save,
                &QAction::triggered,
                this,
                &MainWindow::on_menu_save_clicked);
        menu_actions->addAction(action_save);

    /// "О программе"
         QAction  *action_about  =  this->ui->menuBar->addAction("О
программе");
```

```
    connect(action_about,
            &QAction::triggered,
            this,
            &MainWindow::on_menu_about_clicked);
} /// createMenuBar

/* ____________________________________________________________________ */

/**
   \fn         void MainWindow::createStatusBar()
   \brief      Функция создания строки состояния.
   \note       Необходима из-за невозможности создания подобной в
               дизайнере
*/

void MainWindow::createStatusBar()
{
    /// (QLabel) Собственный IP
    QLabel *label_localIp = new QLabel("Собственный IP: ");
    statusBar()->addWidget(label_localIp);

    /// (QLineEdit) IP получателя
    m_lineEdit_localIP = new QLineEdit();
    m_lineEdit_localIP->setReadOnly(true);
    statusBar()->addWidget(m_lineEdit_localIP);
    m_lineEdit_localIP->setText(getIP());
    ui->lineEdit_recvIP->setText(getIP());

    /// (QLabel) Собственный порт
    QLabel *label_localPort =
            new QLabel("Собственный порт: ");
    statusBar()->addWidget(label_localPort);

    /// (QLineEdit) Порт получателя
    m_lineEdit_localPort = new QLineEdit();
    m_lineEdit_localPort->setReadOnly(true);
    statusBar()->addWidget(m_lineEdit_localPort);
} /// createStatusBar

/* ____________________________________________________________________ */

/**
    \fn         void MainWindow::clearMsgs()
    \brief      Функция очистки принятых сообщений.
    \note       Очищает поле для входящих сообщений.
*/

void MainWindow::clearMsgs()
{
    /// Очистка поля
    ui->textBrowser_recvMsg->setText("");
```

```
} /// clearMsgs

/* _____________________________________________________________ */

/**
    \fn         QString MainWindow::getIP()
    \brief      Функция получения локального IP-адреса.
    \note       Выявляет IP-адреса локальной машины.
*/

QString MainWindow::getIP()
{
    /// Перебор списка адресов текущей машины
                  foreach    (const    QHostAddress    &address,
QNetworkInterface::allAddresses())
    {
        /// Если адрес из списка соответствует протоколу IPv4
        if ( (address.protocol() == QAbstractSocket::IPv4Protocol)
             && (address != (QHostAddress::LocalHost)) )
        {
            /// Возврат подходящего адреса
            return address.toString();
        }
    }

    /// В случае ошибки
    return "getIP_error";
} /// getIP

/* _____________________________________________________________ */

/**
    \fn       void MainWindow::setSettings(QString string_localIP,
                QString string_localPort,
                QString string_recvIP,
                QString string_recvPort)
    \brief     Функция установки локальных параметров и параметров
               получателя после изменения настроек.
    \param      string_localIP - локальный IP-адрес.
    \param      string_localPort - локальный порт.
    \param      string_recvIP - IP-адрес получателя.
    \param      string_recvPort - порт получателя.
    \note      Устанавливает новые параметры после закрытия окна
               настроек.
*/

void MainWindow::setSettings(QString string_localIP,
                             QString string_localPort,
                             QString string_recvIP,
                             QString string_recvPort)
{
```

```
    /// Отображение новых значений
    MainWindow::m_lineEdit_localIP->setText(string_localIP);
    MainWindow::m_lineEdit_localPort->setText(string_localPort);
    ui->lineEdit_recvIP->setText(string_recvIP);
    ui->spinBox_recvPort->setValue(string_recvPort.toInt());

    /// Закрытие сокета
    m_Socket->close();

    /// Инициализация сокета
    m_Socket = new QUdpSocket(this);
    /// Определение нового локального IP-адреса
        QHostAddress  local_address(MainWindow::m_lineEdit_localIP-
>text());
    /// Определение нового локального порта
         quint16  local_port  =  MainWindow::m_lineEdit_localPort-
>text().toUShort();
    /// Связывание сокета с новым локальным IP-адресом и портом
    m_errorSoc = m_Socket->bind(local_address, local_port);

    /// Если выбранный порт занят
    if (m_errorSoc != 1)
    {
        /// Установка автоматического выбора порта
        m_errorSoc = m_Socket->bind(local_address, 0);
        /// Изменение отображения локального порта
           m_lineEdit_localPort->setText(QString().setNum(m_Socket-
>localPort()));
    }

    /// Переустановка функции приёма сообщений
             disconnect(m_Socket,    &QUdpSocket::readyRead,    this,
&MainWindow::recvMsgs);
               connect(m_Socket,    &QUdpSocket::readyRead,    this,
&MainWindow::recvMsgs);

    /// Переустановка функции отображения сообщений
    disconnect(this,
               SIGNAL(newMessage(QString, QString)),
               this,
               SLOT(onAppendMsg(QString, QString)));
    connect(this,
            SIGNAL(newMessage(QString, QString)),
            this,
            SLOT(onAppendMsg(QString, QString)));
} /// setSettings

/* ____________________________________________________________ */

/**
    \fn         QString MainWindow::getDataTime()
```

```
    \brief      Функция получения текущих даты и времени.
    \note       Необходима для установки времени получения
                сообщения.
*/

QString MainWindow::getDataTime()
{
    struct tm tstruct;
    time_t now = time(nullptr);
    /// Результат
    char cBuf[80];
    localtime_s(&tstruct, &now);
    strftime(cBuf, sizeof(cBuf), "%d-%m-%Y %X", &tstruct);

    return cBuf;
} /// getDataTime

/* _____________________________________________________________ */

/**
    \fn         void MainWindow::attachFileContents()
    \brief      Функция прикрепления содержимого файла.
    \note       Отображает содержимое выбранного файла в поле для
                отправки.
*/

void MainWindow::attachFileContents()
{
     QTextCodec::setCodecForLocale(QTextCodec::codecForName("Window
s-1251"));
    /// Файл
    QFile file;
    /// Поток для работы с файлом
    QTextStream stream;
    /// Строка с будущим содержимым файла
    QString wholeFile;

    /// Диалог выбора файла
    QString fileName = QFileDialog::getOpenFileName(this,
                                                    "Выберите
файл",
                                                QString(),
                                                "*.txt");

    /// Если файл был выбран
    if (fileName != nullptr)
    {
        /// Установкка имени файла
        file.setFileName(fileName);
        /// Установка режима чтения
        file.open(QIODevice::ReadOnly);
```

```
        /// Связывание файла с потоком
        stream.setDevice(&file);
        /// Установка кодировки
        stream.setCodec(QTextCodec::codecForName("UTF-8"));
        /// Сохранение содержимого файла
        wholeFile = stream.readAll();

        /// Если содержимого слишком много
        if (wholeFile.length() > MAX_LEN)
        {
            /// Обрезка лишнего хвоста содержимого файла
            wholeFile.resize(MAX_LEN);
        }

        /// Вывод содержимого файла
        ui->textEdit_sendMsg->setText(wholeFile);

        /// Создание полосы прокрутки
                      QScrollBar   *bar   =   ui->textEdit_sendMsg-
>verticalScrollBar();
        /// Отображение конца поля
        bar->setValue(bar->maximum());
    }
} /// attachFileContents

/* _____________________________________________________________ */

/**
    \fn         void MainWindow::saveSelectionText()
    \brief      Функция сохранения выделенного фрагмента файла.
    \note       Сохраняет выделенный в поле приёма фрагмент текста
                в выбранный файл.
*/

void MainWindow::saveSelectionText()
{
    /// Создание курсора
    QTextCursor cursor(ui->textBrowser_recvMsg->textCursor());
    /// Сохранение выделенного
    QString string = cursor.selectedText();
    /// Замена символов переноса
    QString replaceString = string.replace(0x2029, "\n");
    /// Получение массива байт
    QByteArray data = replaceString.toUtf8();
    /// Если текст был выделен
    if (data != nullptr)
    {
        /// Переменная файла
        QFile file;
        /// Диалоговое окно выбора файла
        QString fileName = QFileDialog::getOpenFileName(this,
```

```
                                                        "Выберите
файл",
                                                      QString(),
                                                      "*.txt");
        /// Если файл был выбран
        if (fileName != nullptr)
        {
            /// Установка имени файла
            file.setFileName(fileName);
            /// Установка режима записи в файл
            file.open(QIODevice::WriteOnly);
            /// Запись выделенного в файл
            file.write(data);
            /// Закрытие файла
            file.close();
        }
    }
    else
    {
        QMessageBox::warning(
                    this,
                    "Ошибка",
                    "Пожалуйста, выделите текст, "
                    "который необходимо сохранить!");
    }
} /// saveSelectionText

/* ____________________________________________________________ */

/**
    \fn         void MainWindow::recvMsgs()
    \brief      Функция приёма сообщений.
    \note       Ожидает входящие датаграмы.
*/

void MainWindow::recvMsgs()
{
    /// Принятая датаграма
    QByteArray datagram;
    /// IP-адресс отправителя
    QHostAddress senderIp;
    /// Порт отправителя
    quint16 senderPort;
    /// Сообщение
    QByteArray recvMsgArr;
    /// Строка с данными отправителя
    QString from;

    /// Получение размера датаграмы
    datagram.resize(m_Socket->pendingDatagramSize());
    /// Чтение датаграмы
```

```
    m_Socket->readDatagram(datagram.data(),
                           datagram.size(),
                           &senderIp,
                           &senderPort);

    /// Если удалось получить IP-адрес и порт отправителя
    if ((!senderIp.isNull()) && (senderPort != 0))
    {
            /// Если пришёл запрос на получение ключа и ключ
отправителя
        if (datagram[0] == '0')
        {
            /// Обрезка первого байта
            datagram.remove(0,1);
            /// Присвоение публичного ключа получателя
            pub_key = datagram;
            /// Отправка публичного ключа
            sendPubKey(my_pub_key, senderIp, senderPort);
        }
        /// Если пришёл публичный ключ
        else if (datagram[0] == '1')
        {
            /// Обрезка первого байта
            datagram.remove(0,1);
            /// Присвоение публичного ключа получателя
            pub_key = datagram;

            sendSignMsg();

        }
        /// Если пришло сообщение
        else if (datagram[0] == '2')
        {
            /// Обрезка первого байта
            datagram.remove(0,1);
            /// Дешифрация сообщения приватным ключом
                auto decodeMessage = e.decode(datagram, priv_key,
QRSAEncryption::Rsa::RSA_64);
            /// Проверка подписи
                   if (e.checkSignMessage(decodeMessage, pub_key,
QRSAEncryption::Rsa::RSA_64))
            {
                /// Поиск позиции маркера подписи
                                          auto signStartPos =
decodeMessage.indexOf(SIGN_MARKER);
                /// Получение сообщения
                recvMsgArr = decodeMessage.left(signStartPos);
            }
            /// Формирование строки с параметрами отправителя
                        from = senderIp.toString() + ":" +
QString::number(senderPort);
```

```
            /// Передача сообщения и данных отправителя в функцию
вывода сообщения
            emit newMessage(from, recvMsgArr);
        }
    }
} /// recvMsgs

/* ____________________________________________________________ */

/**
    \fn         sendSignMsg()
    \brief      Отправка закодированного сообщения.
    \note       Отправляет закодированное сообщение.
*/

void MainWindow::sendSignMsg()
{
    /// Сохранение в переменную текста из поля для отправки
    QString data = ui->textEdit_sendMsg->toPlainText();
    QByteArray dataArray = data.toUtf8();

    /// IP-адресс получателя
    QHostAddress address(ui->lineEdit_recvIP->text());

    /// Порт получателя
    quint64 port = ui->spinBox_recvPort->text().toUInt();

    /// Подписывание и шифрование сообщения
    QByteArray signMsg = sign(dataArray);
    QByteArray msgArr;
    /// Формирование сообщения
    msgArr[0] = '2';
    msgArr.append(signMsg);

    /// Отправка сообщения
    m_errorSoc = m_Socket->writeDatagram(
                                    msgArr,
                                    address,
                                    static_cast<quint16>(port));

    /// Если при отправке не возникло ошибок
    if (m_errorSoc != -1)
    {
        /// Если включено отображение собственных сообщений
        if (m_showOurMsgs)
        {
            /// Объявлеине курсора
                    QTextCursor  cursor(ui->textBrowser_recvMsg-
>textCursor());
            /// Снятие выделения
            cursor.clearSelection();
```

```
            /// Помещение курсора в конец поля приёма
            cursor.movePosition(QTextCursor::End);
            /// Применение позиции курсора
            ui->textBrowser_recvMsg->setTextCursor(cursor);

            /// Сохранение исходного цвета текста
            QColor color = ui->textBrowser_recvMsg->textColor();

                 /// Установка отображения собственного сообщения
зелёным цветом
            ui->textBrowser_recvMsg->setTextColor(Qt::green);
            /// Обозначение собственного сообщения
            ui->textBrowser_recvMsg->append("Me: ");

            /// Возврат исходного цвета текста
            ui->textBrowser_recvMsg->setTextColor(color);
            /// Помещение курсора в конец поля приёма
            cursor.movePosition(QTextCursor::End);
            /// Вставка сообщения в окно приёма
            ui->textBrowser_recvMsg->insertPlainText(data);
        }
    }
    else
    {
        QMessageBox::warning(
                    this,
                   "Ошибка",
                    "Не удалось отправить сообщение! Пожалуйста, "
                                "проверьте правильность введённых
параметров.");
    }
}

/* _____________________________________________________________ */

/**
    \fn         void MainWindow::sendMsg()
    \brief      Функция отправки сообщений.
    \note       Отправляет датаграмы по заданным параметрам.
*/

void MainWindow::sendMsg()
{
    /// Если все поля заполнены правильно
    if ( (ui->lineEdit_recvIP->text() != "")
         && (ui->textEdit_sendMsg->toPlainText() != "") )
    {
         /// Если количество символов превышает допустимую длину в
32751
```

```
                if  (ui->textEdit_sendMsg->toPlainText().length()  >
MAX_LEN)
        {
            /// Сохранение текущего сообщения
                        QString  temp_str  =  ui->textEdit_sendMsg-
>toPlainText();
            /// Изменение длины сообщения на 32751
            temp_str.resize(MAX_LEN);
            /// Установка обрезанного сообщения
            ui->textEdit_sendMsg->setText(temp_str);

            /// Создание полосы прокрутки
                         QScrollBar  *bar  =  ui->textEdit_sendMsg-
>verticalScrollBar();
            /// Отображение конца поля
            bar->setValue(bar->maximum());
        }

        requestKey();

    }
    else
    {
        QMessageBox::warning(
                    this,
                    "Ошибка",
                    "Введите сообщение!");
    }
} /// sendMsg

/* _____________________________________________________________ */

/**
    \fn         requestKey()
    \brief      Функция запроса ключа.
    \note       Отправляет датаграмму с запросом ключа.
*/

void MainWindow::requestKey()
{
    /// IP-адресс получателя
    QHostAddress address(ui->lineEdit_recvIP->text());

    /// Порт получателя
    quint64 port = ui->spinBox_recvPort->text().toUInt();

        ///  Формирование  запроса  на  получение  публичного  ключа
получателя
    QByteArray arrReqKey;
    arrReqKey[0] = '0';
    arrReqKey.append(my_pub_key);
```

```
    /// Отправка запроса
    m_errorSoc = m_Socket->writeDatagram(
                                    arrReqKey,
                                    address,
                                    static_cast<quint16>(port));
} /// requestKey

/* ____________________________________________________________ */

/**
        \fn         void sendPubKey(QByteArray inStr,
                    QHostAddress address, quint64 port)
        \brief      Функция отправки ключа.
        \param      key - ключ.
        \param      address - адрес получателя.
        \param      port - порт получателя.
        \note       Отправляет ключ по заданным параметрам.
*/

void MainWindow::sendPubKey(QByteArray key, QHostAddress address,
quint64 port)
{
    QByteArray keyArr;
    /// Формирование сообщения
    keyArr[0] = '1';
    keyArr.append(key);

    /// Отправка ключа
    m_errorSoc = static_cast<int>(m_Socket->writeDatagram(
                                    keyArr,
                                    address,
                                    static_cast<quint16>(port)));

    /// Если при отправке возникли ошибки
    if (m_errorSoc == -1)
    {
        QMessageBox::warning(
                    this,
                   "Ошибка",
                    "Не удалось отправить сообщение! Пожалуйста, "
                                "проверьте правильность введённых
параметров.");
    }
} /// sendPubKey

/* ____________________________________________________________ */

/**
    \fn         void MainWindow::onAppendMsg(const QString &from,
                const QString &message)
```

```
    \brief      Функция отображения принятого сообщения.
    \param      &from - отправитель сообщения.
    \param      &message - полученное сообщение.
    \note       Отображает полученное сообщения в поле для
                принятых данных.
*/

void MainWindow::onAppendMsg(const QString &from, const QString
&message)
{
    /// Создание курсора
    QTextCursor cursor(ui->textBrowser_recvMsg->textCursor());
    /// Снятие выделения
    cursor.clearSelection();

    /// Установка курсора в конец
    cursor.movePosition(QTextCursor::End);
    /// Примемнение позиции курсора
    ui->textBrowser_recvMsg->setTextCursor(cursor);
    /// Установка красного цвета
    ui->textBrowser_recvMsg->setTextColor(Qt::red);
    /// Добавление времени получения и данных отправителя
     ui->textBrowser_recvMsg->append(getDataTime() + " [" + from +
"]: ");

    /// Установка курсора в конец
    cursor.movePosition(QTextCursor::End);
    /// Примемнение позиции курсора
    ui->textBrowser_recvMsg->setTextCursor(cursor);
    /// Установка чёрного цвета
    ui->textBrowser_recvMsg->setTextColor(Qt::black);
    /// Добавление текста сообщения
    ui->textBrowser_recvMsg->textCursor().insertText(message);

    /// Создание полосы прокрутки
                QScrollBar    *bar    =    ui->textBrowser_recvMsg-
>verticalScrollBar();
    /// Отображение конца поля
    bar->setValue(bar->maximum());
} /// onAppendMsg

/* _____________________________________________________________ */

/**
    \fn         QByteArray sign(QByteArray inStr)
    \brief      Функция подписи и шифрования сообщения.
    \param      inStr - сообщение.
    \note       Подписывает и шифрует сообщение.
*/

QByteArray MainWindow::sign(QByteArray inStr)
```

```
{
    /// Подпись сообщения приватным ключом
          auto  signedMessage  =  e.signMessage(inStr,  priv_key,
QRSAEncryption::Rsa::RSA_64);
    /// Шифровка сообщения публичным ключомполучателя
         auto  encryptMessage  =  e.encode(signedMessage,  pub_key,
QRSAEncryption::Rsa::RSA_64);
    return encryptMessage;
} /// sign

/* ____________ end of mainwindow.cpp ____________________________ */
```

Файл «settings.h»

```
/*_________________ settings.h ____________________________________ */

/**
    \file       settings.h
    \brief      Приложение для работы по протоколу UDP.
    \author     Грушевский А.А.
    \version    1.0
    \date       17.05.2020
    \note       Программа реализует возможность приёма и передачи
                данных протоколу UDP.
*/

/* ____________________________________________________________ */

#ifndef SETTINGS_H
#define SETTINGS_H

/* ____________________________________________________________ */

#include <QNetworkInterface>
#include "mainwindow.h"
#include <QTextBrowser>
#include <QHostAddress>
#include <QMessageBox>
#include <QException>
#include <QLineEdit>
#include <QSpinBox>
#include <QDialog>

/* ____________________________________________________________ */

namespace Ui {
class settings;
}

/* ____________________________________________________________ */
```

```
class settings : public QDialog
{
    Q_OBJECT

public:

    explicit settings(MainWindow *parent = nullptr,
                      QLineEdit *lineEdit_localIP = nullptr,
                      QLineEdit *lineEdit_localPort = nullptr,
                      QLineEdit *lineEdit_recvIP = nullptr,
                      QSpinBox *spinBox_recvPort = nullptr);
    ~settings();

private slots:

    /* ____________________________________________________________ */

    /**
        \fn         void on_pushButton_ok_clicked()
        \brief      Кнопка "Ок".
        \note       Закрывает окно настроек и передаёт в главную
                    форму новые параметры.
    */

    void on_pushButton_ok_clicked();

    /* ____________________________________________________________ */

private:

    Ui::settings *ui;
    MainWindow *parentWind;

    /* ____________________________________________________________ */

    /**
        \fn         void settings::getIP()
        \brief      Функция получения локальных IP-адресов.
        \note       Необходима для выявления IP-адресов машины.
    */

    void getIP();

    /* ____________________________________________________________ */

    /**
        \fn         bool settings::validData()
        \brief      Функция проверки всех введённых данных.
        \note       Проверяет, введены ли IP-адреса.
    */
```

```
    bool m_bValidData();

    /* _____________________________________________________________ */

};

/* _____________________________________________________________ */

#endif /// SETTINGS_H

/*     _____________________________      end      of      settings.h
_____________________________ */
```

Файл «settings.cpp»

```
/*_________________ settings.cpp _____________________________ */

/**
    \file       settings.cpp
    \brief      Приложение для работы по протоколу UDP.
    \author     Грушевский А.А.
    \version    1.0
    \date       17.05.2020
    \note       Программа реализует возможность приёма и передачи
                данных по протоколу UDP.
*/

/* _____________________________________________________________ */

#include "ui_settings.h"
#include "settings.h"

/* _____________________________________________________________ */

settings::settings(MainWindow *parent,
                   QLineEdit *lineEdit_localIP,
                   QLineEdit *spinBox_localPort,
                   QLineEdit *lineEdit_recvIP,
                   QSpinBox *spinBox_recvPort) :
    QDialog(parent),
    ui(new Ui::settings)
{
    parentWind = parent;
    ui->setupUi(this);

    /// Получение списка локальных IP-адресов
    getIP();

      /// Установка текущих параметров в полях при открытии окна
настроек
```

```
            ui->comboBox_localIP->setCurrentText(lineEdit_localIP-
>text());
                ui->spinBox_localPort->setValue(spinBox_localPort-
>text().toInt());
    ui->lineEdit_recvIP->setText(lineEdit_recvIP->text());
                  ui->spinBox_recvPort->setValue(spinBox_recvPort-
>text().toInt());

       /// Установка IP-валидатора на поле для ввода IP-адреса
приёмника
       QString ipRange = "(?:[1-9]?[0-9]|1[0-9][0-9]|2[0-4][0-9]|
25[0-5])";
    QRegExp ipRegex ("^" + ipRange
                     + "\\."+ ipRange
                     + "\\."+ ipRange
                     + "\\."+ ipRange + "$");
     QRegExpValidator *ipValidator = new QRegExpValidator(ipRegex,
this);
    ui->lineEdit_recvIP->setValidator(ipValidator);
}

/* _____________________________________________________________________ */

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

/* _____________________________________________________________________ */

/**
    \fn         void settings::on_pushButton_ok_clicked()
    \brief      Кнопка "Ок".
    \note       Закрывает окно настроек и передаёт в главную форму
                новые параметры.
*/

void settings::on_pushButton_ok_clicked()
{
    /// Если введены корректные данные
    if (m_bValidData())
    {
        /// Передача новых параметров в главную форму программы
        parentWind->setSettings(
                    ui->comboBox_localIP->currentText(),
                    ui->spinBox_localPort->text(),
                    ui->lineEdit_recvIP->text(),
                    ui->spinBox_recvPort->text());
        /// Закрытие окна настроек
        this->close();
    }
```

```
}

/* _____________________________________________________________ */

/**
    \fn         void settings::getIP()
    \brief      Функция получения локальных IP-адресов.
    \note       Необходима для выявления IP-адресов машины.
*/

void settings::getIP()
{
    /// Создание списка с сетевыми интерфейсами
                    QList<QNetworkInterface>    allInterfaces    =
QNetworkInterface::allInterfaces();
    QNetworkInterface eth;
    foreach(eth, allInterfaces)
    {
                        QList<QNetworkAddressEntry>   allEntries   =
eth.addressEntries();
        QNetworkAddressEntry entry;
        foreach (entry, allEntries)
        {
            /// Если IP-адрес соответствует протоколу IPv4
                                    if   ((entry.ip().protocol()   ==
QAbstractSocket::IPv4Protocol)
                     && (entry.ip() != (QHostAddress::LocalHost)))
            {
                 /// Добавление нового сетевого интерфейса в список
на форме
                                                ui->comboBox_localIP-
>addItem(entry.ip().toString());
            }
        }
    }
}

/* _____________________________________________________________ */

/**
    \fn        bool settings::validData()
    \brief     Функция проверки всех введённых данных.
    \note      Проверяет, введены ли IP-адреса.
*/

bool settings::m_bValidData()
{
    /// Если локальный IP-адрес не указан
    if (ui->comboBox_localIP->currentText() == "")
    {
        QMessageBox::warning(this,
```

```
                                "Ошибка",
                                    "Необходимо указать локальный IP-
адрес!");
        return 0;
    }

    /// Если IP-адрес получателя не указан
    if (ui->lineEdit_recvIP->text() == "")
    {
        QMessageBox::warning(this,
                                "Ошибка",
                                        "Необходимо указать IP-адрес
получателя!");
        return 0;
    }

    /// В случае корректности всех введённых данных
    return 1;
}

/* _____________ end of settings.cpp ____________________________ */
```

Файл «about.h»

```
/*____________________ about.h ____________________________________ */

/**
    \file       about.h
    \brief      Приложение для работы по протоколу UDP.
    \author     Грушевский А.А.
    \version    1.0
    \date       17.05.2020
    \note       Программа реализует возможность приёма и передачи
                данных по протоколу UDP.
*/

/* ____________________________________________________________ */

#ifndef ABOUT_H
#define ABOUT_H

/* ____________________________________________________________ */

#include <QDialog>

/* ____________________________________________________________ */

namespace Ui {
class about;
}
```

```
/* ____________________________________________________________ */

class about : public QDialog
{

    Q_OBJECT

public:

    explicit about(QWidget *parent = nullptr);

    ~about();

private slots:

    /* ____________________________________________________________ */

    /**
       \fn          void on_pushButton_ok_clicked()
       \brief       Кнопка "Ок".
       \note        Закрывает окно "О программе".
    */

    void on_pushButton_ok_clicked();

    /* ____________________________________________________________ */

private:

    Ui::about *ui;

};

/* ____________________________________________________________ */

#endif /// ABOUT_H

/*_________________ end of about.h ____________________________ */
```

Файл «about.cpp»

```
/*_____________ about.cpp ____________________________________ */

/**
    \file       about.cpp
    \brief      Приложение для работы по протоколу UDP.
    \author     Грушевский А.А.
    \version    1.0
    \date       17.05.2020
    \note       Программа реализует возможность приёма и передачи
                данных по протоколу UDP.
```

```
*/

/* _____________________________________________________________ */

#include "about.h"
#include "ui_about.h"

/* _____________________________________________________________ */

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

/* _____________________________________________________________ */

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

/* _____________________________________________________________ */

/**
   \fn         void about::on_pushButton_ok_clicked()
   \brief      Кнопка "Ок".
   \note       Закрывает окно "О программе".
*/

void about::on_pushButton_ok_clicked()
{
    this->close();
} /// on_pushButton_ok_clicked

/* _____________ end of about.cpp ____________________________ */
```

Файл «main.cpp»

```
/*_____________________ main.cpp ____________________________ */

/**
    \file       main.cpp
    \brief      Приложение для работы по протоколу UDP.
    \author     Грушевский А.А.
    \version    1.0
    \date       17.05.2020
    \note       Программа реализует возможность приёма и передачи
                данных по протоколу UDP.
*/

/* _____________________________________________________________ */
```

```
#include "mainwindow.h"
#include <QApplication>

#include <QCoreApplication>
#include <iostream>
#include <iomanip>

/* ____________________________________________________________ */

/**
   \fn              int main(int argc, char *argv[])
   \brief      Главная функция программы.
   \note       Сгенерирована автоматически.
*/

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    /// Отключение кнопки с вопросительным знаком во всех окнах
     QApplication::setAttribute(Qt::AA_DisableWindowContextHelpButt
on);

    MainWindow w;
    w.show();

    return a.exec();
}

/* _________________ end of main.cpp ____________________________ */
```

Приложение В. Презентация

Тема работы: «ПО приложения для передачи конфиденциальных данных по протоколу UDP»

АВТОР РАБОТЫ: СТУДЕНТ ГР. 16ВВ1 ГРУШЕВСКИЙ АЛЕКСАНДР АНАТОЛЬЕВИЧ

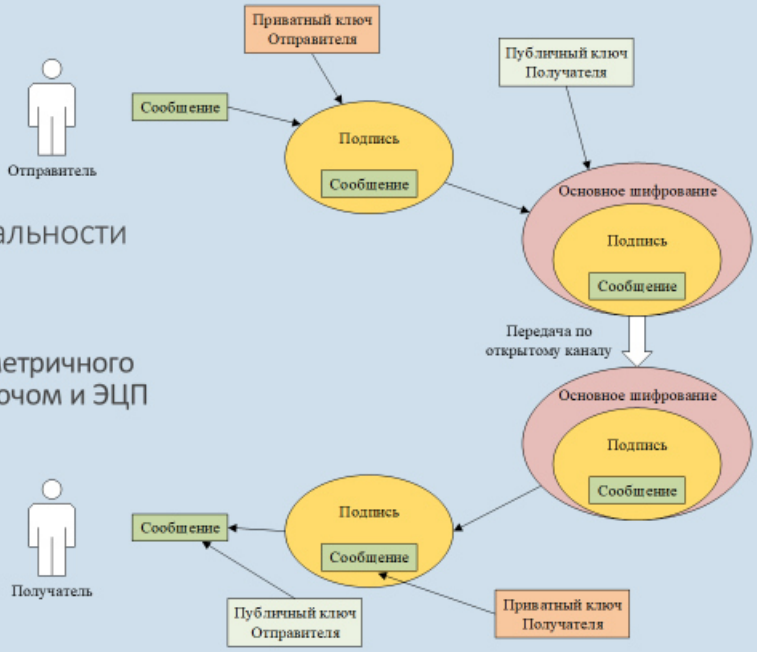
НАУЧНЫЙ РУКОВОДИТЕЛЬ: Д.Т.Н., ПРОФЕССОР КАФ. ВТ ПГУ ГУРИН Е.И.

Постановка задачи

- Требуется разработать приложение, предназначенное для передачи и приёма конфиденциальных данных по протоколу UDP.
- Для передачи данных по открытым каналам в приложении необходимо предусмотреть использование алгоритма асимметричного шифрования и цифровой подписи.
- Приложение должно иметь удобный пользовательский интерфейс, понятный на интуитивном уровне.
- Пользователь должен иметь возможность изменять как IP-адрес и порт получателя, так и собственные параметры.

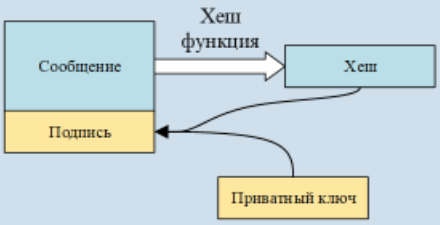
Соблюдение конфиденциальности передаваемых данных

Схема использования асимметричного шифрования с открытым ключом и ЭЦП

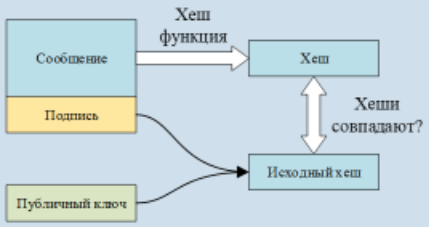


Схемы генерации и проверки подписи

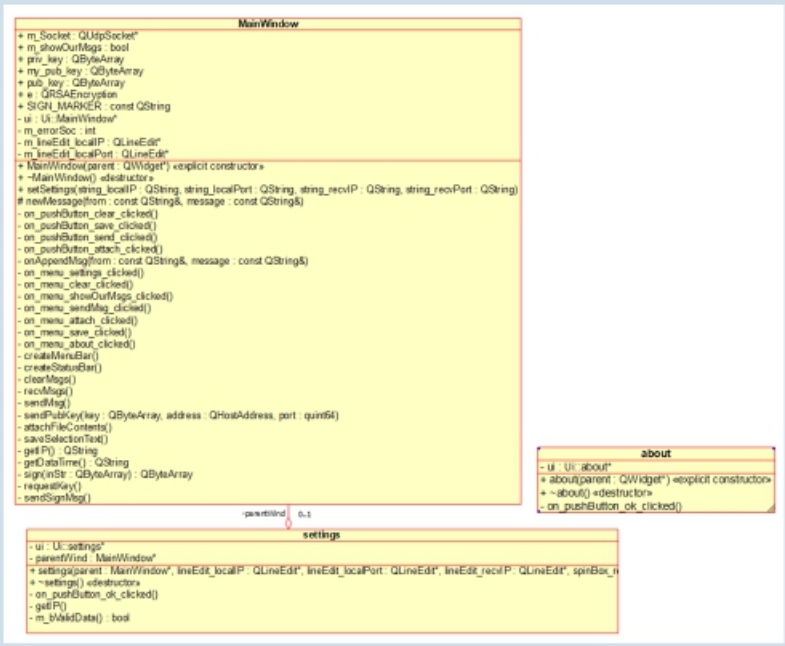
Генерация подписи



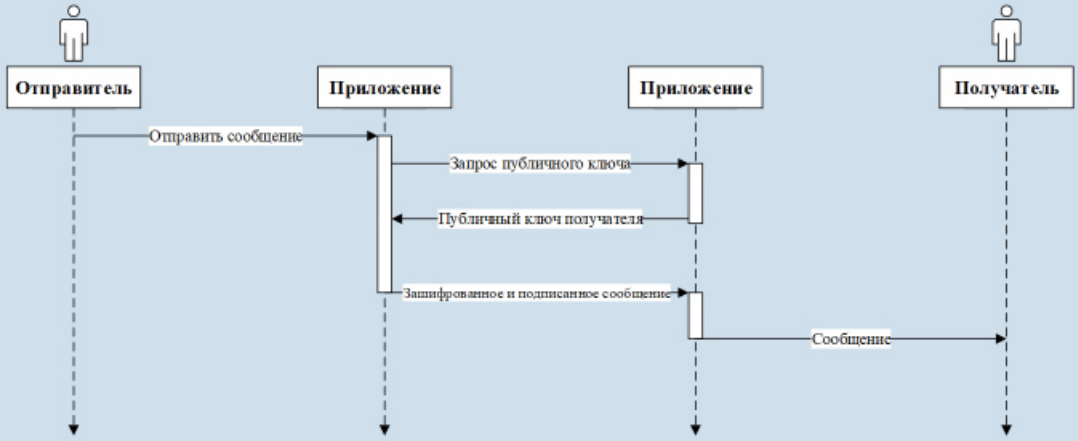
Проверка подписи



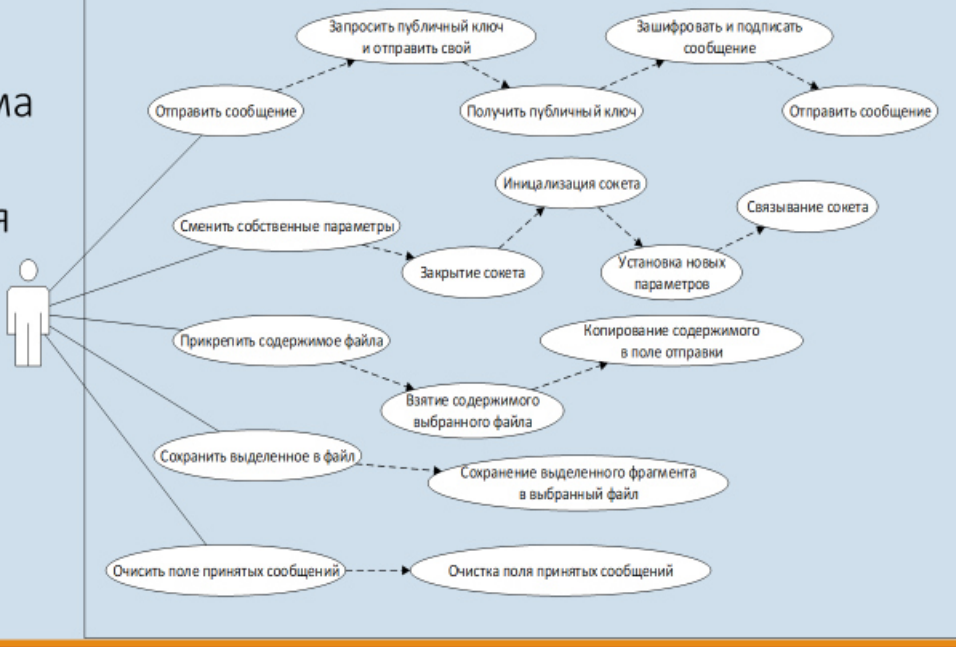
UML-диаграмма классов



UML-диаграмма последовательности

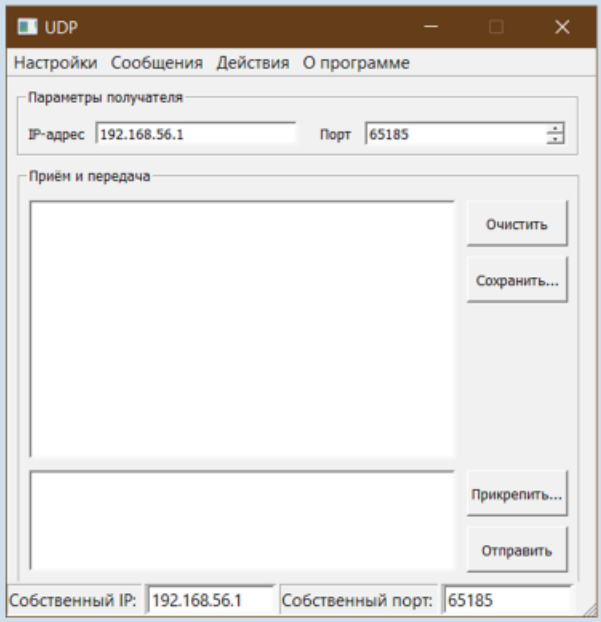


UML-диаграмма вариантов использования

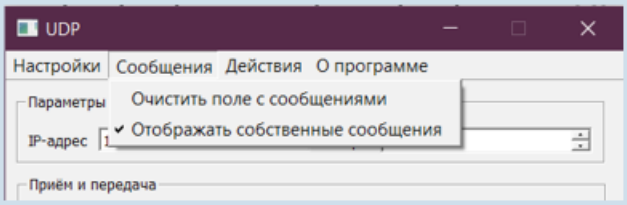


UML-диаграмма деятельности

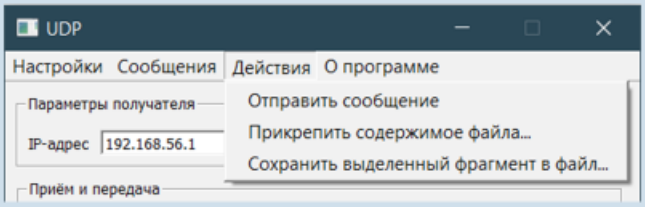
Главное окно приложения



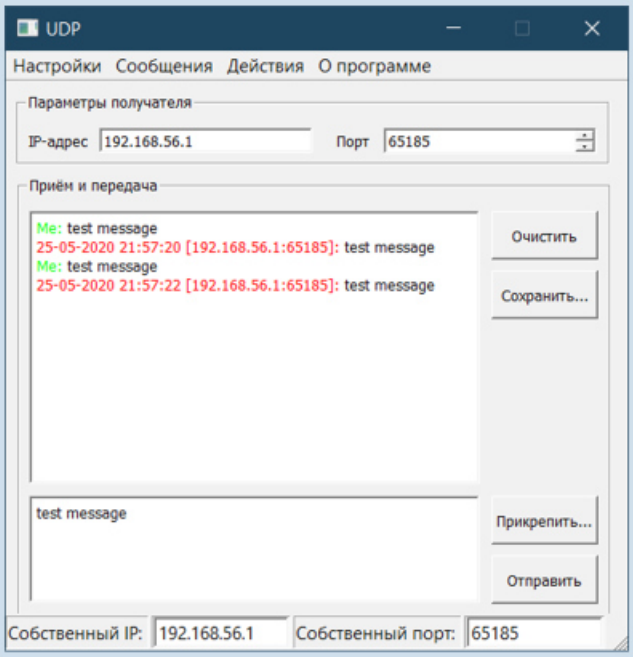
Включение отображения собственных сообщений



Вкладки меню «Действия»



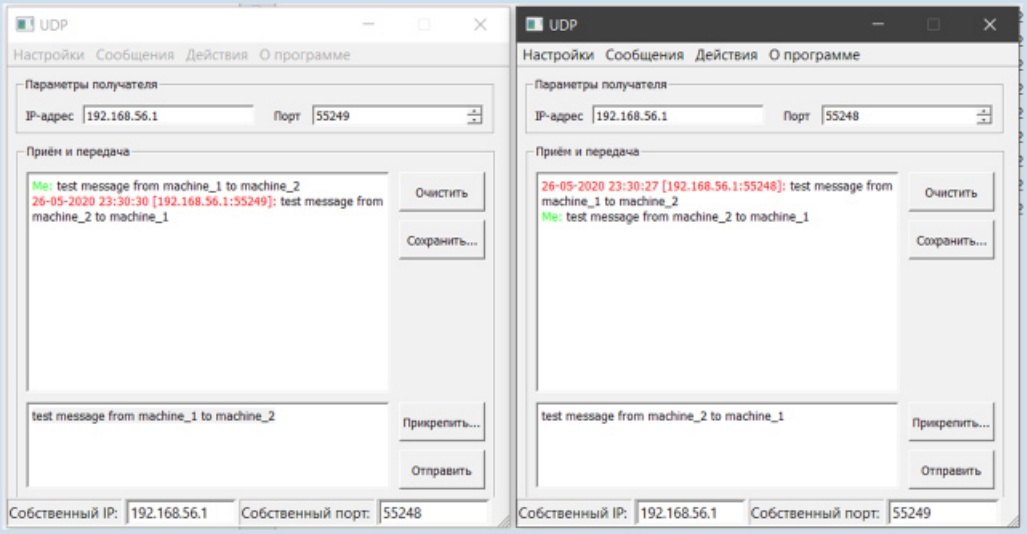
Отправка и получение сообщения



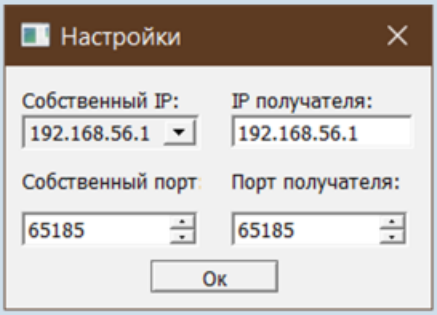
Типы передаваемых в приложении дейтаграмм

Дейтаграмма	
Тип дейтаграммы (первый байт)	Тело дейтаграммы
‘0’	Публичный ключ отправителя
‘1’	Публичный ключ получателя
‘2’	Сообщение

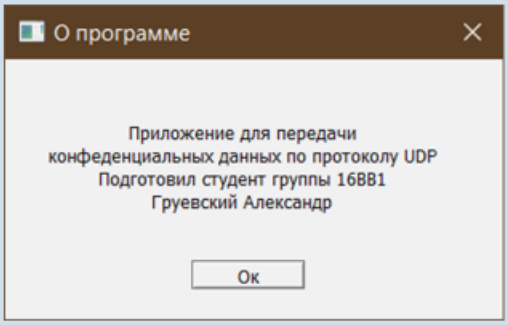
Обмен сообщениями между программами



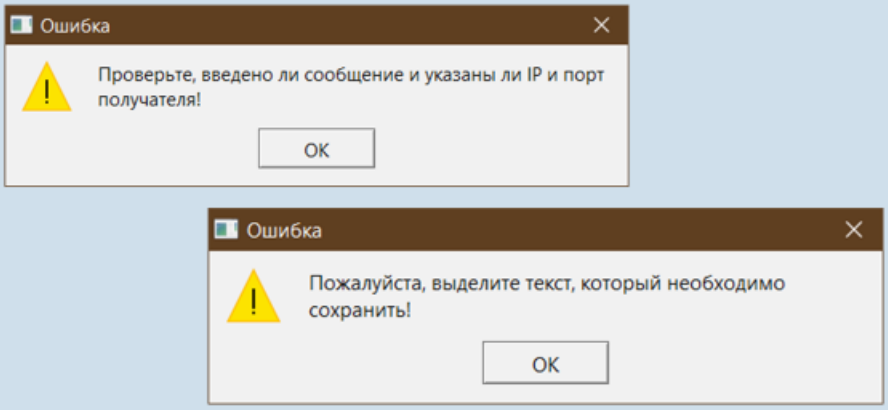
Окно настроек



Окно «О программе»



Обработка исключительных случаев



Заключение

- Результатом решения поставленной задачи можно считать разработанное и протестированное приложение для обмена конфиденциальными данными с использованием протокола UDP. Разработанной приложение позволяет пользователю без страха передавать важную информацию по открытым каналам связи благодаря использованию алгоритмов RSA для шифрования и подписывания данных.
- В ходе выполнения данной выпускной квалификационной работы были изучены способы шифрования данных, способы сетевых взаимодействий на уровне транспортного уровня модели OSI и стека протоколов TCP/IP, также были получены навыки в области программирования на языке C++ на платформе Qt, навыки в разработке пользовательского интерфейса.

Благодарю за внимание!