МИНИСТЕРСТВО образования Кировской области

Кировское областное государственное профессиональное образовательное бюджетное учреждение

"Слободской колледж педагогики и социальных отношений"

ДИПЛОМНЫЙ ПРОЕКТ

**Разработка программного обеспечения ДЛЯ ЗАЩИТЫ ИНФОРМАЦИИ С ИСПОЛЬЗОВАТЕНИЕМ КРИПТОГРАФИЧЕСКИХ МЕТОДОВ**

Выполнил: Катаргин Никита Андреевич

Специальность 09.02.07

Информационные системы и программирование

Группа 20П-1

Форма обучения: очная

Руководитель: Пентин

Николай Сергеевич

Дипломный проект защищен

"\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Секретарь ГЭК\_\_\_\_\_\_\_\_\_\_

Слободской

2024

ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc168043927)

[ВВЕДЕНИЕ 3](#_Toc168043928)

[ГЛАВА 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 6](#_Toc168043929)

[1.1. Анализ предметной области 6](#_Toc168043930)

[ВЫВОД ПО ГЛАВЕ 1 13](#_Toc168043931)

[ГЛАВА 2. КОНСТРУКТОРСКАЯ ЧАСТЬ 14](#_Toc168043932)

[2.1. Архитектура программы 14](#_Toc168043933)

[2.2. Описание алгоритмов и функционирования программы 16](#_Toc168043934)

[ВЫВОД ПО ГЛАВЕ 2 19](#_Toc168043935)

[ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНО-ПРИКЛАДНАЯ ЧАСТЬ 20](#_Toc168043936)

[3.1. Тестирование и опытная эксплуатация программы 20](#_Toc168043938)

[3.2. Руководство оператора для портативного приложения 25](#_Toc168043939)

[3.3. Руководство оператора для серверного приложения 30](#_Toc168043940)

[ВЫВОД ПО ГЛАВЕ 3 36](#_Toc168043941)

[ЗАКЛЮЧЕНИЕ 37](#_Toc168043942)

[СПИСОК ЛИТЕРАТУРЫ 39](#_Toc168043943)

[ПРИЛОЖЕНИЕ 41](#_Toc168043944)

ВВЕДЕНИЕ

В современном информационном обществе защита данных является одной из важнейших проблем. Криптография – это наука об обеспечении конфиденциальности, целостности и аутентичности информации путем применения различных математических и алгоритмических методов.

Суть криптографии заключается в том, чтобы защитить информацию от людей или компаний, для которых она не предназначена. Не имея ключа, человек видит зашифрованные данные в виде неопознанного файла или бессвязного набора символов, поэтому не может понять сообщение. При этом нужный получатель может расшифровать информацию, имея ключ.

C ростом интернета и расширением сферы цифровых технологий, существует все больше угроз информационной безопасности, что делает криптографию актуальной и необходимой. Криптографические алгоритмы и протоколы широко применяются для защиты различных видов информации, таких как персональные данные, финансовые транзакции, коммуникации и многое другое.

Обмен и обработка данных стали неотъемлемой частью повседневной деятельности, вопрос обеспечения безопасности информации становится крайне актуальным. В этом контексте разработка программного обеспечения, способного эффективно защищать конфиденциальность и целостность данных, становится ключевым направлением развития информационных технологий. Криптографические методы играют важную роль в создании надежных систем защиты, позволяя обеспечить непроницаемую оболочку вокруг ценных информационных ресурсов.

В рамках проекта планируется реализовать несколько основных функций. Первая функция включает в себя возможность шифрования и дешифрования информации с использованием RSA (Rivest-Shamir-Adleman). Вторая функция предоставляет возможность отправки сообщений и файлов в зашифрованном виде. Третья функция включает в себя реализацию протоколов обмена ключами для установления защищенного канала связи.

Результатом данной работы будет готовое программное обеспечение, позволяющее пользователям безопасно отправлять и получать информацию, шифровать и дешифровать данные. Это программное обеспечение будет иметь графический интерфейс, удобный и интуитивно понятный для пользователей. Планируется создание пошаговых инструкций и встроенной помощи, которые помогут пользователям быстро освоить функциональность программного обеспечения.

Разработка программного обеспечения криптографии является актуальной и практически значимой задачей. Обеспечение информационной безопасности становится все более важным в современном мире, и данное программное обеспечение может быть полезно для широкого круга пользователей, включая организации и индивидуальных пользователей, которые нуждаются в защите своих данных.

Так же мной были изучены программы аналоги такие как:

* 1. Pretty Good Privacy:

Плюсы: Pretty Good Privacy предоставляет сильное шифрование с открытым исходным кодом и имеет долгую историю использования в обеспечении конфиденциальности коммуникаций.

Минусы: несмотря на свою мощь и гибкость, Pretty Good Privacy может быть сложным в использовании для неопытных пользователей, особенно в настройке ключей и управлении сертификатами.

* 1. Veracrypt:

Плюсы: Veracrypt позволяет создавать зашифрованные тома и контейнеры для хранения файлов, обеспечивая высокий уровень безопасности данных на диске.

Минусы: для использования некоторых функций может потребоваться некоторое время на изучение документации, так как управление шифрованием может быть сложным для новичков.

* 1. Signal:

Плюсы: Signal предоставляет шифрование сообщений, аудио и видео вызовов с открытым исходным кодом, а также широко признан за свою простоту использования и высокий уровень безопасности.

Минусы: Signal работает на основе номеров телефонов, что может быть неудобно для пользователей, желающих сохранить анонимность.

Основаниями для разработки послужили:

1. Актуальность выбранной темы
2. Цель разработки является защита файлов, и сообщений, передаваемых по каналам связи и распространяемых в сети Интернет, путем разработки системы c использованием криптографических методов.

Задачи разработки:

* Описать предметную область.
* Разработать технического задание на создание программного продукта.
* Описать архитектуру программы.
* Описать алгоритмы и функционирование программы.
* Провести тестирование и опытную эксплуатацию.
* Разработать руководство оператора

Объект разработки – процесс шифровки и расшифровки файлов.

Предмет разработки – разработка программной системы криптографии.

Методы разработки: системный анализ и функциональное моделирование.

ГЛАВА 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

* 1. Анализ предметной области

**Что такое криптография?**

Криптография — это наука о защите информации, которая изучает методы и техники шифрования данных для обеспечения их конфиденциальности, целостности и аутентичности.

**Основные принципы работы шифрования:**

Открытый текст (Plaintext): это исходный текст, который нужно зашифровать. Например, сообщение или данные.

Шифр (Cipher): это алгоритм, который применяется к открытому тексту для его шифрования. Он использует ключ для создания шифрованного текста из открытого.

Ключ (Key): Ключ – это параметр, который влияет на то, как работает шифр. Его используют для шифрования и дешифрования данных. Ключ может быть секретным (симметричное шифрование) или открытым (асимметричное шифрование).

В зависимости от того, используется ли один и тот же ключ для шифрования и дешифрования (симметричное) или разные ключи (асимметричное), существует два основных типа шифрования:

Симметричное шифрование: здесь один и тот же ключ используется как для шифрования, так и для дешифрования данных. Примеры включают DES, AES и IDEA.

Асимметричное шифрование: здесь используется пара ключей – открытый и закрытый. Открытый ключ используется для шифрования, а закрытый – для дешифрования. RSA и ECC – примеры асимметричных алгоритмов.

Процесс шифрования может быть представлен следующим образом:

Выбор алгоритма и ключа: Открытый текст и ключ передаются в алгоритм шифрования.

Шифрование: Алгоритм использует ключ для трансформации открытого текста в шифротекст.

Передача шифротекста: Шифротекст может быть безопасно передан по открытым каналам связи.

Дешифрование: Получатель использует ключ дешифрования, чтобы вернуть шифротекст в исходный открытый текст.

Важными аспектами шифрования являются стойкость к взлому (невозможность получить открытый текст без ключа) и безопасность ключа. Шифрование играет ключевую роль в обеспечении конфиденциальности данных в современных системах информационной безопасности.

**Основные функции программы по криптографии:**

1. Шифрование информации;
2. Дешифрование информации;
3. Передача информации;
4. Генерация ключей
5. Подключение к серверу

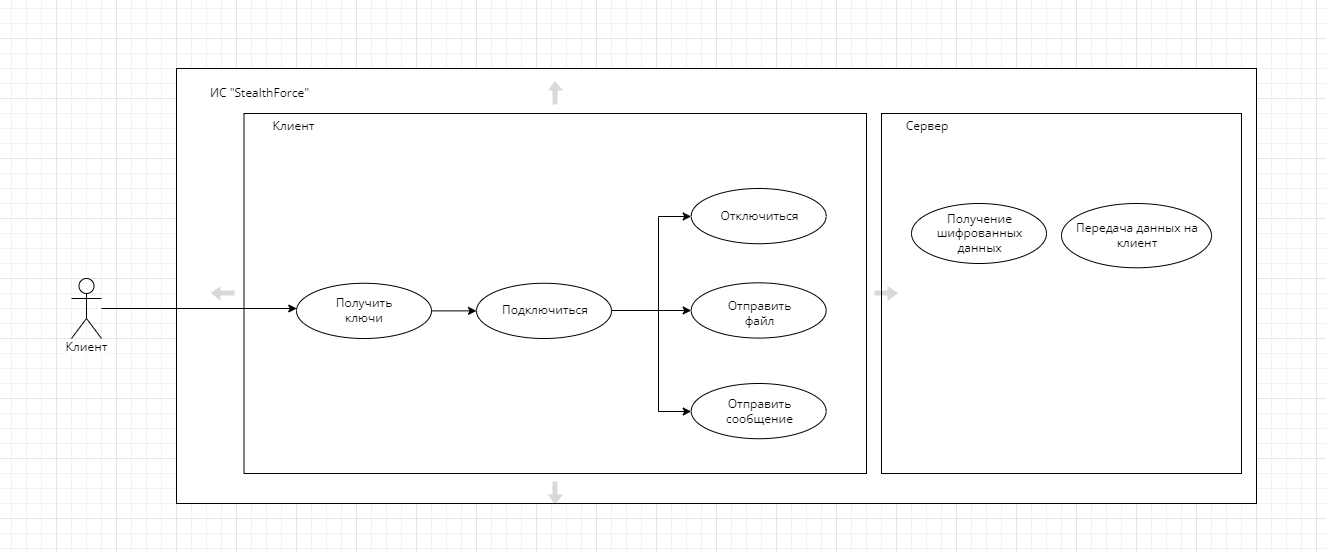


Рисунок 1.1 - Диаграмма вариантов использования

Следует отдельно описать вышеперечисленные функции (Рисунок 1.1).

* + 1. **Функция шифрование информации**

Описание: Эта функция позволяет зашифровать передаваемую информацию с использованием определенного алгоритма шифрования и используя публичный ключ (открытый).

Входные параметры: Передаваемая информация, ключ шифрования.

Выходные параметры: Зашифрованная информация.

* + 1. **Функция дешифрование информации**

Описание: Эта функция позволяет расшифровать зашифрованную информацию с использованием того же алгоритма шифрования и используя секретный (приватный).

Входные параметры: Зашифрованная информация, ключ дешифрования.

Выходные параметры: Расшифрованная информация.

* + 1. **Функция передачи информации**

Описание: Эта функция отправляет файлы или сообщения на серевер для передачи их другому клиенту.

Входные параметры: Файл или сообщение.

Выходные параметры: Зашифрованная информация.

* + 1. **Функция генерация ключей**

Описание: Эта функция генерирует уникальные ключи для шифрования и дешифрования информации.

Входные параметры: Нет.

Выходные параметры: Ключ шифрования, ключ дешифрования.

* + 1. **Функция Подключение к серверу**

Описание: Эта функция позволяет клиенту с помощью уникальных ключей подключиться к другому клиенту через сервер для шифрования и дешифрования информации.

Входные параметры: Публичные ключи.

Выходные параметры: Подключение к серверу.

* 1. **Техническое задание**

Наименование программы – «StealthForce». Программа предназначена для передачи файлов и сообщений в зашифрованном виде между 2 клиентами.

Разработка программы ведется на основании дипломной работы.

Функциональным назначением программы является обмен информацией между клиентами с использованием зашифрованного канала передачи данных с протоколом шифрования RSA.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* Отправлять файлы, документы и изображения.
* Шифровать и расшифровывать сообщения.
* Генерировать ключи шифрования.
* Подключаться к серверу для передачи данных.

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

* организация бесперебойного питания технических средств;
* использование лицензионного программного обеспечения;
* отсутствие вредоносного программного обеспечения, наличие антивирусной программы;
* соблюдение правил и требований по эксплуатации технических средств.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 5 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

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

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий себя:

* процессор с тактовой частотой, 1 ГГц, не менее;
* оперативную память объемом 512 Мб, не менее;
* жесткий диск со свободным местом 500 Мб, не менее;
* монитор, с разрешением экрана 1024\*768, не менее;
* оптический привод;
* компьютерная мышь;
* клавиатура;

Исходные коды программы должны быть реализованы на языке С#. В качестве интегрированной среды разработки программы должны быть использованы среды программирования Microsoft Visual Studio 2022.

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционных систем Windows 7/8/10/11.

Программное обеспечение поставляется в виде изделия на CD диске.

Упаковка программного изделия должна осуществляться в упаковочную тару предприятия-изготовителя компакт диска.

Требования к транспортировке и хранению должны соответствовать условиям эксплуатации носителей, на которых находится программный продукт.

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса.

Предварительный состав программной документации включает в себя следующие документы:

* техническое задание;
* руководство оператора.

Разработка должна быть проведена в следующие стадии и этапы:

1. Анализ требований:

На стадии анализ требований формулируются цели и задачи проекта. Создается основа для дальнейшего проектирования

1. Проектирование:

На стадии проектирование должны быть выполнены перечисленные ниже этапы работ:

* разработка программной документации;

На этапе разработка программной документации должна быть выполнена разработка технического задания.

При разработке технического задания должны быть выполнены перечисленные работы: постановка задачи, определение и уточнение требований к техническим средствам, определение требований к программе, определение стадий, этапов и сроков разработки программы и документации на нее, выбор языков программирования.

* разработка алгоритма программы;

На этапе разработки алгоритма программы должен быть разработан алгоритм работы программы.

* кодирование;

На стадии кодирования происходит реализация алгоритмов в среде программирования.

* тестирование и отладка.

На стадии тестирование и отладка происходит проверка алгоритмов, реализованных в программе на работоспособность в различных ситуациях. Исправление выявленных ошибок, повторное тестирование.

Приемо-сдаточные испытания должны проводиться при использовании технических средств. Приемка программы заключается в проверке работоспособности программы путем ввода реальных или демонстрационных данных.

Во время приемки работы разработчик предоставляет программу и документацию, которая к ней прилагается. Проводятся испытания программы, при успешных испытаниях программа вводится в эксплуатацию. При ошибках, недопустимых для успешной работы программного продукта – отправляется на доработку.

Было описано техническое задание, содержащее в себе информацию о программном продукте, его функциях, эксплуатации и требования, которые должны учитываться при создании программы и документации к ней.

ВЫВОД ПО ГЛАВЕ 1

В этой главе была описана аналитическая часть, содержащая в себе описание функций программы, а также техническое задание, содержащее в себе информацию о программном продукте, его функциях, эксплуатации и требования, которые должны учитываться при создании программы и документации к ней.

Определена значимость и требования к будущему программному обеспечению. В техническом задании были определены основные требования к программному продукту и функциональные характеристики, а также состав программной документации.

ГЛАВА 2. КОНСТРУКТОРСКАЯ ЧАСТЬ

* 1. Архитектура программы

Данная программа представляет собой инструмент передачи информации между двумя клиентами в зашифрованном виде.

В состав программного обеспечения входят следующие файлы:

* Портативное приложение клиента и сервера (файлы setup.exe);

Портативное приложение, именуемое «StealthForce.Client», представляет собой оконное приложение, написанное на языке программирования C# с использованием WPF на платформе Microsoft Visual Studio 2022 и работает на операционной системе Windows. Архитектура приложения построена на основе паттерна "Model-View-ViewModel" (MVVM). В приложении реализованы следующие функции: генерация ключей, подключение к серверу, отправка сообщений и файлов, обработка информации на сервере, функции шифрования и дешифрования.

Главное окно программы, состоящее из функций, реализованных в программе (Рисунок 2.1):

1. Получение публичных ключей, для подключения к серверу;
2. Подключение к серверу по средству обмена ключей;
3. Отправка сообщения в зашифрованном виде;
4. Отправка файла в зашифрованном виде;

Весь код программы описан в разделе приложений (Приложение 1).

Программа имеет современный и интуитивно понятный пользовательский интерфейс, что облегчает работу пользователям. Основное окно приложения содержит меню и кнопки для выполнения основных операций. В приложении предусмотрена возможность настройки параметров подключения и шифрования, что делает его гибким и адаптируемым под различные условия эксплуатации.

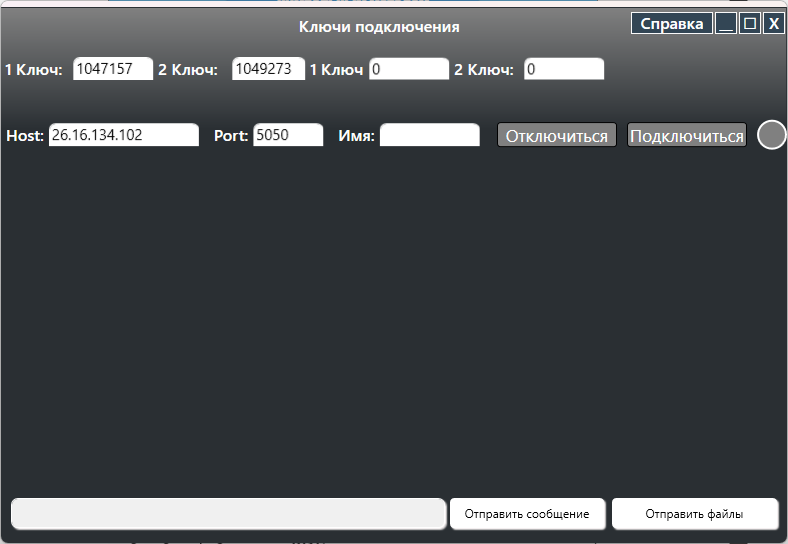


Рисунок 2.1 – Главное окно

Серверное приложение, именуемое «StealthForce.Server», предназначено для подключения пользователей между собой и их обменом информации. Оно написано на языке программирования С# с использованием консольного приложения на платформе Microsoft Visual Studio 2022 и работает на операционной системе Windows. В приложении реализованы следующие функции: обработка новых подключений, отправка файлов и сообщений в зашифрованном виде клиенту для дальнейшей работы с ним, вывод зашифрованных сообщений.

Серверное окно программы, состоящее из функций, реализованных в программе (Рисунок 2.2):

1. Обработка новых подключений на сервер;
2. Обработка сообщений клиентов и их передача;
3. Обработка файлов клиентов и их передача;
4. Вывод в зашифрованном виде сообщений.

Серверное приложение имеет структуру, позволяющую легко масштабировать систему при увеличении количества клиентов. В архитектуре серверного приложения используется многопоточность для обеспечения одновременной обработки множества подключений, что улучшает производительность и устойчивость к нагрузкам. Весь код программы описан в разделе приложений (Приложение 2).

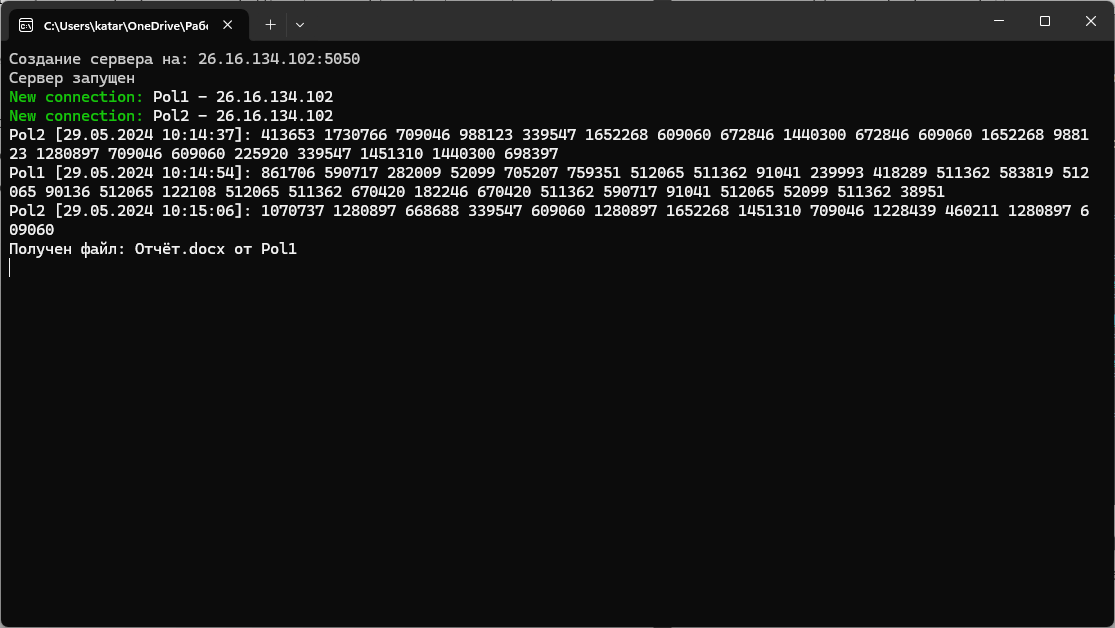


Рисунок 2.2 – Серверное окно программы

Кроме того, в серверное приложение встроены механизмы логирования и мониторинга, позволяющие отслеживать состояние системы и выполнять диагностику при возникновении проблем. Это обеспечивает высокую надежность и удобство в эксплуатации программы.

* 1. Описание алгоритмов и функционирования программы

**Функционирование программы**

При запуске программы должны быть обеспечены соответствующие права для чтения и записи файлов, а также доступ к сети интернет, у пользователя доступ к используемым файлам в файловой системе.

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

* организация бесперебойного питания технических средств;
* использование лицензионного программного обеспечения;
* отсутствие вредоносного программного обеспечения, наличие антивирусной программы;
* обучение пользователей правилам безопасности и лучшим практикам работы с программным обеспечением.
* соблюдение правил и требований по эксплуатации технических средств.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 5 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

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

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

**Алгоритмы программы**

Рассмотрим алгоритмы, используемые в программе.

* Генерация ключей RSA (Приложение 3)

После запуска приложения выбирается случайное число в диапазоне от 500 до 1500. Для каждого из этих чисел проверяется, является ли оно простым. Проверка производится путем поиска делителей. Если делители не найдены, число считается простым. Этот процесс повторяется для каждого случайного числа до тех пор, пока не будут найдены два различных простых числа. Генерация ключа — Ключ генерируется путем умножения первого простого числа на второе. Полученное произведение становится значением модуля «n». Для каждого простого числа вычисляется значение функции Эйлера. После чего выбирается открытая экспонента «e», которая должна быть простым числом, меньшим, чем вычисленное значение функции Эйлера «fi» Последним шагом генерации ключей, вычисление закрытой экспоненты для дешифрования.

* Подключение к серверу

После запуска приложения пользователь получает ключи, обменявшись, с вторым пользователем ключами, необходимо ввести IP адрес сервера и его порт, а также своё имя в системе. После чего нажав на кнопку «Подключиться» приложение отправляет запрос на подключение к серверу по указанному адресу и порту. Если подключение успешное сервер отправляет подтверждение. В противном случае, сервер сообщает об ошибке. Сообщение об успешном подключении или ошибке. Сообщение об успешном подключении или ошибке отображается в чате пользователю. В этот момент сервер ожидает действий со стороны клиента.

* Шифрование информации

Пользователь вводит текстовое сообщение или выбирает файл для отправки. Приложение при передаче данных шифрует данные с помощью открытого ключа получателя (полученного от сервера при подключении) методом RSA шифрования. Зашифрованные данные (текст или содержимое файла) отправляются на сервер в виде зашифрованного объекта и передаются второму клиенту.

* Дешифрование информации

При получении текстового сообщения или файла клиентом от сервера он использует свой закрытый ключ RSA для расшифровки этих данных. Расшифрованные данные (текст или содержимое файла) отображаются пользователю в чате.

ВЫВОД ПО ГЛАВЕ 2

В результате работы над конструкторской частью была разработана архитектура созданного программного продукта, описаны алгоритмы и функционирование программы.

Для более наглядного понимания архитектуры программы были приведены макеты главного окна портативного и серверного приложения. Каждый макет был описан функциями, реализуемыми в портативном и мобильном приложении.

Так же были описаны алгоритмы приложения, а именно «Цифровой водяной знак», «Текстовый водяной знак», «Водяной знак изображением», «Мультимедийный формат» и «PDF файлы».

ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНО-ПРИКЛАДНАЯ ЧАСТЬ

1. 1. Тестирование и опытная эксплуатация программы

Для проведения тестирования программы мною было произведено базовое тестирование во время прохождения преддипломной практики, которое показало готовность программы к эксплуатации на предприятии.

* Отправка текстового сообщения другому клиенту.

Шаги тестирования:

* 1. Запустить приложение;
  2. Обменяться ключами с вторым клиентом;
  3. Подключиться к пользователю;
  4. Написать сообщение;
  5. Отправить сообщение;
  6. Проверить пришло ли сообщение.

Данные тестирования: Сообщение «Привет!!!», «Здравствуйте».

Ожидаемый результат: Сообщение отправлено, на сервере зашифровано, пользователь получил сообщение в не зашифрованном виде.

Фактический результат: Сообщение отправлено, на сервере зашифровано, пользователь получил сообщение в не зашифрованном виде (Рисунок 3.1).

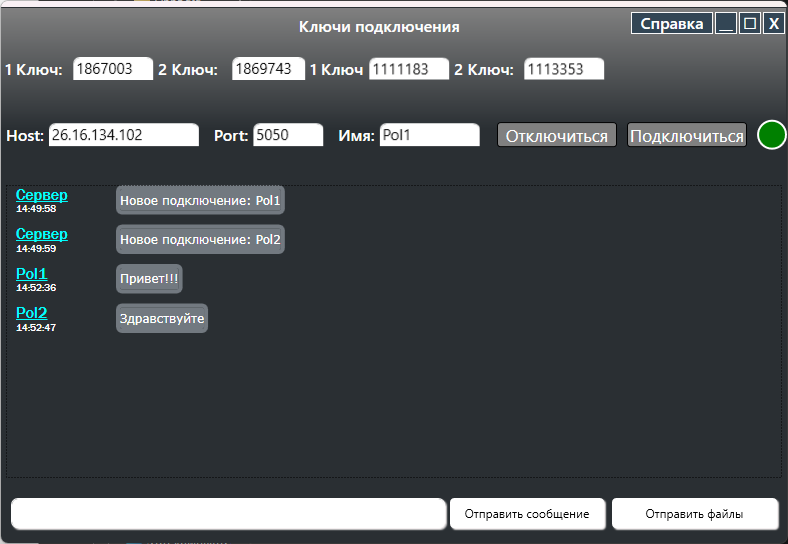


Рисунок 3.1 – Отправка сообщения

* Отправка файла другому клиенту.

Шаги тестирования:

* 1. Запустить приложение;
  2. Обменяться ключами с вторым клиентом;
  3. Подключиться к пользователю;
  4. Нажать на кнопку отправить файл;
  5. Выбрать файл;
  6. Отправить файл;
  7. Проверить пришел ли файл.

Данные тестирования: имя первого клиента -Pol1, имя второго клиента -Pol2, Ключ 1,2 – получен от другого клиента, файл «test.txt».

Ожидаемый результат: файл сохранён на рабочем столе в созданной приложением папке файлы.

Фактический результат: файл сохранён на рабочем столе в созданной приложением папке файлы. (Рисунок 3.2).

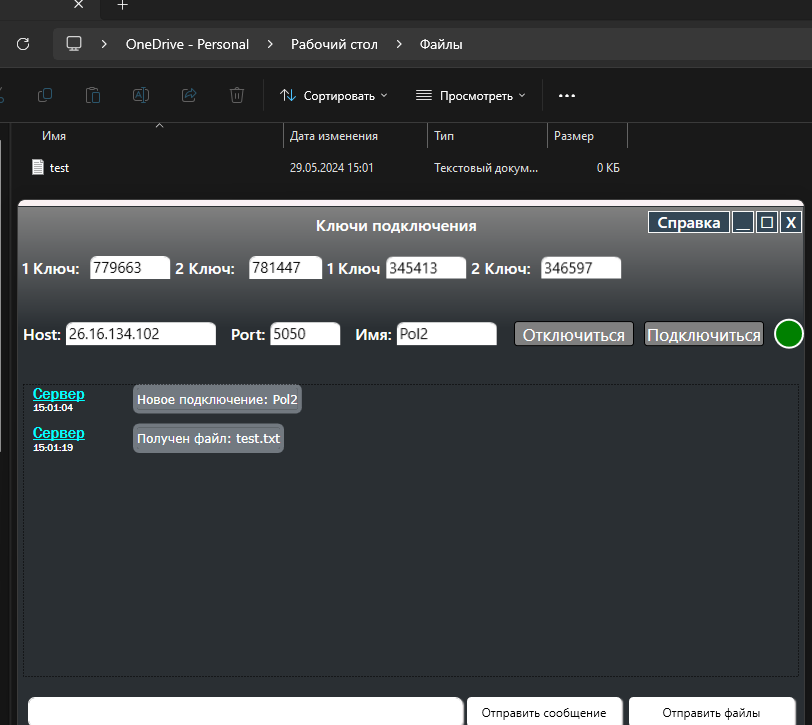


Рисунок 3.2 – Отправка файла

* Подключение к клиенту

Шаги тестирования:

* 1. Запустить приложение;
  2. Нажать на кнопку получить ключи;
  3. Обменяться ключами;
  4. Ввести ключи в необходимые поля;
  5. Ввести имя пользователя;
  6. Нажать на кнопку подключиться.

Данные тестирования: имя первого клиента -Pol1, имя второго клиента -Pol2, Ключ 1,2 – получен от другого клиента.

Ожидаемый результат: вывод cобщения о новом подключении Pol1.

Фактический результат: вывод cобщения о новом подключении Pol1 (Рисунок 3.3).

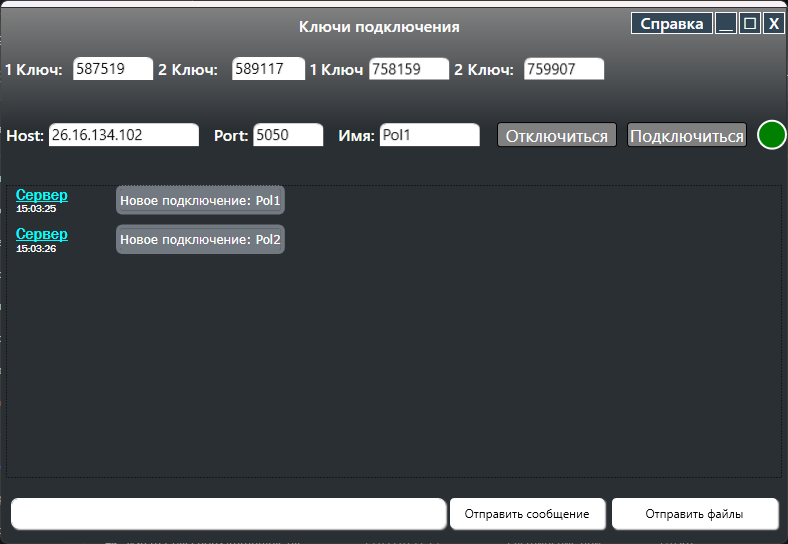


Рисунок 3.3 – Подключение к клиенту

* Вывод подключения пользователя на сервере.

Шаги тестирования:

* 1. Запустить клиентское приложение;
  2. Обменяться ключами с вторым клиентом;
  3. Выполнить подключение;
  4. Ожидание вывода на сервере успешного подключения;

Данные тестирования: имя первого клиента -Pol1, имя второго клиента -Pol2, Ключ 1,2 – получен от другого клиента.

Ожидаемый результат: вывод в серверном окне сообщения о подключении нового пользователя.

Фактический результат: вывод в серверном окне сообщения о подключении нового пользователя. (Рисунок 3.4).

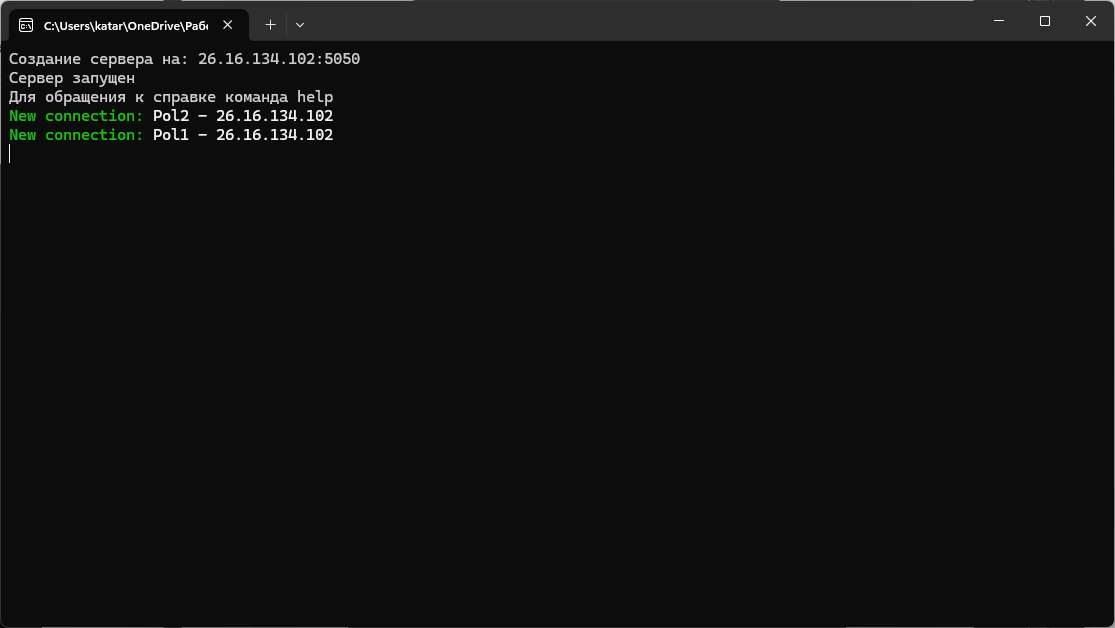


Рисунок 3.4 – Сообщение о новом подключении на сервере

* Вывод зашифрованного текста на сервере.

Шаги тестирования:

* 1. Запустить клиентское приложение;
  2. Обменяться ключами с вторым клиентом;
  3. Выполнить подключение;
  4. Отправить сообщение;
  5. Проверить шифрование на сервере;

Данные тестирования: имя первого клиента -Pol1, имя второго клиента -Pol2, Ключ 1,2 – получен от другого клиента, текст «Привет».

Ожидаемый результат: вывод в серверном окне сообщения с зашифрованным текстом.

Фактический результат: вывод в серверном окне сообщения с зашифрованным текстом.

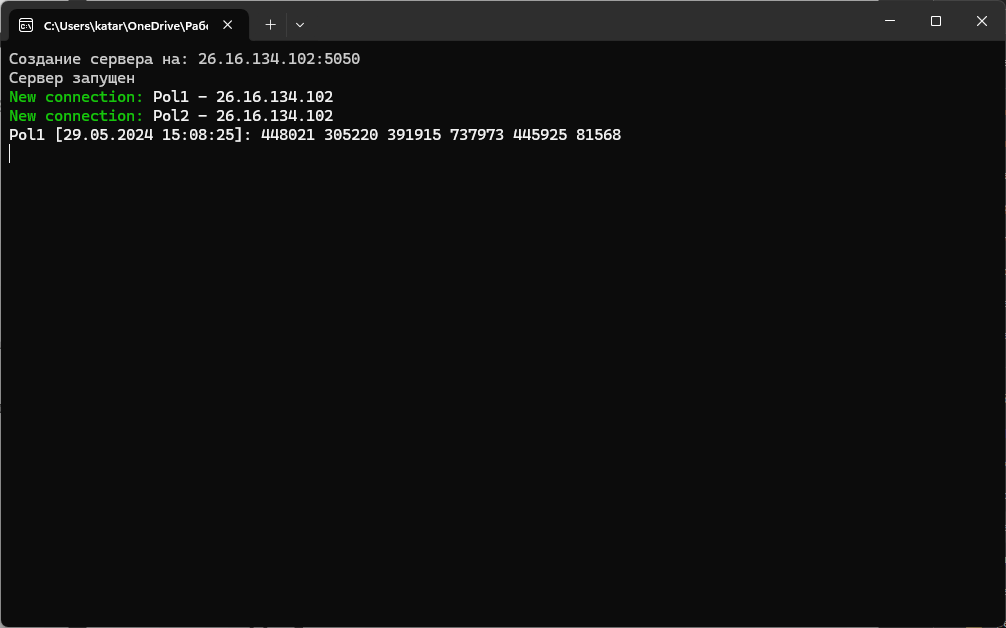


Рисунок 3.5 – Шифрование на сервере

* 1. Руководство оператора для портативного приложения

Функциональным назначением программы является шифрование и защита передаваемой информации.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* Создавать подключения к серверу.
* Оправлять сообщения и файлы любых форматов.
* Шифровать и расшифровывать переданную пользователями информацию.
* Генерировать ключи шифрования и дешифрования.

Условия выполнения программы

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средства в части условий их эксплуатации.

Минимальный состав технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

* процессор с тактовой частотой, 1 ГГц, не менее;
* оперативную память объемом, 512 Мб, не менее;
* жесткий диск со свободным местом 500 Мб, не менее;
* монитор, с разрешением экрана 1024\*768, не менее;
* компьютерная мышь;
* клавиатура;
* принтер;
* CD – привод.

Минимальный состав программных средств

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7 и выше.

Требования к персоналу (пользователю)

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 2 штатных единиц – пользователь программы, оператор.

Пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы семейства Windows.

**Выполнение программы**

Для установки программы необходимо открыть файл setup.exe от имени администратора с CD–диска (Приложение 4).

При установке выбрать установку для всех пользователей.

После установки ярлыки приложений отобразиться на рабочем столе.

Для запуска клиентской программы необходимо открыть ярлык либо файл StealthForce.Client.exe в папке установки, после пользователю отобразиться главное окно программы (Рисунок 3.6).

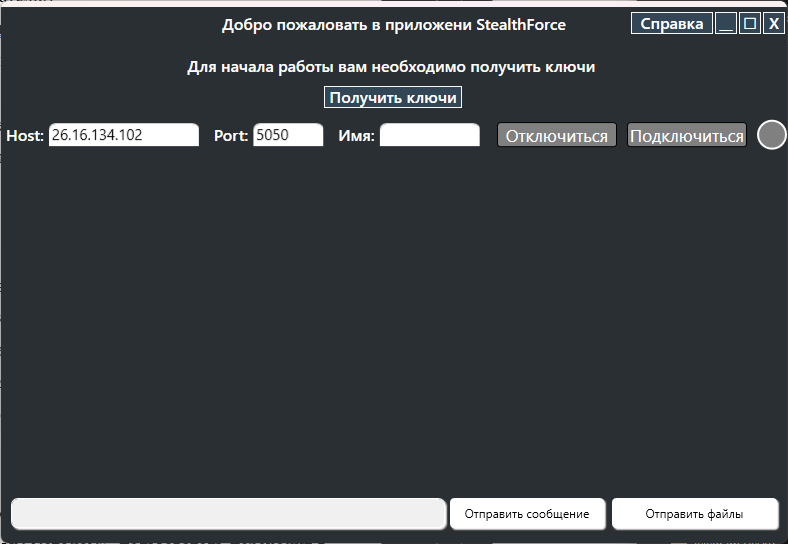


Рисунок 3.6 – Главное окно

Для того что бы начать использовать приложение вам необходимо, получить ключи подключения. Для этого, необходимо нажать на кнопку «Получить ключи». На окне приложения будут доступны 2 публичных ключа позволяющие пользователю подключиться к серверу. (Рисунок 3.7).

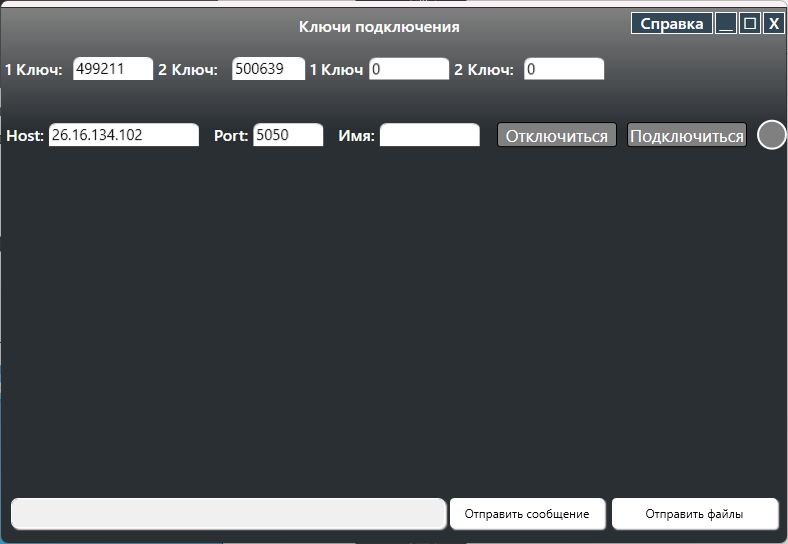


Рисунок 3.7 – Ключи подключения

После того как мы получили ключи, пользователю необходимо обменяться своими ключами с вторым пользователем. Полученные ключи требуется ввести в поля «1 Ключ», «2 Ключ» для того, чтобы создать подключение на сервере и создать защищённую передачу информации. (Рисунок 3.8).

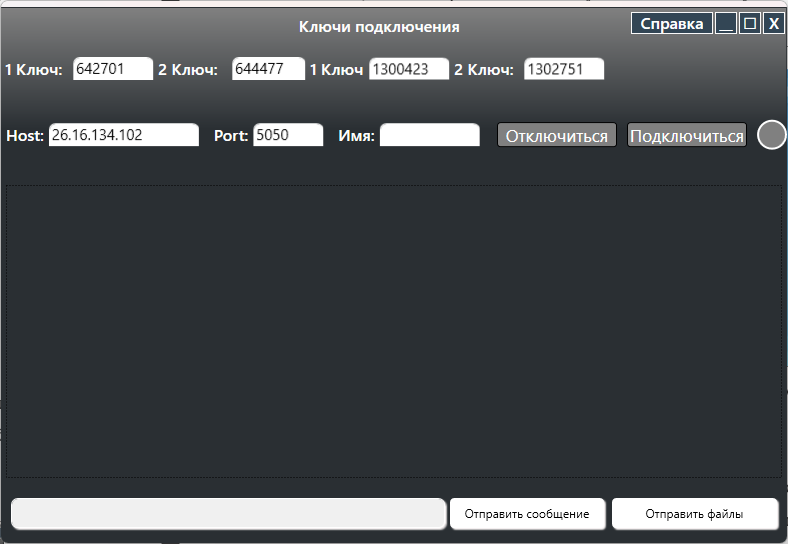


Рисунок 3.8 – Ввод ключей

Следующим шагом будет ввод имени пользователя, и нажатие на кнопку «Подключиться» для начала работы с приложением. Если подключение будет выполнено, сигнал справа от кнопки будет зелёным что будет сигнализировать об успешном подключении. (Рисунок 3.9).

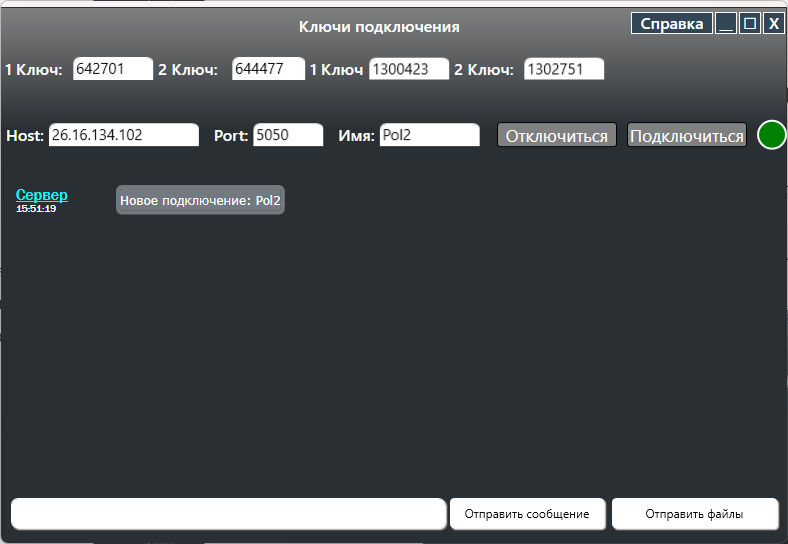


Рисунок 3.9 – Ввод имени и подключение к серверу

Приложение готово для использования. Теперь после подключения к серверу вам доступна отправка сообщений и файлов. Для того что бы отправить сообщение пользователю необходимо ввести его в поле для сообщений, и нажать на кнопку «Отправить сообщение» (Рисунок 3.10).

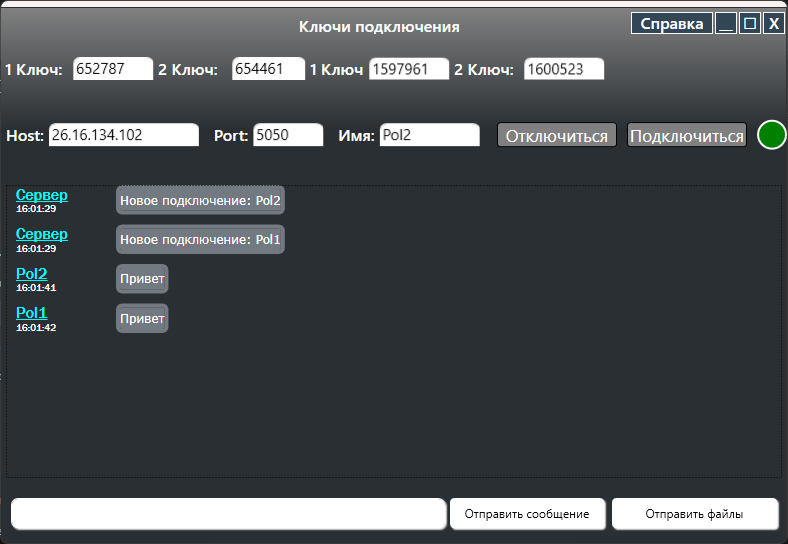


Рисунок 3.10 – Отправка сообщения

Для того что бы отправить файл другому пользователю вам необходимо нажать на кнопку «Оправить файл» и выбрать необходимый документ. Файл будет отправлен второму клиенту и сохранён на рабочий стол в созданную приложением папку «Файлы». (Рисунок 3.11).

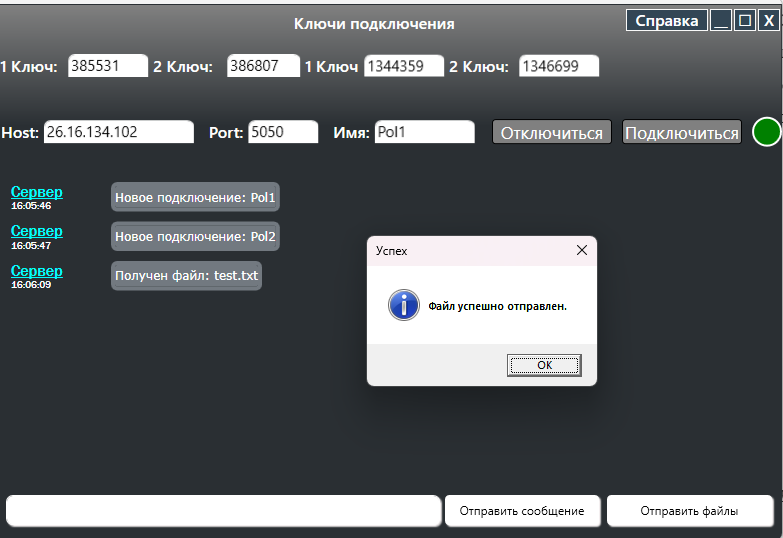


Рисунок 3.11 – Отправка файла

Для того что бы завершить работу с приложением, необходимо нажать на кнопку «Отключиться». (Рисунок 3.12)

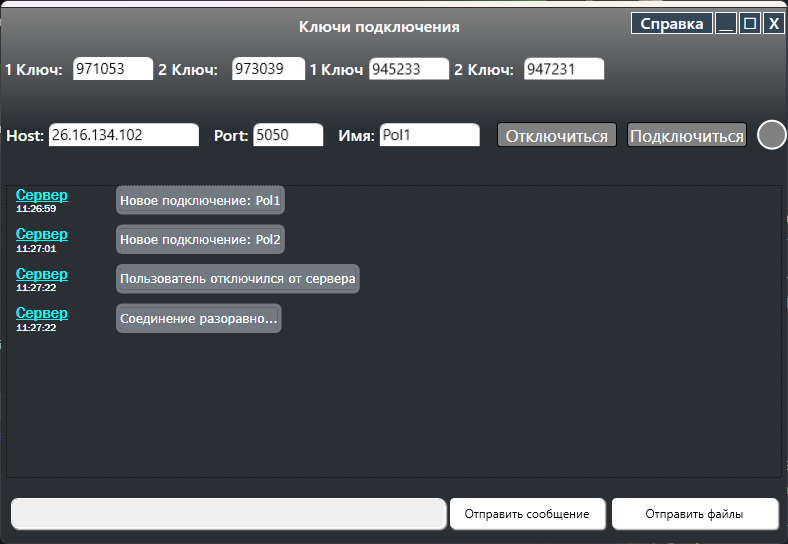


Рисунок 3.12 – Отключение от сервера

Оператору доступны функции подключения, отправки файлов и сообщений, генерации сообщений.

У Оператора есть возможность изменять имя пользователя, IP-адрес, порт, размер окна.

* 1. Руководство оператора для серверного приложения

Функциональным назначением программы является шифрование и защита передаваемой информации.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* Передавать информацию в зашифрованном виде.
* Выводить информацию новых подключений.
* Обработка сообщений клиентов и их передача
* Обработка сообщений клиентов и их передача.

Условия выполнения программы

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средства в части условий их эксплуатации.

Минимальный состав технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий в себя:

* процессор с тактовой частотой, 1 ГГц, не менее;
* оперативную память объемом, 512 Мб, не менее;
* жесткий диск со свободным местом 500 Мб, не менее;
* монитор, с разрешением экрана 1024\*768, не менее;
* компьютерная мышь;
* клавиатура;
* принтер;
* CD – привод.

Минимальный состав программных средств

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7 и выше.

Требования к персоналу (пользователю).

Минимальное количество персонала, требуемого для работы программы, должно составлять не менее 1 штатной единицы – пользователь программы, оператор.

Пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы семейства Windows.

**Выполнение программы**

Для установки программы необходимо открыть файл setup.exe от имени администратора с CD–диска (Приложение 3).

При установке выбрать установку для всех пользователей.

После установки ярлыки приложений отобразиться на рабочем столе.

Для запуска клиентской программы необходимо открыть ярлык либо файл StealthForce.Server.exe в папке установки, после пользователю отобразиться главное окно программы (Рисунок 3.13).

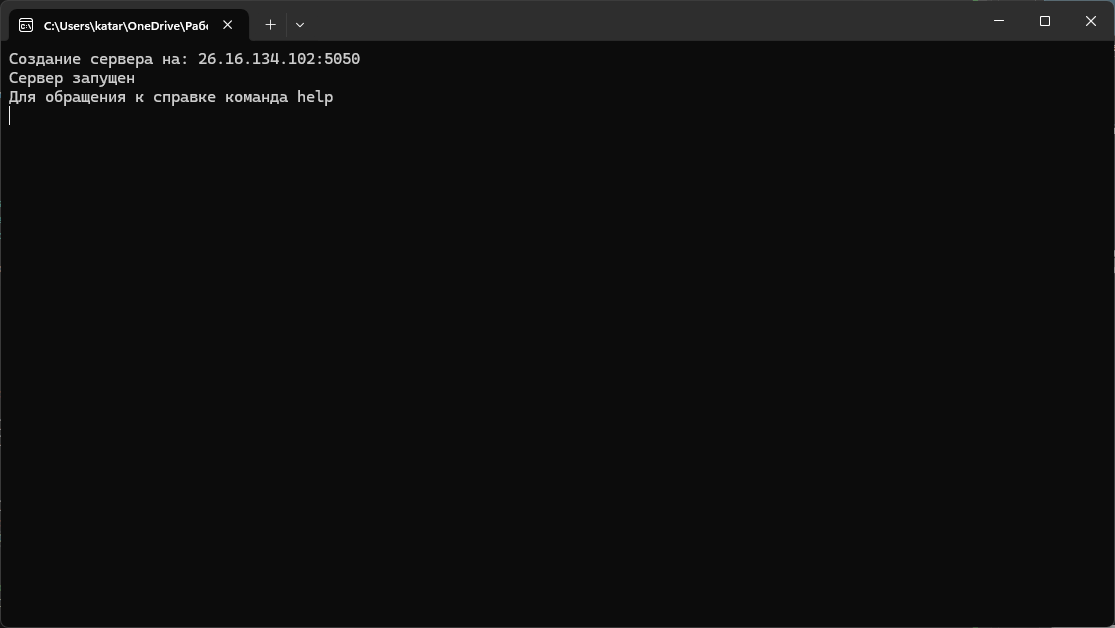


Рисунок 3.13 – Серверное окно

При новом подключении в серверном окне появиться сообщение о том, что выполнено новое подключение, оператору приложения доступен IP-адрес нового пользователя. (Рисунок 3.14).

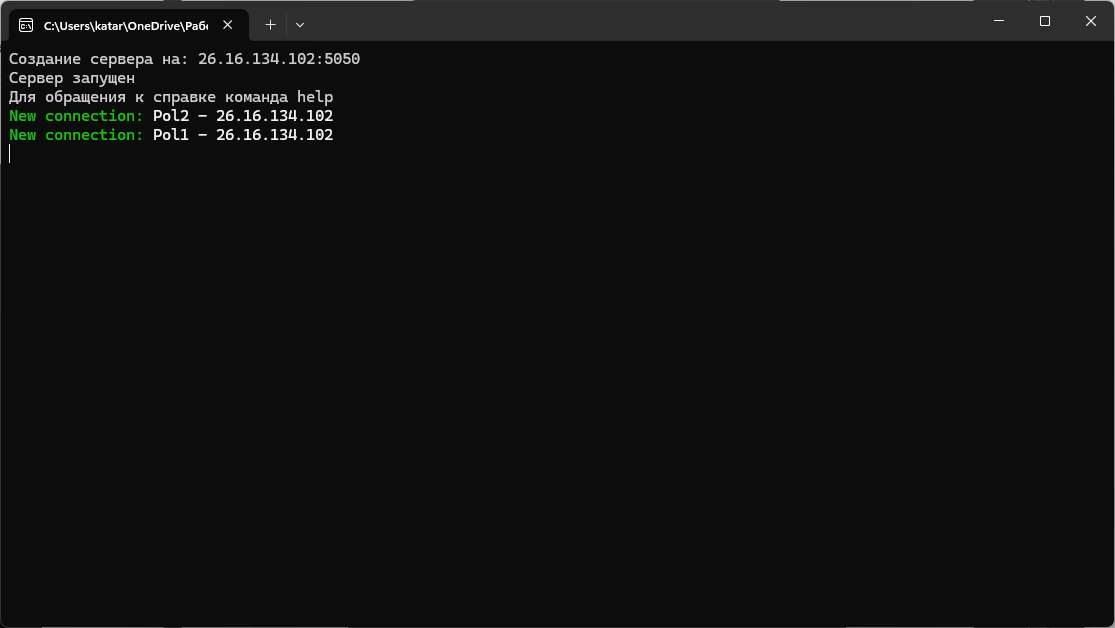


Рисунок 3.14 – Новое подключение

При отправке сообщения оператору будут доступны, имя пользователя, а также зашифрованное сообщение. (Рисунок 3.15).

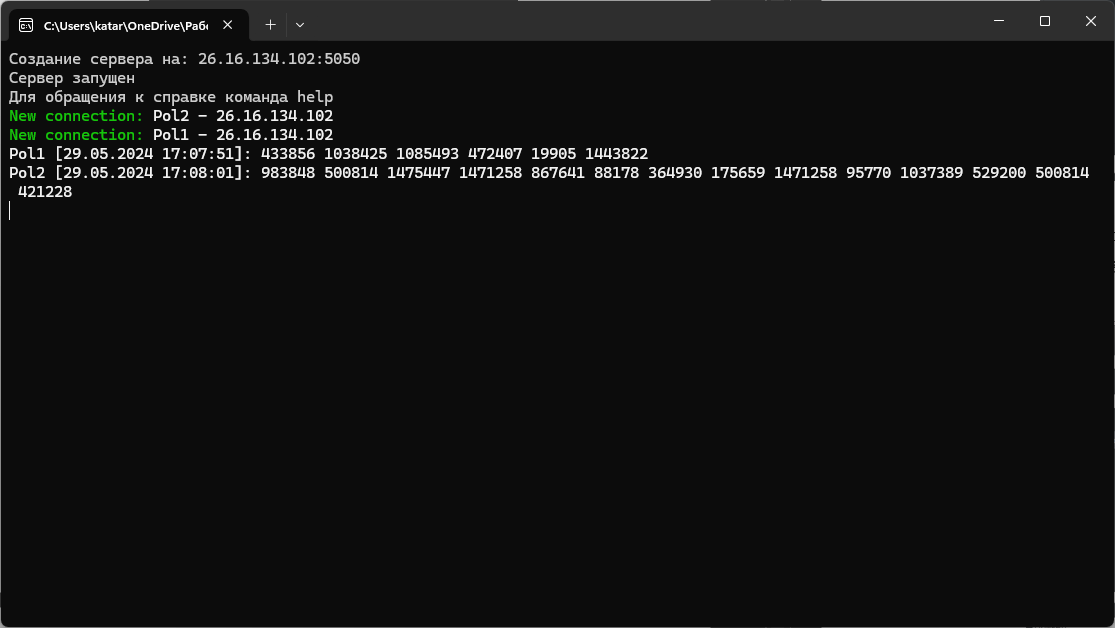


Рисунок 3.15 – Передача сообщений

При отправке файла оператору также будут доступны имя пользователя, а также имя файла. (Рисунок 3.16).

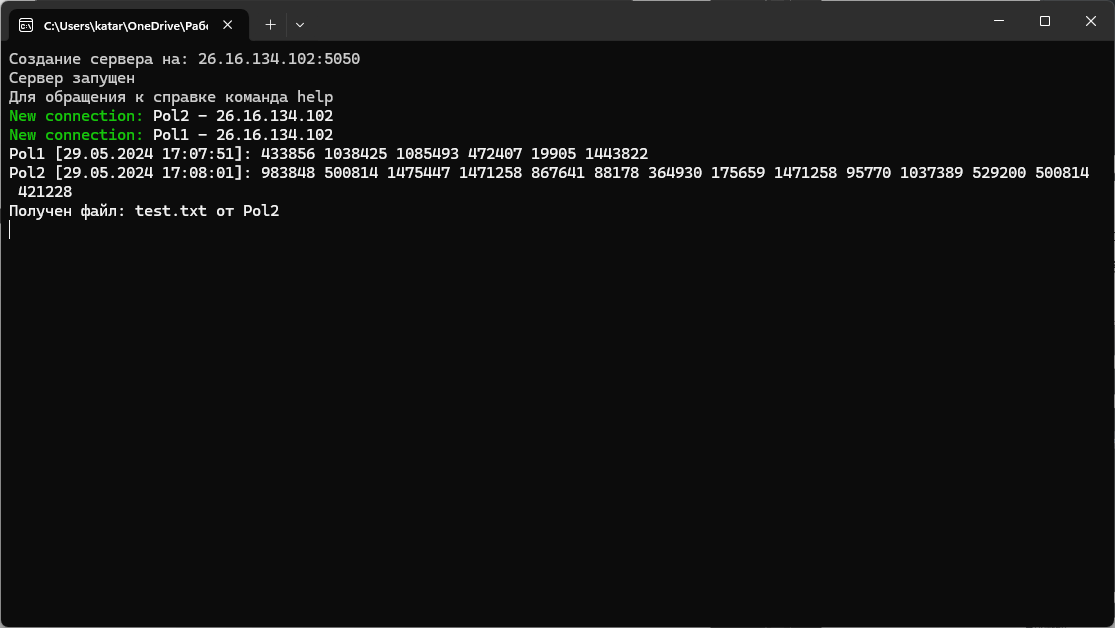


Рисунок 3.16 – Передача файла

При отключении пользователя так же выводиться сообщение об его отключении. (Рисунок 3.17).

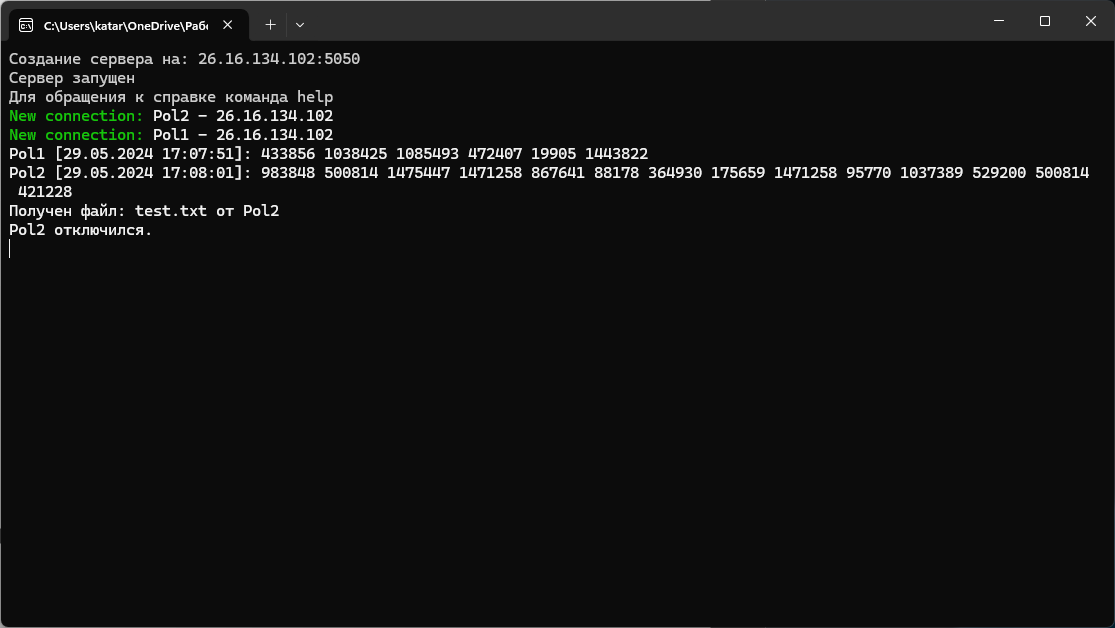


Рисунок 3.17 – Отключение пользователя

Для того что бы обратиться к справке, необходимо ввести команду help. (Рисунок 3.18).

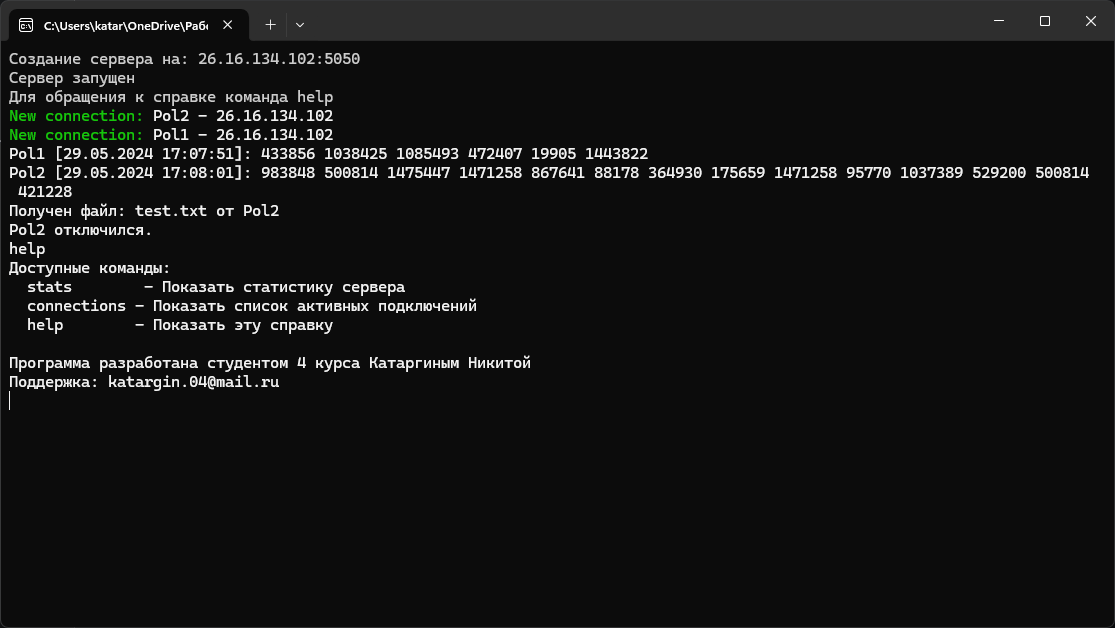


Рисунок 3.18 – Справочная система

Для того что бы обратиться к статистике сервера, необходимо ввести команду stats. (Рисунок 3.19).

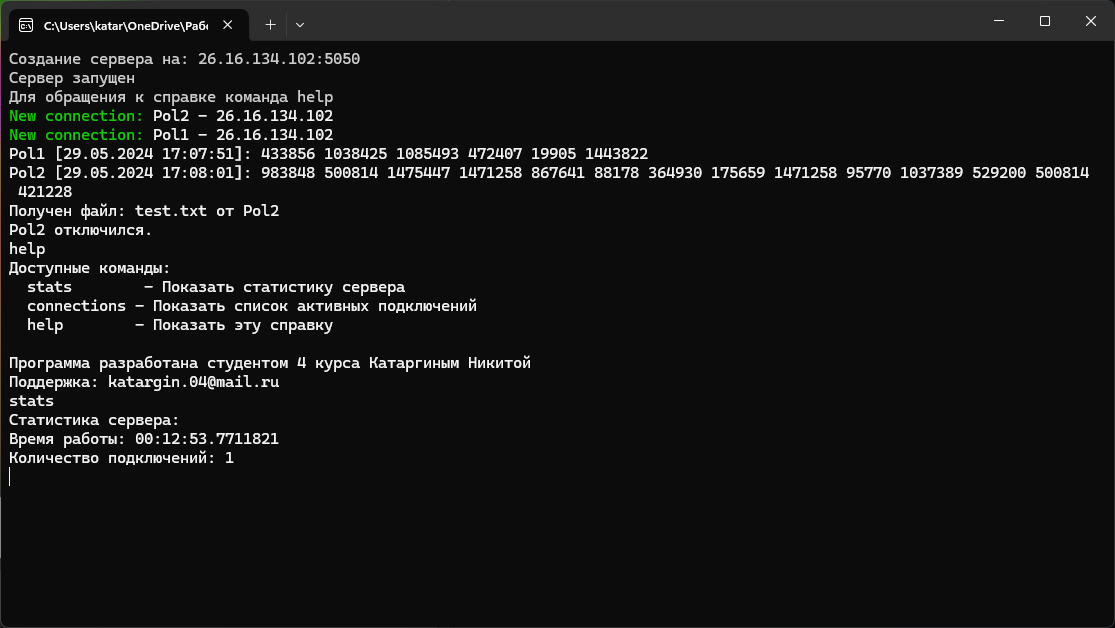


Рисунок 3.19 – Статистика сервера

Для того что бы посмотреть список активных подключений сервера, необходимо ввести команду connections. (Рисунок 3.20).

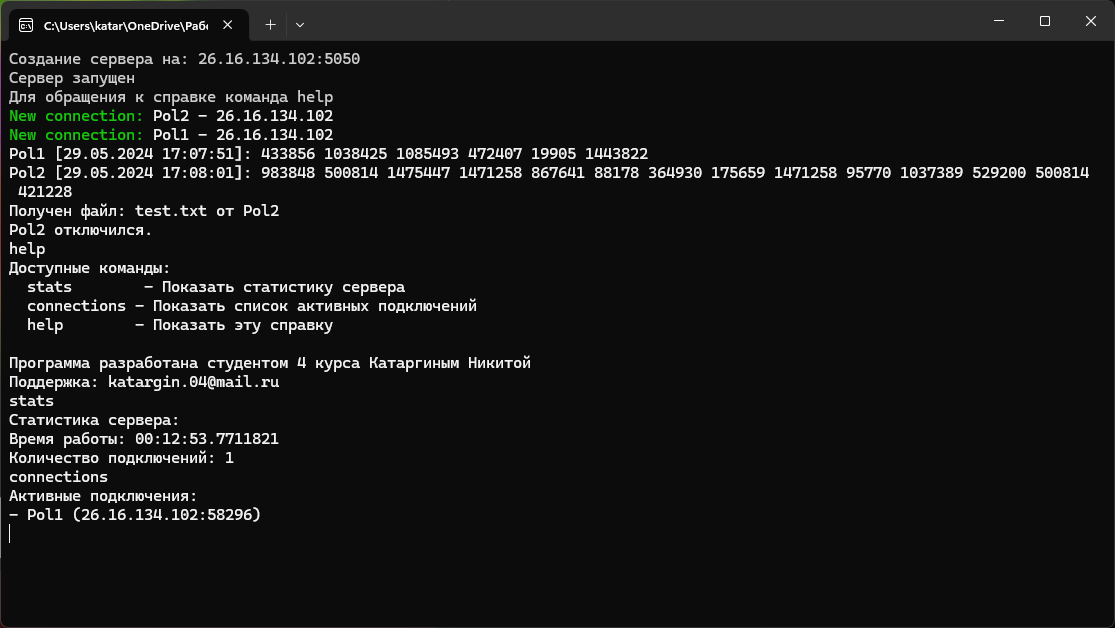


Рисунок 3.20 – Список активных подключений

Для того что бы запустить сервер на определённом порту и адресе для этого необходимо перейти в репозиторий приложения и в поле, где находиться путь написать команду «cmd» после чего в командной строке прописать EncryptedChat.Server.exe (ваш IP-адрес и порт например - 192.168.11.1 5050) (Рисунок 3.21).

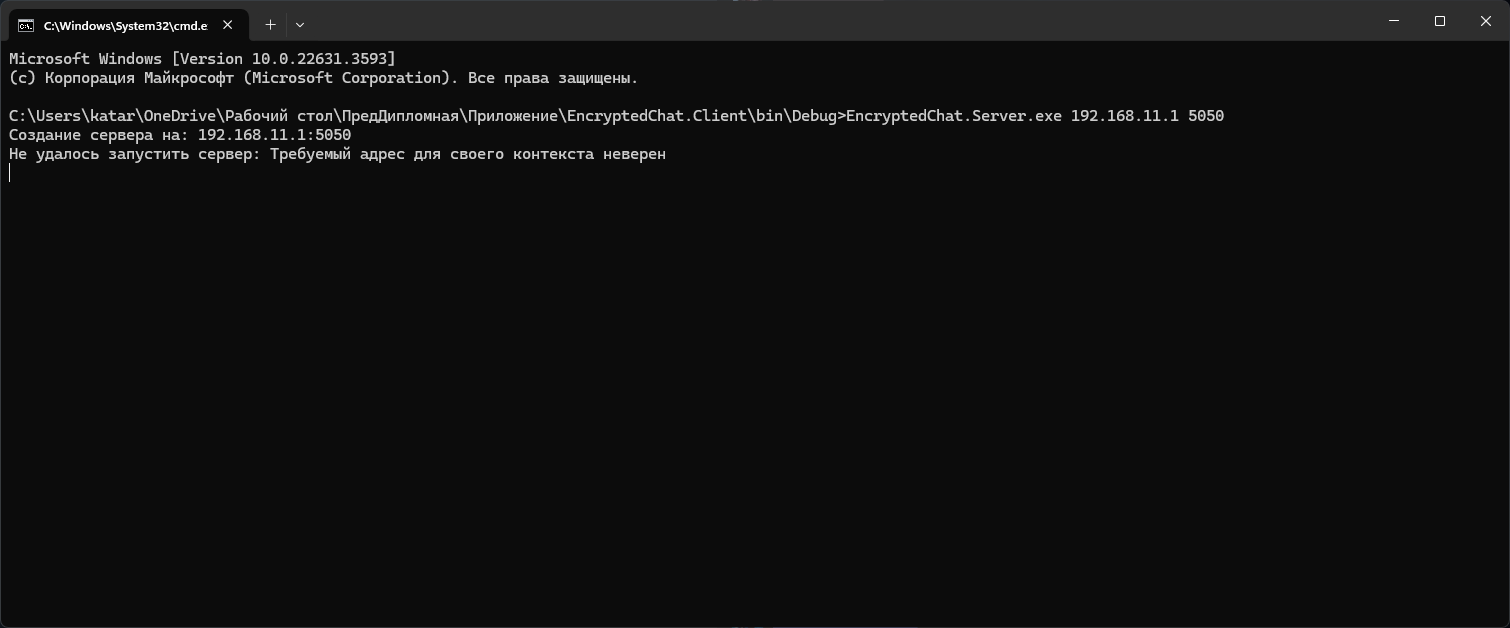


Рисунок 3.21 – Создание сервера на собственном IP-адресе

Оператору доступны функции статистики сервера, просмотра зашифрованных сообщений и файлов, просмотра активных подключений, вызова справочной системы.

ВЫВОД ПО ГЛАВЕ 3

В результате написания третьей главы было проведено тестирование. Была описана экспериментально-прикладная часть программной документации. В ходе данной главы была осуществлена опытная эксплуатация.

Проведя тестирование и опытную эксплуатации. Были выявлены ошибки при работе программы, которые были решены при отладке. Были проверены все функциональные возможности программы при работе с реальными данными. Также было выявлено, что запас функциональных возможностей предусматривает долгосрочное использование программы.

На основе созданной программы было написано руководство оператора, которое достаточно подробно описывает алгоритм работы с программой.

ЗАКЛЮЧЕНИЕ

Основной задачей дипломного проекта было создание программного обеспечения для защиты файлов, и документов, передаваемых по каналам связи и распространяемых в сети Интернет, путем разработки системы с использованием криптографических методов.

В ходе написания дипломного проекта был создан программный продукт и техническая документация к нему, описаны алгоритмы и архитектура работы программы.

В ходе работы над аналитической частью были описаны наиболее часто используемые понятия. В техническом задании описаны функциональные значения, технический состав ЭВМ.

В процессе работы над конструкторской частью была разработана архитектура программы, описан алгоритм работы программы.

Было проведено тестирование программного продукта на наличие ошибок. Была проведена опытная эксплуатация, в процессе которой куратор ознакомился с программным продуктом. Во время опытной эксплуатации было разработано руководство оператора, упрощающее ознакомление пользователя с программой. Программный продукт «StealthForce» получил хороший отзыв от куратора.

Разработанный программный продукт имеет визуально приятный интерфейс, в освоении и использовании он достаточно прост.

За время написания дипломного проекта была изучена предметная область, которая помогла правильно создать свой собственный алгоритм шифрования и разработать сервер для обмена информацией.

Программный продукт создавался при помощи Microsoft Visual Studio 2022.

При проверке работоспособности программного обеспечения было выявлено, что программа работает верно с различными наборами входных данных.

В результате написания дипломного проекта было разработано программное обеспечение для защиты информации с использованием криптографических методов.

Программа имеет следующие перспективы улучшения:

* Изменение интерфейса;
* Добавление групповых чатов;
* Добавление уведомлений;
* Создание Web версии.

Все задачи были реализованы, каждая из них выполнялась вовремя и в соответствии с заданием на дипломную работу.

Цели дипломного проекта были успешно достигнуты. Программное обеспечение получило хороший отзыв. Требования заказчика были выполнены в полном объеме.

Дополнительно, в процессе разработки были учтены современные требования к безопасности и удобству использования программного обеспечения. Программа была адаптирована для работы в различных сетевых условиях, обеспечивая стабильное соединение и защищенную передачу данных. Внедрение асимметричного шифрования с использованием алгоритма RSA позволило достичь высокого уровня защиты информации.

Благодаря гибкой архитектуре, программа легко поддается модификациям и улучшениям, что открывает возможности для дальнейшего развития и расширения функционала. В перспективе возможно интеграция с другими системами безопасности и разработка мобильных приложений для платформ Android, что сделает использование программного обеспечения еще более удобным и доступным.

В ходе работы над проектом были приобретены ценные навыки в области программирования, криптографии и разработки защищенных систем передачи данных. Полученные знания и опыт будут полезны в дальнейшей профессиональной деятельности.

Программный продукт «StealthForce» демонстрирует высокие стандарты качества и надежности, подтверждая правильность выбранного подхода к разработке и реализации поставленных задач.

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 19.101-77. Единая система программной документации. Виды  
   программ и программных документов, введ. 01.01.1978 – г. Москва: Изд-во  
   стандартов, 1980. – 4 с.
2. ГОСТ 19.103-77. Единая система программной документации.  
   Обозначение программ и программных документов, введ. 01.07.1978 – г.  
   Москва: Изд-во стандартов, 1978 – 2 с.
3. ГОСТ 19.104-78. Единая система программной документации.  
   Основные надписи, введ. 01.01.1978 – г. Москва: Изд-во стандартов, 1980 – 6  
   с.
4. ГОСТ 19.106-78. Единая система программной документации.  
   Требования к программным документам, выполненным печатным способом,  
   введ. 01.01.1980 – г. Москва: Изд-во стандартов, 1989 – 6 с.
5. ГОСТ 19.201-78. Единая система программной документации.  
   Техническое задание. Требования к содержанию и оформлению, введ.  
   01.01.1980 – г. Москва: Изд-во стандартов, 1988 – 3 с.
6. ГОСТ 19.401-78. Единая система программной документации.  
   Текст программы. Требования к содержанию и оформлению, введ. 01.01.1980,  
   г. Москва: Изд-во стандартов, 1988. - 2 с.
7. ГОСТ 19.505-79. Единая система программной документации.  
   Руководство оператора. Требования к содержанию и оформлению, введ.  
   01.01.1980 – г. Москва: Изд-во стандартов, 1987 – 6 с.
8. Бернет С., Пэйн С. Криптография, официальное руководство по RSA Security. – Москва.: БИНОМ, 2002. – 384 с.
9. Виноградов И.М. Основы теории чисел: учебное пособие [Текст] – М.:Лань, 2009. – 176 с.
10. Коутинохо С. Введение в теорию чисел. Алгоритм RSA – Москва: Постмаркет, 2001. – 328 с.
11. Кудряшов Б., Д. Основы теории кодирования. – Санкт-Петербург: БХВ-Петербург, 2016. – 400 с.
12. Алгоритм RSA. [Электронный ресурс] URL: https://studfile.net/preview/2983620/ (дата обращения: 01.03.2024).
13. Асимметричная криптосистема RSA. [Электронный ресурс] URL: https://spravochnick.ru/informatika/asimmetrichnaya\_kriptosistema\_rsa/ (дата обращения: 02.03.2024).
14. Использование сокетов для отправки и получения данных через TCP-.NET. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/fundamentals/networking/sockets/socket-services (дата обращения: 05.03.2024).
15. Обмен информацией между компьютерами. Введение в интернет. [Электронный ресурс] URL: https://ru.hexlet.io/courses/internet-fundamentals/lessons/tcp-ip/theory\_unit (дата обращения: 26.02.2024)
16. Отправка и получение данных в TCP. Двунаправленная связь. [Электронный ресурс] URL: https://metanit.com/sharp/net/4.6.php (дата обращения: 26.02.2024).
17. Сетевое программирование в С#. TCP-сервер на сокетах [Электронный ресурс] URL: https://metanit.com/sharp/net/3.2.php (дата обращения: 26.02.2024).
18. RSA Algorithm in Cryptography - GeeksforGeeks [Электронный ресурс] URL: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/ (дата обращения: 02.02.2024).

ПРИЛОЖЕНИЕ

**Приложение 1**

Код портативного приложения

**Файл DateTimeConverter.cs**

using System;

using System.Globalization;

using System.Windows.Data;

namespace EncryptedChat.Client.Converters

{

public class DateTimeConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is null)

return default;

if (!(value is DateTime))

throw new ArgumentException(nameof(value));

var date = (DateTime)value;

return date.ToString("HH:mm:ss");

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Файл EmptyTextConverter.cs**

using System;

using System.Globalization;

using System.Windows.Data;

namespace EncryptedChat.Client.Converters

{

public class EmptyTextConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is string str)

{

return !string.IsNullOrEmpty(str);

}

return false;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

**Файл ButtonsResources.xaml**

<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="CommonResources.xaml" />

</ResourceDictionary.MergedDictionaries>

<Style x:Key="CommandButton" TargetType="Button">

<Setter Property="Foreground" Value="White" />

<Setter Property="FontSize" Value="17" />

<Setter Property="Margin" Value="0,0,10,0" />

<Setter Property="MinWidth" Value="120" />

<Setter Property="MaxHeight" Value="25" />

<Setter Property="Background" Value="Gray" />

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border

Background="{TemplateBinding Background}"

BorderBrush="Black"

BorderThickness="1"

CornerRadius="3">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" />

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="#FF5BE6D3" />

</Trigger>

<Trigger Property="IsEnabled" Value="False">

<Setter Property="Background" Value="Gray" />

</Trigger>

</Style.Triggers>

</Style>

<Style x:Key="SendButton" TargetType="Button">

<Setter Property="MinWidth" Value="70" />

<Setter Property="Margin" Value="5" />

<Setter Property="BorderThickness" Value="0" />

<Setter Property="MinHeight" Value="30" />

<Setter Property="Height" Value="30" />

<Setter Property="MaxHeight" Value="30" />

<Setter Property="Background" Value="White" />

<Setter Property="Effect" Value="{StaticResource DefaultShadow}" />

<Setter Property="Content" Value="Отправить"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border Background="{TemplateBinding Background}" CornerRadius="4">

<ContentPresenter

Margin="3"

HorizontalAlignment="Center"

VerticalAlignment="Center" />

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="Gray" />

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="Button">

<Setter Property="Width" Value="22" />

<Setter Property="Height" Value="22" />

<Setter Property="BorderThickness" Value="1" />

<Setter Property="BorderBrush" Value="#FFF9F9F9" />

<Setter Property="Margin" Value="0,5,2,0" />

<Setter Property="Background" Value="#FF334655" />

<Setter Property="Foreground" Value="White" />

<Setter Property="Padding" Value="0,-2,0,0" />

<Setter Property="FontWeight" Value="Medium" />

<Setter Property="FontSize" Value="16" />

</Style>

</ResourceDictionary>

**Файл CommonResources.xaml**

<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<DropShadowEffect

x:Key="DefaultShadow"

BlurRadius="2"

ShadowDepth="2.5"

Color="#FFFFFDFD" />

</ResourceDictionary>

**Файл TextResources.xaml**

<ResourceDictionary

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:converters="clr-namespace:EncryptedChat.Client.Converters">

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="CommonResources.xaml" />

</ResourceDictionary.MergedDictionaries>

<converters:EmptyTextConverter x:Key="EmptyTextConverter" />

<converters:DateTimeConverter x:Key="DateTimeConverter" />

<Style TargetType="Label">

<Setter Property="FontSize" Value="16" />

<Setter Property="FontWeight" Value="DemiBold" />

<Setter Property="Foreground" Value="White" />

<Setter Property="VerticalAlignment" Value="Center" />

</Style>

<Style TargetType="TextBox">

<Setter Property="FontSize" Value="15" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="HorizontalAlignment" Value="Stretch" />

</Style>

<Style x:Key="AuthorStyle" TargetType="TextBlock">

<Setter Property="Foreground" Value="Aqua" />

<Setter Property="HorizontalAlignment" Value="Left" />

<Setter Property="VerticalAlignment" Value="Top" />

<Setter Property="FontSize" Value="16" />

<Setter Property="TextDecorations" Value="Underline" />

<Setter Property="FontFamily" Value="Franklin Gothic Medium" />

</Style>

<Style

x:Key="SendTimeStyle"

BasedOn="{StaticResource AuthorStyle}"

TargetType="TextBlock">

<Setter Property="FontSize" Value="10" />

<Setter Property="Foreground" Value="White" />

<Setter Property="TextDecorations" Value="{x:Null}" />

</Style>

<Style x:Key="MessageStyle" TargetType="TextBlock">

<Setter Property="MaxWidth" Value="500" />

<Setter Property="TextWrapping" Value="Wrap" />

<Setter Property="FontFamily" Value="Verdana" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="Foreground" Value="White" />

</Style>

<DataTemplate x:Key="TextBlockDataTemplate">

<Border

BorderBrush="#FFD8D4D4"

BorderThickness="2"

CornerRadius="5">

<TextBox

Padding="5,0,0,0"

VerticalAlignment="Center"

Background="#FF727272"

BorderThickness="0"

FontSize="16"

Foreground="White"

Text="{Binding Path=Content}" />

</Border>

</DataTemplate>

<ControlTemplate x:Key="TextBoxBaseControlTemplate" TargetType="{x:Type TextBoxBase}">

<Border

x:Name="Bd"

Height="30"

Background="{TemplateBinding Background}"

BorderBrush="White"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="6"

Effect="{StaticResource DefaultShadow}">

<ScrollViewer x:Name="PART\_ContentHost" />

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsEnabled" Value="False">

<Setter TargetName="Bd" Property="Background" Value="{DynamicResource {x:Static SystemColors.ControlBrushKey}}" />

<Setter Property="Foreground" Value="{DynamicResource {x:Static SystemColors.GrayTextBrushKey}}" />

</Trigger>

<Trigger Property="Width" Value="Auto">

<Setter Property="MinWidth" Value="100" />

</Trigger>

<Trigger Property="Height" Value="Auto">

<Setter Property="MinHeight" Value="20" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

<Style x:Key="InputMessageTextStyle" TargetType="TextBlock">

<Setter Property="MaxWidth" Value="500" />

<Setter Property="TextWrapping" Value="Wrap" />

<Setter Property="FontFamily" Value="Verdana" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="Foreground" Value="White" />

</Style>

</ResourceDictionary>

**Файл HelpWindow.xaml**

<Window x:Class="EncryptedChat.Client.View.HelpWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:EncryptedChat.Client.View"

mc:Ignorable="d"

Title="HelpWindow" Height="450" Width="717" WindowStyle="None" ResizeMode="NoResize">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/Resources/TextResources.xaml" />

<ResourceDictionary Source="/Resources/ButtonsResources.xaml" />

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<Window.Background >

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">

<GradientStop Offset="1" Color="#FF2A2F33" />

<GradientStop Color="Gray" />

</LinearGradientBrush>

</Window.Background>

<Grid MouseLeftButtonDown="Grid\_MouseLeftButtonDown" MouseMove="Grid\_MouseMove">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="115\*"/>

<ColumnDefinition Width="458\*"/>

<ColumnDefinition Width="115\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="30\*"/>

<RowDefinition Height="319\*"/>

<RowDefinition Height="101\*"/>

</Grid.RowDefinitions>

<Button Click="MinimizeWindow" Content="\_\_\_" Margin="60,4,38,4" Grid.Column="2" />

<Button Click="CloseWindow" Content="X" Margin="88,4,10,4" Grid.Column="2" />

<Button Style="{StaticResource SendButton}" x:Name="btnPrevious" Content="Предыдущее" HorizontalAlignment="Center" Margin="0,8,0,0" VerticalAlignment="Top" Click="btnPrevious\_Click" Height="30" Width="86" Grid.Row="2"/>

<Button Style="{StaticResource SendButton}" x:Name="btnNext" Content="Следующее" HorizontalAlignment="Left" Margin="6,8,0,0" VerticalAlignment="Top" Click="btnNext\_Click" Height="30" Width="87" Grid.Row="2" Grid.Column="2"/>

<TextBlock x:Name="tbInstruction" HorizontalAlignment="Center" Margin="0,310,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Height="74" Width="390" Background="Gray" Foreground="White" Grid.Row="1" Grid.RowSpan="2" Grid.Column="1"/>

<Image x:Name="imgHelp" Margin="57,10,48,29" RenderTransformOrigin="0.5,0.5" Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="3"/>

<Label Content="Справка" HorizontalAlignment="Left" Margin="0,1,0,0" VerticalAlignment="Top" Grid.RowSpan="2" Height="31" Width="73"/>

<Label Height="30" Margin="0,0,64,0" Grid.ColumnSpan="3"/>

</Grid>

</Window>

**Файл HelpWindow.cs**

using System;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media.Imaging;

namespace EncryptedChat.Client.View

{

public partial class HelpWindow : Window

{

private string[] imagePaths = { "/Resources/FAQ1.png", "/Resources/FAQ2.png", "/Resources/FAQ3.png", "/Resources/FAQ4.png", "/Resources/FAQ5.png", "/Resources/FAQ6.png", "/Resources/FAQ7.png" };

private string[] instructions = { "Для начала работы вам необходимо получить ключи. Для этого нажмите на кнопку получить ключи.", "Получив ключи вам необходимо обменяться с 2 клиентом 1 и 2 ключом. После чего ввести их в необходимые поля", "Следующим этапом подготовки будет ввод имени пользователя. После чего нажав на кнопку 'Подключиться' приложение готово к использованию.", "Для того что бы отправить сообщение, вам необходимо ввести текст сообщения, в необходимое поле, и нажать на кнопку 'Отправить сообщение'.", "Для того что бы отправть файл вам необходимо нажть на кнопку 'Отправть файл', и выбрать файл после чего файл будет отправлен", "Для того что бы закончить работу вам необходимо нажать на кнопку 'Отключиться'", "Поддержка - +7 (953) 685 25-71 Почта - katargin.04@mail.ru Программу разработал студент 4 курса Катаргин Никита"};

private int currentIndex = 0;

public HelpWindow()

{

InitializeComponent();

ShowImageAndInstruction();

}

private void Grid\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.Left)

DragMove();

}

private void Grid\_MouseMove(object sender, MouseEventArgs e)

{

if (e.LeftButton == MouseButtonState.Pressed)

DragMove();

}

private void ShowImageAndInstruction()

{

imgHelp.Source = new BitmapImage(new Uri(imagePaths[currentIndex], UriKind.Relative));

tbInstruction.Text = instructions[currentIndex];

}

private void btnPrevious\_Click(object sender, RoutedEventArgs e)

{

if (currentIndex > 0)

{

currentIndex--;

ShowImageAndInstruction();

}

}

private void btnNext\_Click(object sender, RoutedEventArgs e)

{

if (currentIndex < imagePaths.Length - 1)

{

currentIndex++;

ShowImageAndInstruction();

}

}

private void MinimizeWindow(object sender, RoutedEventArgs e)

{

SystemCommands.MinimizeWindow(this);

}

private void CloseWindow(object sender, RoutedEventArgs e)

{

SystemCommands.CloseWindow(this);

}

}

}

**Файл MainWindow.xaml**

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:av="http://schemas.microsoft.com/expression/blend/2008" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="av"

x:Class="EncryptedChat.Client.View.MainWindow"

Title="StealthForce"

Width="800"

Height="550"

MinWidth="800"

MinHeight="200"

Background="#FF2A2F33"

WindowStartupLocation="CenterScreen"

WindowStyle="None" Icon="/View/icon.ico">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/Resources/TextResources.xaml" />

<ResourceDictionary Source="/Resources/ButtonsResources.xaml" />

</ResourceDictionary.MergedDictionaries>

<Storyboard x:Key="Storyboard1">

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="dockPanel" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<EasingDoubleKeyFrame KeyTime="00:00:00" Value="0.813"/>

<EasingDoubleKeyFrame KeyTime="00:00:00.7000000" Value="125.924"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="dockPanel" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<EasingDoubleKeyFrame KeyTime="00:00:00" Value="-0.813"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="label" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<EasingDoubleKeyFrame KeyTime="00:00:00" Value="0.812"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="label" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<EasingDoubleKeyFrame KeyTime="00:00:00" Value="33.309"/>

<EasingDoubleKeyFrame KeyTime="00:00:00.7000000" Value="158.42"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="getKeysButton" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<EasingDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

<EasingDoubleKeyFrame KeyTime="00:00:01" Value="251.846"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="welcomeLabel" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<EasingDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

<EasingDoubleKeyFrame KeyTime="00:00:01" Value="-92.614"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="getKeysLabel" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<EasingDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

<EasingDoubleKeyFrame KeyTime="00:00:01" Value="-92.614"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="welcomeLabel" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<EasingDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

<EasingDoubleKeyFrame KeyTime="00:00:01" Value="-3.25"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetName="getKeysLabel" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<EasingDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

<EasingDoubleKeyFrame KeyTime="00:00:01" Value="-3.25"/>

</DoubleAnimationUsingKeyFrames>

</Storyboard>

</ResourceDictionary>

</Window.Resources>

<Grid MouseDown="DockPanel\_MouseDown">

<Grid.RowDefinitions>

<RowDefinition Height="33" />

<RowDefinition Height="Auto"/>

<RowDefinition Height="auto" MinHeight="66.72" />

<RowDefinition />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>

<DockPanel x:Name="dockPanel" Grid.Row="0" Margin="0,-125,0,37" RenderTransformOrigin="0.5,0.5">

<DockPanel.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</DockPanel.RenderTransform>

<DockPanel.Background>

<LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1">

<GradientStop Offset="1" Color="#FF2A2F33" />

<GradientStop Color="Gray" />

</LinearGradientBrush>

</DockPanel.Background>

<StackPanel Orientation="Horizontal">

<Label Content="1 Ключ:" Width="73" />

<TextBox

Width="80"

Height="23"

IsReadOnly="True"

Template="{StaticResource TextBoxBaseControlTemplate}"

Text="{Binding LocalE, Mode=OneWay}" />

<Label Content="2 Ключ:" Width="79" />

<TextBox

Width="73"

Height="23"

IsReadOnly="True"

Template="{StaticResource TextBoxBaseControlTemplate}"

Text="{Binding LocalN, Mode=OneWay}" />

<Label Content="1 Ключ" Width="64" />

<TextBox

Width="80"

Height="22"

IsReadOnly="{Binding CanDisconnect}"

Template="{StaticResource TextBoxBaseControlTemplate}"

Text="{Binding RemoteE, UpdateSourceTrigger=PropertyChanged}" />

<Label Content="2 Ключ:" Width="75" />

<TextBox

Width="80"

Height="22"

IsReadOnly="{Binding CanDisconnect}"

Template="{StaticResource TextBoxBaseControlTemplate}"

Text="{Binding RemoteN, UpdateSourceTrigger=PropertyChanged}" />

</StackPanel>

</DockPanel>

<WrapPanel

HorizontalAlignment="Right"

VerticalAlignment="Top"

Orientation="Horizontal">

<Button Content="Справка" HorizontalAlignment="Left" VerticalAlignment="Bottom" Click="HelpButton\_Click" Width="82" Height="22"/>

<Button Click="MinimizeWindow" Content="\_\_\_" />

<Button Click="MaximizeWindow" Content="⬜" />

<Button Click="CloseWindow" Content="X" />

</WrapPanel>

<DockPanel Grid.Row="2" Margin="0,10,0,24">

<StackPanel Grid.Row="0" Orientation="Horizontal">

<Label Content="Host:" />

<TextBox

Height="23"

MinWidth="150"

Template="{StaticResource TextBoxBaseControlTemplate}"

Text="{Binding Host}" />

</StackPanel>

<StackPanel Margin="10,0,0,0" Orientation="Horizontal">

<Label Content="Port:" />

<TextBox

Width="70"

Height="23"

Template="{StaticResource TextBoxBaseControlTemplate}"

Text="{Binding Port}" />

</StackPanel>

<StackPanel Margin="10,0,0,0" Orientation="Horizontal">

<Label Content="Имя:" />

<TextBox

Width="100"

Height="23"

Template="{StaticResource TextBoxBaseControlTemplate}"

Text="{Binding Login, UpdateSourceTrigger=PropertyChanged}" />

</StackPanel>

<StackPanel

HorizontalAlignment="Right"

VerticalAlignment="Center"

Orientation="Horizontal">

<Button

Command="{Binding DisconnectCommand}"

Content="Отключиться"

IsEnabled="{Binding CanDisconnect}"

Style="{StaticResource CommandButton}" />

<Button

Command="{Binding ConnectCommand}"

Content="Подключиться"

IsEnabled="{Binding Connected}"

Style="{StaticResource CommandButton}" />

<Ellipse Width="30" Height="30">

<Ellipse.Style>

<Style TargetType="{x:Type Ellipse}">

<Setter Property="Fill" Value="Gray" />

<Setter Property="Stroke" Value="White" />

<Setter Property="StrokeThickness" Value="2" />

<Style.Triggers>

<DataTrigger Binding="{Binding CanDisconnect}" Value="True">

<Setter Property="Fill" Value="Green" />

</DataTrigger>

</Style.Triggers>

</Style>

</Ellipse.Style>

</Ellipse>

</StackPanel>

</DockPanel>

<DockPanel Grid.Row="1">

<StackPanel>

<Label x:Name="getKeysLabel" Content="Для начала работы вам необходимо получить ключи " HorizontalAlignment="Center" Margin="0,10,0,0" Width="423" VerticalAlignment="Top" Grid.Row="1" Opacity="1" RenderTransformOrigin="0.5,0.5">

<Label.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Label.RenderTransform>

</Label>

<Button

x:Name="getKeysButton"

Content="Получить ключи"

Width="138"

Opacity="1"

Click="GetKeysButton\_Click" RenderTransformOrigin="0.5,0.5" >

<Button.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Button.RenderTransform>

</Button>

</StackPanel>

</DockPanel>

<DockPanel Grid.Row="3" Margin="5,10,5,5">

<ScrollViewer x:Name="Scroller" VerticalScrollBarVisibility="Auto">

<ItemsControl x:Name="MessagesItems" ItemsSource="{Binding Messages}">

<ItemsControl.ItemTemplate>

<DataTemplate>

<Grid Margin="10,0,0,10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="80" />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<StackPanel MinWidth="120">

<TextBlock Style="{StaticResource AuthorStyle}" Text="{Binding Login}" />

<TextBlock Style="{StaticResource SendTimeStyle}" Text="{Binding SendTime, Converter={StaticResource DateTimeConverter}}" />

</StackPanel>

<Border

Grid.Column="1"

Margin="20,0,0,0"

HorizontalAlignment="Left"

Background="#FF727980"

BorderBrush="{Binding Background, RelativeSource={RelativeSource Mode=Self}}"

BorderThickness="4"

CornerRadius="4">

<TextBlock Style="{StaticResource MessageStyle}" Text="{Binding Content}" />

</Border>

</Grid>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</ScrollViewer>

</DockPanel>

<Grid Grid.Row="4" Margin="10,10,10,10">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition Width="auto" MinWidth="258" />

</Grid.ColumnDefinitions>

<TextBox

VerticalContentAlignment="Center"

IsEnabled="{Binding CanDisconnect}"

Template="{StaticResource TextBoxBaseControlTemplate}"

Text="{Binding Text, UpdateSourceTrigger=PropertyChanged}" Height="30" Margin="0,0,332,0" Grid.ColumnSpan="2" />

<Button

Grid.Column="1" IsDefault="True"

Command="{Binding SendMessageCommand}"

IsEnabled="{Binding Text, Converter={StaticResource EmptyTextConverter}}"

Style="{StaticResource SendButton}" Margin="299,5,173,5" Content="Отправить сообщение"/>

<Button

Grid.Column="2"

IsEnabled="{Binding CanDisconnect}"

IsDefault="True"

Content="Отправить файлы"

HorizontalAlignment="Left"

Margin="461,0,0,0"

VerticalAlignment="Center"

Style="{StaticResource SendButton}"

Command="{Binding SendFileCommand}"

Height="32"

Width="165"

Grid.ColumnSpan="2" />

</Grid>

<Label x:Name="label" Content="Ключи подключения" HorizontalAlignment="Left" Margin="292,-156,0,0" Width="171" VerticalAlignment="Top" RenderTransformOrigin="0.5,0.5">

<Label.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Label.RenderTransform>

</Label>

<Label x:Name="welcomeLabel" Content="Добро пожаловать в приложени StealthForce" HorizontalAlignment="Center" Width="439" Opacity="1" RenderTransformOrigin="0.5,0.5" HorizontalContentAlignment="Center">

<Label.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Label.RenderTransform>

</Label>

</Grid>

</Window>

**Файл MainWindow.xaml.cs**

using EncryptedChat.Server.ClientModel;

using Microsoft.Win32;

using System;

using System.Collections.Specialized;

using System.IO;

using System.Net.Sockets;

using System.Runtime.InteropServices.ComTypes;

using System.Runtime.Serialization.Formatters.Binary;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media.Animation;

namespace EncryptedChat.Client.View

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

Loaded += (s, e) => ((INotifyCollectionChanged)MessagesItems.ItemsSource).CollectionChanged += (q, z) => Scroller.ScrollToEnd();

}

private void CloseWindow(object sender, RoutedEventArgs e)

{

SystemCommands.CloseWindow(this);

}

private void Storyboard1\_Completed(object sender, EventArgs e)

{

label.Visibility = Visibility.Hidden;

welcomeLabel.Visibility = Visibility.Hidden;

getKeysButton.Visibility = Visibility.Hidden;

}

private void MaximizeWindow(object sender, RoutedEventArgs e)

{

if (WindowState == WindowState.Normal)

SystemCommands.MaximizeWindow(this);

else

SystemCommands.RestoreWindow(this);

}

private void MinimizeWindow(object sender, RoutedEventArgs e)

{

SystemCommands.MinimizeWindow(this);

}

private void DockPanel\_MouseDown(object sender, MouseButtonEventArgs e)

{

Application.Current.MainWindow.DragMove();

}

// Not correct

private void TextBox\_TextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

var text = (sender as TextBox).Text;

var result = string.Empty;

foreach (var ch in text)

{

if (char.IsDigit(ch))

result += ch;

}

(sender as TextBox).Text = result;

}

private void GetKeysButton\_Click(object sender, RoutedEventArgs e)

{

Storyboard storyboard = (Storyboard)FindResource("Storyboard1");

storyboard.Begin(dockPanel);

DoubleAnimation welcomeLabelAnimation = new DoubleAnimation

{

To = 0,

Duration = TimeSpan.FromSeconds(0.5)

};

DoubleAnimation getKeysLabelAnimation = new DoubleAnimation

{

To = 0,

Duration = TimeSpan.FromSeconds(0.5)

};

DoubleAnimation getKeysButtonAnimation = new DoubleAnimation

{

To = 0,

Duration = TimeSpan.FromSeconds(0.5)

};

storyboard.Children.Add(welcomeLabelAnimation);

storyboard.Children.Add(getKeysLabelAnimation);

storyboard.Children.Add(getKeysButtonAnimation);

Storyboard.SetTargetName(welcomeLabelAnimation, "welcomeLabel");

Storyboard.SetTargetProperty(welcomeLabelAnimation, new PropertyPath(OpacityProperty));

Storyboard.SetTargetName(getKeysLabelAnimation, "getKeysLabel");

Storyboard.SetTargetProperty(getKeysLabelAnimation, new PropertyPath(OpacityProperty));

Storyboard.SetTargetName(getKeysButtonAnimation, "getKeysButton");

Storyboard.SetTargetProperty(getKeysButtonAnimation, new PropertyPath(OpacityProperty));

storyboard.Begin();

// Отключить кнопку "Получить ключи"

getKeysButton.IsEnabled = false;

}

private void HelpButton\_Click(object sender, RoutedEventArgs e)

{

HelpWindow helpWindow = new HelpWindow();

helpWindow.ShowDialog();

}

}

}

**Файл MainViewModel.cs**

using Encrypted;

using EncryptedChat.Server.ClientModel;

using Microsoft.Win32;

using Prism.Commands;

using System;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Runtime.Serialization.Formatters.Binary;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Input;

using System.Windows.Threading;

namespace EncryptedChat.Client.ViewModels

{

public class MainViewModel : INotifyPropertyChanged

{

private RSA \_rsa = new RSA();

private TcpClient \_tcpClient;

private NetworkStream \_stream;

private ConnectedClient \_client;

#region Keys

public int LocalE => (int)\_rsa.e;

public int LocalN => (int)\_rsa.n;

public int RemoteE { get => remoteE; set { remoteE = value; OnPropertyChanged(); } }

public int RemoteN { get => remoteN; set { remoteN = value; OnPropertyChanged(); } }

#endregion

public string Host { get; set; }

public int Port { get; set; }

public string Login { get; set; } = "";

public bool CanDisconnect { get; set; } = false;

public ICommand SendFileCommand { get; set; }

public ICommand SendMessageCommand { get; set; }

public ICommand ConnectCommand { get; set; }

public ICommand DisconnectCommand { get; set; }

public string Text

{

get => \_text;

set

{

\_text = value;

OnPropertyChanged();

}

}

public bool Connected => RemoteE != 0 && RemoteN != 0 && \_canConnect;

public ObservableCollection<MessageItem> Messages { get; set; }

private bool \_canConnect = true;

private int remoteE;

private int remoteN;

private string \_text;

public MainViewModel(IPAddress host, int port)

{

Host = host.ToString();

Port = port;

Messages = new ObservableCollection<MessageItem>();

SendFileCommand = new DelegateCommand(SendFile);

SendMessageCommand = new DelegateCommand(() => SendMessage(Text));

ConnectCommand = new DelegateCommand(Connect);

DisconnectCommand = new DelegateCommand(Disconnect);

}

private void Disconnect()

{

try

{

if (\_tcpClient?.Connected == true)

{

// Создаем объект MessageItem

var messageItem = new MessageItem

{

Content = "Пользователь отключился от сервера",

SendTime = DateTime.Now,

Login = "Сервер"

};

// Отправляем MessageItem на сервер

new BinaryFormatter().Serialize(\_stream, messageItem);

// Добавляем MessageItem в локальный чат

Application.Current.Dispatcher.Invoke(() =>

{

Messages.Add(messageItem);

});

}

\_tcpClient?.Close();

\_canConnect = true;

CanDisconnect = false;

OnPropertyChanged();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка отключения: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Connect()

{

if (string.IsNullOrEmpty(Login))

{

MessageBox.Show("Введите имя пользователя!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

try

{

// Если уже подключены, разрываем соединение

if (\_tcpClient?.Connected == true)

{

Disconnect();

}

\_tcpClient = new TcpClient();

// Попытка подключения

\_tcpClient.Connect(Host, Port);

\_stream = \_tcpClient.GetStream();

// Создание объекта клиента

\_client = new ConnectedClient(GetLocalIPAddress().ToString(), Login);

// Отправка информации о клиенте на сервер

var bf = new BinaryFormatter();

bf.Serialize(\_stream, \_client);

// Запуск задачи для приема сообщений от сервера

Task.Factory.StartNew(() =>

{

try

{

// Прием данных от сервера

while (\_tcpClient.Connected)

{

var serverObject = bf.Deserialize(\_stream);

// Обработка полученных объектов

if (serverObject is MessageItem messageItem)

{

Application.Current.Dispatcher.Invoke(() =>

{

Messages.Add(messageItem);

// Проверка на сообщение о дублировании логина

if (messageItem.Content.Contains("Пользователь с таким логином уже подключен"))

{

Disconnect();

MessageBox.Show("Пользователь с таким логином уже подключен. Пожалуйста, выберите другой логин.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

});

}

else if (serverObject is ConnectedClient conClient)

{

\_canConnect = false;

CanDisconnect = true;

Application.Current.Dispatcher.Invoke(() =>

{

Messages.Add(new MessageItem()

{

Login = "Сервер", // Имя "Сервер" для сообщений от сервера

Content = $"Новое подключение: {conClient.Login}",

SendTime = DateTime.Now

});

});

OnPropertyChanged();

}

else if (serverObject is EncryptedObject encObj)

{

if (encObj.Client.Login == \_client.Login)

continue;

Application.Current.Dispatcher.Invoke(() =>

{

var text = \_rsa.Decrypt(encObj.Message.Content);

Messages.Add(encObj.MapEncObjToMessageItem(text));

}, DispatcherPriority.Background);

}

else if (serverObject is FileTransferObject fileTransferObject)

{

// Получение файла

Application.Current.Dispatcher.Invoke(() =>

{

// Создание папки "Файлы" на рабочем столе, если она не существует

string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

string filesFolderPath = Path.Combine(desktopPath, "Файлы");

if (!Directory.Exists(filesFolderPath))

{

Directory.CreateDirectory(filesFolderPath);

}

// Сохранение файла

string filePath = Path.Combine(filesFolderPath, fileTransferObject.FileName);

File.WriteAllBytes(filePath, fileTransferObject.Content);

// Оповещение о получении файла

Messages.Add(new MessageItem() { Content = $"Получен файл: {fileTransferObject.FileName}", SendTime = DateTime.Now, Login = "Сервер" });

});

}

}

}

catch (Exception e)

{

ShowDisconnectMessage();

Console.WriteLine(e.Message);

\_canConnect = true;

CanDisconnect = false;

OnPropertyChanged();

}

}, TaskCreationOptions.LongRunning);

}

catch (Exception e)

{

MessageBox.Show(e.Message, "Ошибка подключения к серверу", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

}

private void ShowDisconnectMessage()

{

Application.Current.Dispatcher.Invoke(() =>

{

Messages.Add(new MessageItem() { Content = "Соединение разоравно...", SendTime = DateTime.Now, Login = "Сервер" });

});

}

private void SendMessage(string messageText)

{

try

{

var data = \_rsa.Encrypt(messageText, RemoteE, RemoteN);

Dispatcher.CurrentDispatcher.Invoke(() =>

{

Messages.Add(new MessageItem() { Login = \_client.Login, Content = messageText, SendTime = DateTime.Now });

});

var message = new EncryptedObject(new Message(data, DateTime.Now), \_client);

new BinaryFormatter().Serialize(\_stream, message);

Text = string.Empty;

OnPropertyChanged(nameof(Text));

}

catch (Exception e)

{

MessageBox.Show(e.Message, "Ошибка отправки сообщения", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

}

private void SendFile()

{

try

{

// Определение пути к рабочему столу

string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

// Создание папки "Файлы" на рабочем столе, если она не существует

string filesFolderPath = Path.Combine(desktopPath, "Файлы");

if (!Directory.Exists(filesFolderPath))

{

Directory.CreateDirectory(filesFolderPath);

}

// Показываем диалоговое окно выбора файла

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

// Читаем файл в байтовый массив

byte[] fileBytes = File.ReadAllBytes(openFileDialog.FileName);

// Отправка файла

var fileTransferObject = new FileTransferObject { FileName = Path.GetFileName(openFileDialog.FileName), Content = fileBytes };

var bf = new BinaryFormatter();

bf.Serialize(\_stream, fileTransferObject);

// Оповещаем пользователя об успешной отправке файла

MessageBox.Show("Файл успешно отправлен.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

// Обрабатываем возможные ошибки

MessageBox.Show($"Ошибка отправки файла: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private static IPAddress GetLocalIPAddress()

{

var host = Dns.GetHostEntry(Dns.GetHostName());

foreach (var ip in host.AddressList)

if (ip.AddressFamily == AddressFamily.InterNetwork)

return ip;

throw new Exception("No network adapters with an IPv4 address in the system!");

}

public void OnPropertyChanged(string name = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));

}

public event PropertyChangedEventHandler PropertyChanged;

}

}

**Файл App.xamlы**

<Application

x:Class="EncryptedChat.Client.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Startup="OnStartup">

<Application.Resources />

</Application>

**Файл App.xaml.cs**

using System.Net;

using System.Net.Sockets;

using EncryptedChat.Client.ViewModels;

using System.Windows;

using System;

using EncryptedChat.Client.View;

namespace EncryptedChat.Client

{

public partial class App : Application

{

protected void OnStartup(object sender, StartupEventArgs e)

{

MainViewModel mainVM;

if (e.Args.Length == 2 &&

IPAddress.TryParse(e.Args[0], out IPAddress host) &&

int.TryParse(e.Args[1], out int port))

{

mainVM = new MainViewModel(host, port);

}

else

{

mainVM = new MainViewModel(GetLocalIPAddress(), 5050);

}

new MainWindow() { DataContext = mainVM }.ShowDialog();

}

private static IPAddress GetLocalIPAddress()

{

var host = Dns.GetHostEntry(Dns.GetHostName());

foreach (var ip in host.AddressList)

{

if (ip.AddressFamily == AddressFamily.InterNetwork)

{

return ip;

}

}

throw new Exception("No network adapters with an IPv4 address in the system!");

}

}

}

**Приложение 2**

Код серверного приложения

**Файл BaseClient.cs**

using System;

namespace EncryptedChat.Server.ClientModel

{

[Serializable]

public abstract class BaseClient

{

public Guid ID { get; protected set; }

public string Login { get; protected set; }

}

}

Файл ConnectedClient.cs

using System;

namespace EncryptedChat.Server.ClientModel

{

[Serializable]

public class ConnectedClient : BaseClient

{

public ConnectedClient(string source, string login)

{

Source = source ?? throw new ArgumentNullException(nameof(source));

Login = login ?? throw new ArgumentNullException(nameof(login));

}

public virtual string Source { get; }

}

}

**Файл EncryptedObject.cs**

using System;

namespace EncryptedChat.Server.ClientModel

{

[Serializable]

public class EncryptedObject

{

public EncryptedObject(Message message, ConnectedClient client)

{

Message = message;

Client = client;

}

public Message Message { get; }

public ConnectedClient Client { get; }

}

[Serializable]

public class Message

{

public Message(string[] content, DateTime sendTime)

{

Content = content;

SendTime = sendTime;

}

public string[] Content { get; set; }

public DateTime SendTime { get; set; }

}

[Serializable]

public class FileTransferObject

{

public string FileName { get; set; }

public byte[] Content { get; set; }

}

[Serializable]

public class MessageItem

{

public string Content { get; set; }

public DateTime SendTime { get; set; }

public string Login { get; set; }

}

}

**Файл Extensions.cs**

namespace EncryptedChat.Server.ClientModel

{

public static class Extensions

{

public static MessageItem MapEncObjToMessageItem(this EncryptedObject encObj, string text)

{

return new MessageItem()

{

Content = text,

Login = encObj.Client.Login,

SendTime = encObj.Message.SendTime

};

}

}

}

**Файл ServerClient.cs**

using System;

using System.Net.Sockets;

namespace EncryptedChat.Server.ClientModel

{

public class ServerClient : BaseClient

{

public ServerClient()

{

}

public ServerClient(TcpClient client, Guid id, string login)

{

Client = client ?? throw new ArgumentNullException(nameof(client));

Login = login ?? throw new ArgumentNullException(nameof(login));

ID = id;

}

public TcpClient Client { get; }

}

}

**Файл Bootstrapper.cs**

using System;

using System.Net;

using System.Net.Sockets;

namespace EncryptedChat.Server

{

internal class Bootstrapper

{

static void Main(string[] args)

{

if (args.Length == 2 &&

IPAddress.TryParse(args[0], out IPAddress host) &&

int.TryParse(args[1], out int port))

{

new Server(host, port).Start();

}

else

{

new Server(GetLocalIPAddress(), 5050).Start();

}

}

public static IPAddress GetLocalIPAddress()

{

var host = Dns.GetHostEntry(Dns.GetHostName());

foreach (var ip in host.AddressList)

if (ip.AddressFamily == AddressFamily.InterNetwork)

return ip;

throw new Exception("No network adapters with an IPv4 address in the system!");

}

}

}

**Файл Server.cs**

using EncryptedChat.Server.ClientModel;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Net;

using System.Net.Sockets;

using System.Runtime.Serialization.Formatters.Binary;

using System.Threading;

using System.Threading.Tasks;

namespace EncryptedChat.Server

{

internal class Server

{

static TcpListener \_listener;

static List<ServerClient> \_clients;

static Stopwatch \_uptime;

internal Server(IPAddress host, int port)

{

\_listener = new TcpListener(host, 5050);

\_clients = new List<ServerClient>();

\_uptime = new Stopwatch();

Console.WriteLine($"Создание сервера на: {host}:{port}");

}

public void Start()

{

try

{

\_listener.Start();

\_uptime.Start();

Console.WriteLine("Сервер запущен");

Console.WriteLine("Для обращения к справке команда help");

}

catch (Exception e)

{

Console.WriteLine($"Не удалось запустить сервер: {e.Message}");

Console.ReadKey();

Environment.Exit(0);

}

// Запуск задачи для обработки консольных команд

Task.Run(() => HandleConsoleCommands());

try

{

while (true)

{

var client = \_listener.AcceptTcpClient();

Task.Factory.StartNew(() =>

{

var stream = client.GetStream();

var bf = new BinaryFormatter();

var serverClient = new ServerClient();

try

{

var conClient = bf.Deserialize(stream) as ConnectedClient;

if (\_clients.Exists(c => c.Login == conClient.Login))

{

// Обработка случая, если клиент с таким логином уже подключен

var messageItem = new MessageItem

{

Content = "Пользователь с таким логином уже подключен",

SendTime = DateTime.Now,

Login = "Сервер"

};

bf.Serialize(client.GetStream(), messageItem);

client.Close();

return;

}

else

{

serverClient = new ServerClient(client, conClient.ID, conClient.Login);

\_clients.Add(serverClient);

WriteSignalAboutConnection(conClient);

SendToAllClients(conClient);

}

while (client.Connected)

{

if (stream.DataAvailable)

{

var serverObject = bf.Deserialize(stream);

if (serverObject is EncryptedObject encObject)

{

WriteTextToConsole(encObject);

SendToAllClients(encObject);

}

else if (serverObject is FileTransferObject fileTransferObject)

{

Console.WriteLine($"Получен файл: {fileTransferObject.FileName} от {serverClient.Login}");

SendToAllClients(fileTransferObject);

}

else if (serverObject is MessageItem messageItem && messageItem.Content == "Пользователь отключился от сервера")

{

// Клиент отключился

Console.WriteLine($"{serverClient.Login} отключился.");

\_clients.Remove(serverClient);

SendToAllClients(new MessageItem

{

Content = $"{serverClient.Login} отключился от сервера",

SendTime = DateTime.Now,

Login = "Сервер"

});

client.Close();

break;

}

}

}

}

catch (Exception e)

{

// Обработка ошибки чтения или отключения клиента

Console.WriteLine($"Ошибка в соединении с клиентом: {e.Message}");

if (client.Connected)

{

client.Close();

}

\_clients.Remove(serverClient);

SendToAllClients(new MessageItem

{

Content = $"{serverClient.Login} отключился от сервера",

SendTime = DateTime.Now,

Login = "Сервер"

});

}

}, TaskCreationOptions.LongRunning);

}

}

catch (Exception e)

{

Console.WriteLine($"FATAL ERROR: {e.Message}");

}

}

static async void SendToAllClients(object serverObj)

{

await Task.Factory.StartNew(() =>

{

var bf = new BinaryFormatter();

lock (\_clients)

{

foreach (var client in \_clients.ToArray()) // Создаем копию списка для безопасного перебора

{

try

{

if (client.Client.Connected)

{

bf.Serialize(client.Client.GetStream(), serverObj);

}

else

{

\_clients.Remove(client);

}

}

catch (Exception e)

{

// Обработка ошибки отправки

Console.WriteLine($"Ошибка отправки данных клиенту: {e.Message}");

client.Client.Client.Disconnect(false);

\_clients.Remove(client);

}

}

}

});

}

private static void WriteExceptionToConsole(Exception e)

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(e.Message);

Console.ForegroundColor = ConsoleColor.White;

}

private static void WriteTextToConsole(EncryptedObject eObj)

{

Console.WriteLine($"{eObj.Client.Login} " +

$"[{eObj.Message.SendTime.ToString()}]: " +

$"{string.Join(" ", eObj.Message.Content)}");

}

private static void WriteSignalAboutConnection(ConnectedClient connectedClient)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.Write($"New connection: ");

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine($"{connectedClient.Login} - {connectedClient.Source}");

}

// Метод для обработки консольных команд

private static void HandleConsoleCommands()

{

while (true)

{

string command = Console.ReadLine();

switch (command.ToLower())

{

case "stats":

ShowServerStats();

break;

case "connections":

ShowActiveConnections();

break;

case "help":

ShowHelp();

break;

default:

Console.WriteLine("Неизвестная команда. Введите 'help' для справки.");

break;

}

}

}

// Вывод статистики сервера

private static void ShowServerStats()

{

Console.WriteLine("Статистика сервера:");

Console.WriteLine($"Время работы: {\_uptime.Elapsed}");

Console.WriteLine($"Количество подключений: {\_clients.Count}");

// Добавьте другую статистику по мере необходимости

}

// Вывод списка активных подключений

private static void ShowActiveConnections()

{

Console.WriteLine("Активные подключения:");

if (\_clients.Count == 0)

{

Console.WriteLine("Нет активных подключений.");

}

else

{

foreach (var client in \_clients)

{

Console.WriteLine($"- {client.Login} ({client.Client.Client.RemoteEndPoint})");

}

}

}

private static void ShowHelp()

{

Console.WriteLine("Доступные команды:");

Console.WriteLine(" stats - Показать статистику сервера");

Console.WriteLine(" connections - Показать список активных подключений");

Console.WriteLine(" help - Показать эту справку");

Console.WriteLine();

Console.WriteLine("Программа разработана студентом 4 курса Катаргиным Никитой");

Console.WriteLine("Поддержка: katargin.04@mail.ru");

}

}

}

**Приложение 3**

Код RSA шифрования

**Файл NumericExtensions.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Encrypted

{

public static class NumericExtensions

{

public static bool IsSimpleNum(this long num)

{

if (num < 2)

return false;

if (num == 2)

return true;

for (long i = 2; i < num; i++)

if (num % i == 0)

return false;

return true;

}

}

}

**Файл PrimeNumberGenerator.cs**

using System;

using System.Collections.Generic;

namespace Encrypted

{

internal class PrimeNumberGenerator

{

private static Random \_random = new Random();

public static long Generate()

{

var values = new List<int>();

int left = 500, right = 1500;

while (left++ < right)

{

if (IsPrime(left))

values.Add(left);

}

return values[\_random.Next(0, values.Count)];

}

public static bool IsPrime(long n)

{

if (n == 1)

return false;

for (int d = 2; d \* d <= n; d++)

if (n % d == 0)

return false;

return true;

}

}

}

**Файл RSA.cs**

using System;

using System.Collections.Generic;

using System.Numerics;

using System.Text;

namespace Encrypted

{

public class RSA

{

private bool \_isReady;

private long local\_D;

public long p;

public long q;

public long n;

public long e;

public RSA()

{

Initialize();

}

public void Initialize()

{

p = PrimeNumberGenerator.Generate();

q = PrimeNumberGenerator.Generate();

n = p \* q;

var fi = (p - 1) \* (q - 1);

e = GetPublicPartKey(fi);

local\_D = GetPrivatePartKey(fi, e);

\_isReady = true;

}

public string[] Encrypt(string text, long publicE, long publicN)

{

if (!\_isReady)

throw new ArgumentException("Метод инициализации не вызывается");

return Encode(text, publicE, publicN);

}

public string Decrypt(string[] data)

{

if (!\_isReady)

throw new ArgumentException("Метод инициализации не вызывается");

return Decode(data, local\_D, n);

}

private string[] Encode(string text, long e, long n)

{

var data = new List<string>();

BigInteger num;

foreach (var ch in text)

{

int index = ch;

num = BigInteger.ModPow(index, e, n);

data.Add(num.ToString());

}

return data.ToArray();

}

private string Decode(string[] data, long d, long n)

{

var strBuilder = new StringBuilder();

BigInteger num;

foreach (var item in data)

{

var val = new BigInteger(Convert.ToInt64(item));

num = BigInteger.ModPow(val, d, n);

strBuilder.Append((char)num);

}

return strBuilder.ToString();

}

private long GetPrivatePartKey(long fi, long e)

{

long d = e + 1;

while (true)

{

if ((d \* e) % fi == 1)

break;

d++;

}

return d;

}

private long GetPublicPartKey(long fi)

{

long e = fi - 1;

while (true)

{

if (PrimeNumberGenerator.IsPrime(e) &&

e < fi &&

BigInteger.GreatestCommonDivisor(new BigInteger(e), new BigInteger(fi)) == BigInteger.One)

break;

e--;

}

return e;

}

}

}

**Приложение 4**

Компакт-диск с материалами проекта

На диске располагается:

* Проект портативного приложения
* Проект серверного приложения
* Установщик портативного и серверного приложения
* Файл дипломного проекта в формате MS Word