**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

**Институт (Филиал)** № 8 «Компьютерные науки и прикладная математика» **Кафедра**  806

**Группа** М8О-411Б-19  **Направление подготовки** 02.03.02 Фундаментальная информатика и информационные технологии

**Профиль**  Информатика и компьютерные науки

**Квалификация**  **бакалавр**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

На тему: Распределенная система криптографической защиты данных на основе эллиптических кривых

Автор ВКРБ Тришин Дмитрий Александрович (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Руководитель Романенков Александр Михайлович (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Консультант (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Консультант (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

Рецензент (\_\_\_\_\_\_\_\_\_\_)

(фамилия, имя, отчество полностью)

**К защите допустить**

Заведующий кафедрой 806 Крылов Сергей Сергеевич (\_\_\_\_\_\_\_\_\_)

(№ каф) (фамилия, имя, отчество полностью)

\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023г.

Москва 2023

**РЕФЕРАТ**

Выпускная квалификационная работа бакалавра состоит из 54 страницы, 19 рисунков, 00 использованных источников.

РАСПРЕДЕЛЕННАЯ СИСТЕМА, ЭЛЛИПТИЧЕСКАЯ КРИПТОГРАФИЯ, ЭЛЛИПТИЧЕСКАЯ КРИВАЯ, ЭЛЕКТРОННАЯ ПОДПИСЬ ДОКУМЕНТА, МИКРОСЕРВИС, ASP.NET, REST API, POSTGRESQL, WPF, DOCKER.

Целью данной работы является исследование применения криптографических алгоритмов шифрования на основе эллиптических кривых для защиты данных и реализация распределенной системы криптографической защиты данных с использованием разработанных инструментов для работы с эллиптическим кривыми. Были использованы материалы о существующих алгоритмах шифрования и о применении этих алгоритмов в существующих системах, использующих эти криптографические алгоритмы для шифрования и дальнейшего хранения данных. Для реализации был использован язык программирования C#, фреймворки ASP.NET и WPF. В результате было разработана библиотека исходного кода для работы с эллиптическими кривыми и клиент-серверное приложение мессенджера, состоящее из распределенной системы, использующей алгоритм шифрования на основе эллиптических кривых для защиты данных пользователей, и приложения для персональных компьютеров для взаимодействия с распределенной системой. Разработанное приложение может использоваться для обеспечения безопасности общения и хранения данных компаний.

СОДЕРЖАНИЕ

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 5](#_Toc135597639)

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 6](#_Toc135597640)

[ВВЕДЕНИЕ 7](#_Toc135597641)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 11](#_Toc135597642)

[1.1 Криптографические алгоритмы шифрования 11](#_Toc135597643)

[1.1.1 Асимметричные алгоритмы шифрования 12](#_Toc135597644)

[1.1.2 Асимметричные алгоритмы шифрования 14](#_Toc135597645)

[1.1.3 Алгоритмы шифрования на эллиптических кривых 16](#_Toc135597646)

[1.2 Распределенные системы 18](#_Toc135597647)

[1.2.1 Сведения о распределенных системах 18](#_Toc135597648)

[1.2.2 Структура распределенной системы 19](#_Toc135597649)

[1.2.3 Виды распределенных систем 21](#_Toc135597650)

[1.3 Исследование существующих распределенных криптосистем 22](#_Toc135597651)

[2 РАЗРАБОТКА РЕШЕНИЯ 24](#_Toc135597652)

[2.1 Описание решения поставленной задачи 24](#_Toc135597653)

[2.1.1 Математическая библиотека 24](#_Toc135597654)

[2.1.2 Криптосистема 25](#_Toc135597655)

[2.2 Выбор программных средств разработки 27](#_Toc135597656)

[2.2.1 C# 27](#_Toc135597657)

[2.2.2 ASP.NET 28](#_Toc135597658)

[2.2.3 WPF 30](#_Toc135597659)

[2.2.4 PostgreSQL 31](#_Toc135597660)

[2.2.5 Docker и Docker Compose 32](#_Toc135597661)

[2.3 Описание программной разработки 33](#_Toc135597662)

[2.3.1 Распределенная система 33](#_Toc135597663)

[2.3.2 REST API сервис 34](#_Toc135597664)

[2.3.3 Сервис баз данных 36](#_Toc135597665)

[2.3.4 Сервис файлового хранилища 37](#_Toc135597666)

[3 РЕЗУЛЬТАТЫ РАБОТЫ 39](#_Toc135597667)

[3.1 Характеристика условий и места применения разработки 39](#_Toc135597668)

[3.2 Результаты работы разработанного программного кода 40](#_Toc135597669)

[3.2.1 Серверное приложение 40](#_Toc135597670)

[3.2.1.1 Сервис основного приложения 40](#_Toc135597671)

[3.2.1.2 Сервис файлового хранилища 43](#_Toc135597672)

[3.2.1.3 Сервис электронной подписи 45](#_Toc135597673)

[3.2.2 Клиентское приложение 46](#_Toc135597674)

[3.2.2.1 Авторизация 46](#_Toc135597675)

[3.2.2.2 Чат 50](#_Toc135597676)

[3.2.2.3 Документы и электронная подпись 52](#_Toc135597677)

[3.3 Технические характеристики разработанного приложения 53](#_Toc135597678)

[ЗАКЛЮЧЕНИЕ 55](#_Toc135597679)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 57](#_Toc135597680)

# **ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

В настоящей выпускной квалификационной работе бакалавра применяют следующие термины с соответствующими определениями:

|  |  |
| --- | --- |
| *C#* | *– определение* |
| *.NET* | *– определение* |
| *ASP.NET* | *– определение* |
| *WPF* | *– определение* |
| *Docker* | *– определение* |
| *Docker Compose* | *– определение* |
| *REST API* | *– определение* |
| *Entity Framework* | *– определение* |

# **ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

В настоящей выпускной квалификационной работе бакалавра применяют следующие сокращения и обозначения:

|  |  |
| --- | --- |
| *ЭП* | *– Электронная подпись* |
| *Аббревиатура* | *– детальная расшифровка* |
| *Аббревиатура* | *– детальная расшифровка* |
| *Аббревиатура* | *– детальная расшифровка* |
| *Обозначение* | *– детальная расшифровка* |
| *Обозначение* | *– детальная расшифровка* |
| *Обозначение* | *– детальная расшифровка* |
| *Обозначение* | *– детальная расшифровка* |
| *Обозначение* | *– детальная расшифровка* |
| *Обозначение* | *– детальная расшифровка* |
| *Обозначение* | *– детальная расшифровка* |
| *Обозначение* | *– детальная расшифровка* |

# **ВВЕДЕНИЕ**

В современном мире, который быстро и неумолимо изменяется, данные играют все более важную роль. Мы живем в эпоху, когда объем данных, которые мы производим, растет экспоненциально. Не только компьютерные системы и интернет генерируют огромные объемы данных, но и мобильные устройства, социальные сети, онлайн-торговые платформы и другие технологии собирают и хранят данные о нас и нашем окружении.

Данные имеют огромное значение, потому что они могут дать нам важную информацию о нашем мире, помочь нам принимать решения, оптимизировать процессы и увеличивать эффективность наших действий. Данные используются в различных сферах деятельности, таких как экономика, медицина, наука, технологии, государственное управление, и многие другие.

Одним из ключевых аспектов данных является их количественная и качественная характеристика. В современном мире существует огромное количество данных, их обработка и интерпретация требуют высококвалифицированных специалистов и современных технологий. Правильно обработанные данные могут помочь нам принимать более обоснованные решения, оптимизировать бизнес-процессы, повышать эффективность производственных и научных процессов, а также создавать новые продукты и услуги.

Важным аспектом данных является их интерпретация и использование. Данные могут быть использованы для обнаружения закономерностей и тенденций, которые могут помочь нам принимать более обоснованные решения. Однако, неправильное использование данных или неправильная их интерпретация может привести к ошибкам и нежелательным последствиям. Поэтому, правильная обработка и интерпретация данных является критически важным элементом в их значимости для нашей жизни.

Несмотря на значимость данных в нашей жизни, существуют также определенные ограничения и проблемы, связанные с их использованием. Одна из главных проблем заключается в защите данных и приватности. С постоянным увеличением объема данных и их значимости, возрастает их уязвимость, и проблема безопасности данных становится все более актуальной.

Данные могут быть украдены, изменены или уничтожены злоумышленниками, что может привести к серьезным последствиям, включая финансовые потери, утечку конфиденциальной информации и нарушение прав человека. Поэтому безопасность данных становится ключевой проблемой, требующей серьезного внимания и многогранного подхода.

Актуальность темы данной работы связана с увеличивающейся необходимостью защиты данных в интернете. В современном мире большинство процессов осуществляется через компьютеры и Интернет, и большое количество конфиденциальных данных передается по сети. Однако это также означает, что данные могут быть украдены или скомпрометированы злоумышленниками.

Для защиты этих данных используются криптографические методы, такие как шифрование. Одним из наиболее эффективных методов шифрования является криптография на основе эллиптических кривых. Этот метод обеспечивает высокую стойкость к взлому, что делает его идеальным для защиты конфиденциальных данных в распределенной среде.

Кроме того, в современном мире распределенные системы становятся все более популярными. Эти системы включают в себя несколько компьютеров, которые работают вместе для выполнения задач. Криптографическая защита данных в распределенных системах также становится все более важной, так как она может быть наиболее уязвимой точкой в системе.

Таким образом, разработка распределенной системы криптографической защиты данных на основе эллиптических кривых является актуальной темой, которая может помочь обеспечить безопасность данных в распределенных системах и защитить конфиденциальные данные от несанкционированного доступа.

Цель работы – создание математической библиотеки исходного кода для работы с эллиптическими кривыми и разработка распределенной системы клиент-серверного приложения. Для достижения поставленной цели в работе были решены следующие задачи:

1. изучение теоретического материала по эллиптическим кривым над полями конечной характеристики;
2. написание математической библиотеки исходного кода для работы с эллиптическими кривыми над полями конечной характеристики;
3. реализация симметричного и асимметричного алгоритма шифрования на основе эллиптических кривых для создания гибридной криптосистемы;
4. разработка с учетом вопрос балансировки нагрузки клиент-серверного приложения, применяющего разработанную криптосистему.

Работа основывалась на следующих инструментах и методах. Для разработки был выбран язык программирования C#. C# – это объектно-ориентированный язык общего назначения, разработанный компанией Microsoft в 2001 году. Клиент-серверное приложения задействует в себе два фреймворка. ASP.NET - платформа разработки веб-приложений, в состав которой входят: веб-сервисы, программная инфраструктура, модель программирования. ASP.NET используется для создания REST API приложения, отвечающего за основную логику приложения. Для хранения данных была выбрана реляционная база данных PostgreSQL. Для разработки клиентского приложения была выбрана система для построение приложений под систему Windows под названием Windows Presentation Foundation, сокращенно WPF. Для развертывания распределенной системы на серверной инфраструктуре был использован Docker – программное обеспечение для автоматизации развертывания и управления приложениями в средах и Docker Compose – инструментальное средство, входящее в состав утилиты Docker.

Основными результатами, полученными в работе, являются:

1. библиотека исходного кода для работы с эллиптическими кривыми над конечными полями;
2. распределенная система, состоящая из основного REST API приложения, сервера баз данных и сервера хранилища документов, использующее разработанную криптосистему для защиты данных;
3. приложение для ПК под управлением ОС Windows, реализующее функционал онлайн-чата.

Результаты работы предназначены для использования как в бизнес-сфере, так и в государственном управлении для обеспечения безопасности и конфиденциальности данных.

Использование разработки позволяет обеспечить высокий уровень защиты данных в распределенной среде. Система использует алгоритмы криптографической защиты данных на основе эллиптических кривых, которые обеспечивают высокую степень безопасности и эффективности при обработке больших объемов данных.

Кроме того, система обеспечивает возможность распределения данных между различными узлами сети, что позволяет обеспечить высокую отказоустойчивость и надежность при хранении и передаче данных. В целом, разработанная распределенная система позволяет обеспечить безопасное и эффективное хранение и передачу данных в распределенной среде.

# **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **Криптографические алгоритмы шифрования**

Криптографические алгоритмы шифрования — это методы преобразования информации из открытого вида в зашифрованный, защищенный от несанкционированного доступа и вмешательства. Они являются неотъемлемой частью современных систем безопасности и используются для защиты конфиденциальной информации в различных областях, таких как банковское дело, электронная коммерция, государственная безопасность, медицинские записи и многих других.

Криптографические алгоритмы шифрования позволяют обеспечить конфиденциальность, целостность и доступность данных. Они защищают данные от несанкционированного доступа и изменения, обеспечивая конфиденциальность передачи информации от отправителя к получателю. Также они обеспечивают аутентификацию, т.е. проверку подлинности отправителя и получателя информации.

Одним из основных принципов криптографии является использование ключей шифрования и дешифрования. Криптографический ключ — это секретный код или последовательность символов, используемая в криптографических алгоритмах для защиты конфиденциальности и целостности данных. Он служит для шифрования и дешифрования информации, а также для создания электронной подписи и проверки подлинности сообщения. Ключ может быть симметричным, когда используется один и тот же ключ для шифрования и дешифрования данных, или асимметричным, когда используется пара ключей — открытый и закрытый — для шифрования и дешифрования информации. Криптографический ключ является критически важным элементом в обеспечении безопасности данных и защите от несанкционированного доступа к информации. Ключи могут быть симметричными (когда один и тот же ключ используется для шифрования и дешифрования) и асимметричными (когда разные ключи используются для шифрования и дешифрования). Шифрование происходит с помощью алгоритма, который осуществляет преобразование исходной информации (открытого текста) в зашифрованный вид (шифрованный текст). Зашифрованный текст может быть передан по открытым каналам связи или храниться на ненадежных устройствах без риска несанкционированного доступа к информации.

Криптографические алгоритмы шифрования постоянно совершенствуются и обновляются, чтобы усилить защиту данных от новых видов атак и взломов. Без использования криптографических алгоритмов шифрования информация может быть легко украдена или изменена, что может привести к серьезным последствиям для людей, компаний и государств.

## **Асимметричные алгоритмы шифрования**

Симметричный алгоритм шифрования — это метод шифрования данных, при котором используется один и тот же ключ как для зашифровки, так и для расшифровки сообщения. То есть, если отправитель и получатель используют один и тот же ключ для шифрования и расшифровки сообщения, то такой алгоритм называется симметричным.

В симметричном шифровании данные передаются в виде шифрованного текста, который можно расшифровать только при помощи ключа, известного только отправителю и получателю сообщения. Примерами симметричных алгоритмов шифрования являются AES (Advanced Encryption Standard), DES (Data Encryption Standard) и IDEA (International Data Encryption Algorithm).

В настоящее время для симметричного шифрования используются четыре разных виды шифров.

1. Поточные шифры – это шифры, которые преобразуют один байт данных за раз. Они используют один ключ для генерации псевдослучайной последовательности, которая затем комбинируется с сообщением для получения зашифрованного сообщения. Примерами поточных шифров являются RC4, Salsa20 и ChaCha20;
2. Блочные шифры – это разновидность шифров, где данные преобразуются блоками фиксированного размера (обычно 64 или 128 бит). Каждый блок данных заменяется на зашифрованный блок данных с использованием ключа. Примерами блочных шифров являются AES, Blowfish и DES.
3. Сеть Фейстеля — один из методов построения блочных шифров. Преобразования в таком шифре состоят из раундов, в каждом из которых блок данных разбивается на две половины, которые затем преобразуются с использованием ключа. Примерами сетей Фейстеля являются алгоритмы шифрования DES и Blowfish.
4. Шифры с заменой и перестановкой – это шифры, которые используют операции замены и перестановки для шифрования данных. Они состоят из раундов, в каждом из которых происходят операции замены и перестановки данных. Примерами шифров с заменой и перестановкой являются AES и Serpent.

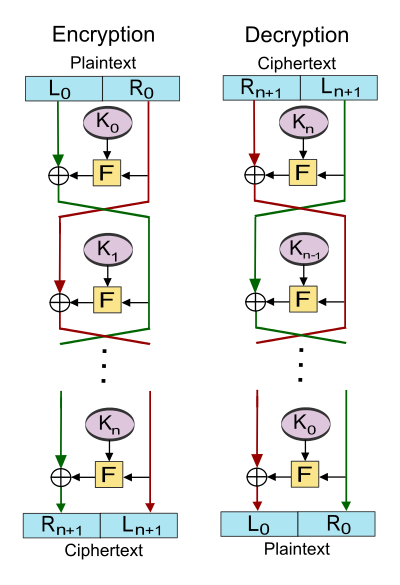


Рисунок 1 – Сеть Фейстеля.

Каждый из этих типов шифров имеет свои преимущества и недостатки, и выбор шифра зависит от конкретной задачи и уровня безопасности, который требуется достичь. Характеристиками алгоритмов симметричного шифрования являются стойкость, длина ключа, число раундов, длина шифруемого блока, сложность реализации, сложность преобразований.

## **Асимметричные алгоритмы шифрования**

Асимметричное шифрование — это метод криптографии, в котором для шифрования и дешифрования используются два ключа: открытый и закрытый. Открытый ключ шифрования может быть передан по открытому каналу связи и использован для зашифрования сообщения или проверки цифровой подписи. Закрытый ключ же используется для расшифровки сообщения или создания цифровой подписи.

Использование двух ключей позволяет решить проблему безопасного обмена ключами, которая возникает при использовании симметричного шифрования. При симметричном шифровании обе стороны должны знать один и тот же секретный ключ, который может быть использован как для шифрования, так и для расшифровки сообщений. Однако, передача секретного ключа по открытому каналу является небезопасной, так как любой злоумышленник может перехватить ключ и использовать его для расшифровки сообщений.

Системы с открытым ключом решают эту проблему, позволяя каждому участнику создать свою пару ключей. Открытый ключ может быть передан любому, кто хочет отправить сообщение, и использован для шифрования, но только владелец закрытого ключа может расшифровать сообщение. Кроме того, владелец закрытого ключа может использовать его для создания цифровой подписи, которая может быть проверена с помощью соответствующего открытого ключа.

Таким образом, системы с открытым ключом позволяют создавать безопасный канал связи между двумя участниками, используя открытый канал для передачи открытых ключей, которые могут быть свободно распространены. Это делает асимметричное шифрование очень удобным для использования в сетях, таких как интернет, где коммуникации происходят между несколькими участниками, которые не имеют заранее согласованных секретных ключей.

Схема шифрования и дешифрования выглядит следующим образом. Пусть – пространство ключей, а и – это ключи шифрования и дешифрования. - функция шифрования для произвольного ключа такая, что , где – пространство шифротекстов, а –пространство сообщений. Функция дешифрования позволяет расшифровать шифротекст и найти исходное сообщение : – набор шифрования, – набор для расшифрования. Каждая пара имеет свойство: зная , невозможно решить уравнение , то есть для данного произвольного шифротекста невозможно найти сообщение . Это означает, что по данному невозможно определить ключ дешифрования .

Ниже представлена схема шифрования информации, передаваемой лицом А лицу Б. Это могут быть как физические лица, так и организации, но для простоты восприятия принято отождествлять участников передачи информации. В таком случае А – это собеседник по имени Алиса, а Б – собеседник по имени Боб.

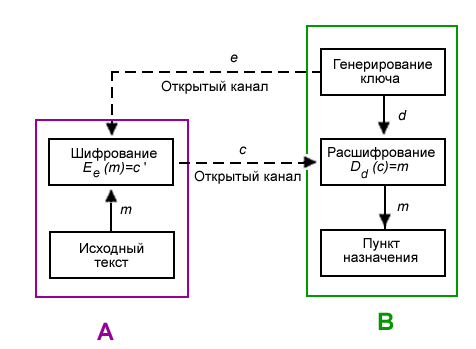


Рисунок 2 – Схема передачи информации между участниками беседы.

## **Алгоритмы шифрования на эллиптических кривых**

Эллиптическая криптография — это метод криптографии, который использует свойства эллиптических кривых для защиты данных. Она основывается на трудности вычисления дискретного логарифма в конечном поле, который используется для генерации ключей.

Ключевым элементом эллиптической криптографии является эллиптическая кривая над полем конечной характеристики, которая представляет собой график уравнения вида , где *a* и *b* – константы, определяющие форму кривой, а x и y – координаты точки на кривой. Поля конечной характеристики 2 () отличаются от бесконечных полей тем, что они содержат только конечное число элементов. Это означает, что на эллиптической кривой над полем конечной характеристики существует только конечное число точек.

Одним из основных преимуществ эллиптической криптографии является то, что она обеспечивает такой же уровень безопасности при более коротких ключах, чем традиционные методы криптографии. Это делает ее особенно полезной для мобильных устройств, где ограниченный объем памяти и скорость работы являются ограничениями.

Кроме того, эллиптическая криптография считается более безопасной, поскольку существующие алгоритмы взлома сложнее, чем в традиционных методах криптографии. Существуют методы атаки на эллиптическую криптографию, но они требуют намного больше вычислительных ресурсов и времени, чем атаки на другие криптографические системы.

Однако, несмотря на все преимущества, эллиптическая криптография требует более сложных математических вычислений, чем традиционные методы, что делает ее более медленной и требовательной к вычислительным ресурсам.

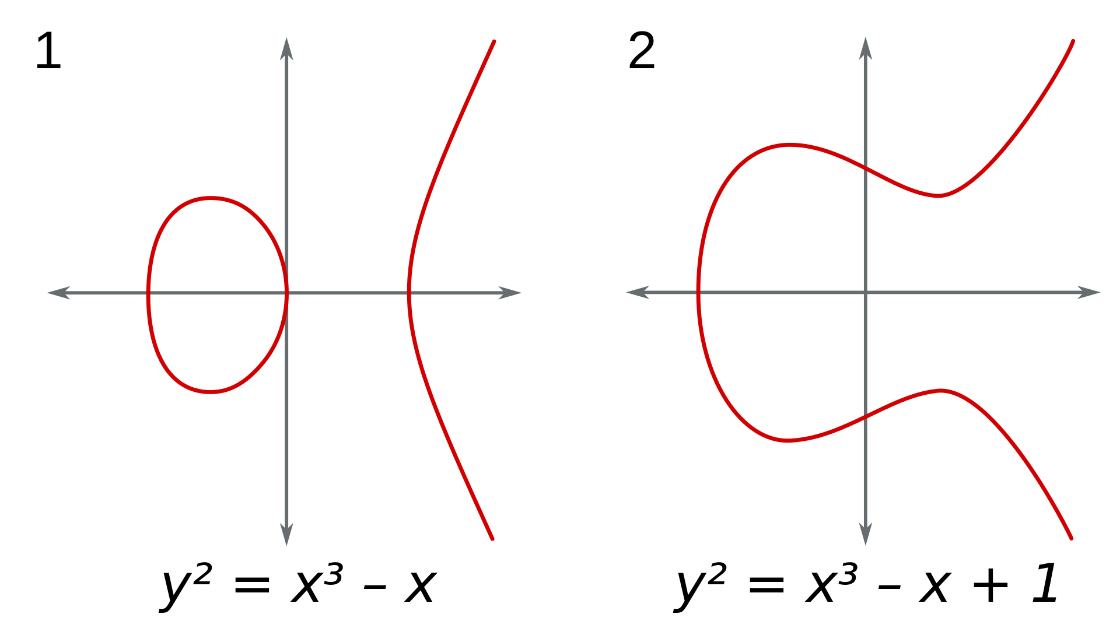


Рисунок 3 – Примеры эллиптических кривых.

## **Распределенные системы**

## **Сведения о распределенных системах**

Распределенная система — это система, которая состоит из множества компьютеров (узлов), связанных между собой сетью и работающих вместе как единое целое. В такой системе каждый узел выполняет свою задачу и обменивается информацией с другими узлами. Распределенная система может располагаться на нескольких физических узлах или же на нескольких компьютерах внутри одного физического узла.

Распределенные системы используются во многих областях, где требуется координация децентрализованных процессов и обработка больших объемов данных. Например, в области интернета вещей, облачных вычислений, сенсорных сетей, блокчейн-технологий, машинного обучения, игровой индустрии, распределенных баз данных, вычислительных кластерах и т.д. Распределенные системы также широко используются для решения задач высокопроизводительных вычислений, таких как моделирование климата, анализ геномных данных, расчет физических процессов и т.д.

Криптография играет важную роль в защите распределенных систем от различных угроз и атак. Распределенные системы обычно состоят из множества узлов, которые обмениваются данными и информацией между собой. Криптографические методы и протоколы используются для обеспечения конфиденциальности, целостности и аутентификации данных, передаваемых между узлами. Например, алгоритмы шифрования и дешифрования, цифровые подписи, протоколы аутентификации, ключевое управление — все это является необходимым для обеспечения безопасности в распределенных системах. Кроме того, криптография может использоваться для реализации различных механизмов защиты от атак, таких как атаки на отказ в обслуживании (DoS-атаки), атаки на перехват и изменение данных, атаки на подмену узлов, и многих других. Таким образом, криптография является важным компонентом в области распределенных систем и играет ключевую роль в обеспечении их безопасности и защите.

## **Структура распределенной системы**

Распределенная система — это сеть, в которой узлы не централизованы и не зависят от единого управления. Она состоит из множества узлов, которые могут обмениваться информацией и ресурсами между собой напрямую. Пример распределенной системы приведен на рисунке 4.

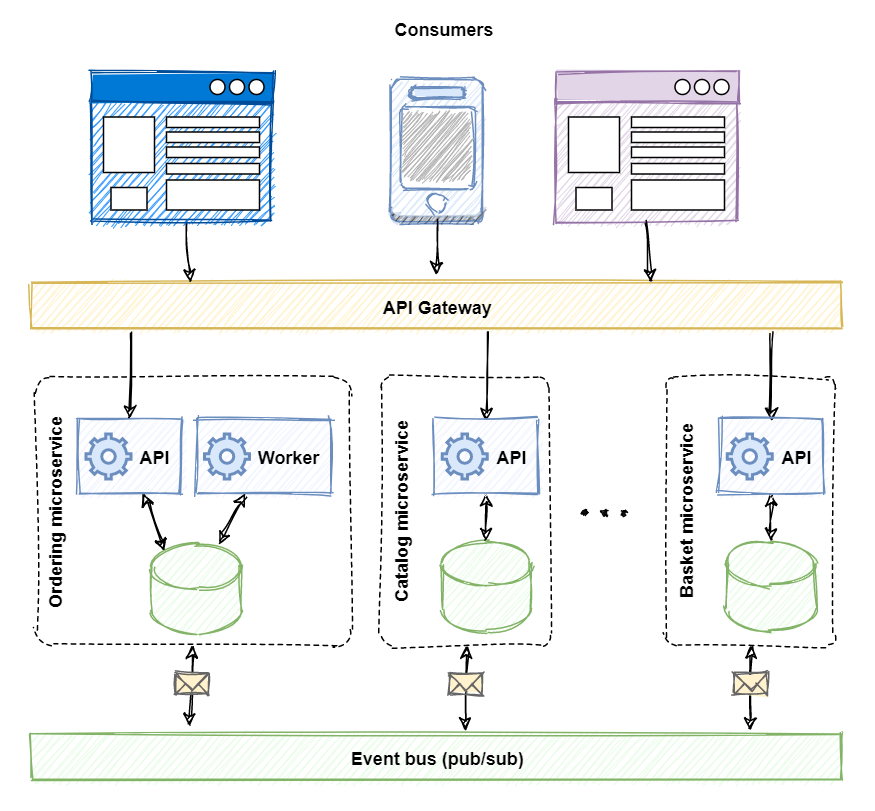


Рисунок 4 – Пример распределенной системы.

Структура распределенной сети зависит от конкретной реализации и ее целей. Однако, в общем случае, можно выделить следующие компоненты.

1. Узлы в распределенной сети могут выполнять различные функции, например, хранить данные, обрабатывать запросы, передавать сообщения и т.д. Узлы могут быть реализованы на различных устройствах, таких как компьютеры, мобильные устройства, роутеры, и т.д.
2. Обмен информацией между узлами происходит путем передачи сообщений. Сообщения могут содержать данные, запросы или команды. Каждое сообщение обычно содержит информацию о отправителе, получателе и содержании сообщения.
3. Распределенная сеть требует протоколов для установления соединений между узлами, обработки запросов и ответов, передачи данных, обеспечения безопасности и т.д.
4. Распределенная сеть может использовать распределенные ресурсы, такие как хранилище данных, процессорное время, пропускная способность каналов связи и т.д.
5. Топология сети описывает структуру связей между узлами в сети. Распределенная сеть может использовать различные топологии, такие как полная сеть, дерево, кольцо, шина, P2P сеть и т.д.
6. Распределенная сеть может использовать различные алгоритмы для решения задач, таких как маршрутизация, репликация данных, консенсус и т.д.

В целом, структура распределенной сети может быть очень сложной и зависеть от конкретного применения. Однако, понимание основных компонентов и принципов ее работы позволяет разработчикам создавать более эффективные и надежные распределенные системы.

## **Виды распределенных систем**

Распределенные системы могут быть различных видов в зависимости от архитектуры и способа организации взаимодействия между узлами. Ниже приведены некоторые примеры распределенных систем и компаний, которые их используют.

1. Клиент-серверная архитектура – это архитектура, в которой существует сервер, который обслуживает несколько клиентов. Клиент-серверная архитектура используется, к примеру, в компании Google, Facebook, Amazon в онлайн-сервисах.
2. Peer-to-Peer архитектура – архитектура, все узлы которой имеют равные возможности и работают независимо друг от друга, без централизованного управления. Skype — это программа для видео- и голосовой связи, использующая P2P технологии.
3. Кластеризованная архитектура – вид архитектуры, в которой несколько узлов объединяются в кластер, который работает как единый узел. Компании, использующие кластеризованную архитектуру, - Apache и Google.
4. Grid-системы — это сеть из географически распределенных компьютеров, которые работают вместе для решения одной задачи.
5. Cloud-системы — это сеть из удаленных серверов, которые предоставляют вычислительные ресурсы, хранилище данных и другие услуги через интернет. Используется в таких продукатх, как Amazon Web Services, Microsoft Azure и Google Cloud. Все эти компании предоставляют облачные сервисы для хранения данных, разработки приложений и т.д.

## **Исследование существующих распределенных криптосистем**

Для того, чтобы разработать надежную криптосистему, необходимо изучить уже существующие технологии, чтобы знать сильные, а главное слабые стороны в используемых криптографических алгоритмах. Это позволяет найти уязвимости и мотивирует найти решения, способствующие повышению уровня безопасности системы. Одной из самых известных и защищенных систем является приложение Telegram.

Telegram — это мессенджер, который использует криптографические протоколы для обеспечения конфиденциальности сообщений, передаваемых через платформу. Telegram использует протокол MTProto для защиты конфиденциальности сообщений. Этот протокол использует асимметричное шифрование для защиты сообщений при передаче по сети, и дополнительно шифрует сообщения на устройстве отправителя, что делает невозможным чтение сообщений даже при их перехвате.

Telegram использует 2048-битные RSA-ключи для аутентификации пользователей и защиты их данных. Ключи генерируются на стороне клиента, и только клиент имеет доступ к своему ключу. Telegram имеет функцию двухфакторной аутентификации, которая обеспечивает защиту от взлома учетной записи. При этом после входа в приложение через одно устройство, необходимо ввести дополнительный пароль для входа на другом устройстве. Также Telegram защищает голосовые вызовы с помощью шифрования на уровне конечной точки, что означает, что разговоры шифруются на устройстве отправителя и не могут быть прослушаны третьими лицами.

В целом, Telegram использует современные криптографические протоколы и технологии, чтобы обеспечить безопасность и конфиденциальность пользовательских данных, однако у него есть и некоторые недостатки. Алгоритм MTProto не является открытым стандартом, и пользователи не могут проверить его на уязвимости и ошибки. Это может привести к тому, что уязвимости в алгоритме могут оставаться необнаруженными и использоваться злоумышленниками для атак. MTProto использует свою собственную систему ключей, которая не соответствует стандартам шифрования, что может повлечь за собой проблемы с безопасностью и приватностью данных. Как проприетарный алгоритм, MTProto не может быть проверен независимыми экспертами на наличие уязвимостей или ошибок. Это может привести к тому, что проблемы с безопасностью могут оставаться необнаруженными.

# **РАЗРАБОТКА РЕШЕНИЯ**

## **Описание решения поставленной задачи**

## **Математическая библиотека**

В первую очередь для работы с эллиптическими кривыми и алгоритмами, которые их используют, необходимо разработать библиотеку исходного кода для работы с эллиптическими кривыми над полями конечной характеристики. Библиотека состоит из набора основных сущностей, необходимых для дальнейшей работы, а также реализует набор операций над эллиптическими кривыми.

1. Сложение точек — это операция, при которой две точки на кривой складываются, чтобы получить третью точку, которая также лежит на кривой. Сумма точек используется для шифрования и подписывания сообщений в криптографии на эллиптических кривых.
2. Умножение точки на скаляр — это операция, при которой точка на кривой умножается на целое число, чтобы получить новую точку на кривой. Эта операция используется для генерации ключей в криптографии на эллиптических кривых.
3. Вычисление порядка точки — это операция, при которой вычисляется количество точек на кривой, которые можно получить, складывая выбранную точку на кривой себя саму многократно. Знание порядка точки важно для безопасности в криптографии на эллиптических кривых.
4. Вычисление дискретного логарифма — это операция, при которой вычисляется целое число, которое является степенью известной точки на кривой, чтобы получить другую заданную точку на кривой. Эта операция служит основой для создания криптографических протоколов на эллиптических кривых, таких как протокол Диффи-Хеллмана на эллиптических кривых.

## **Криптосистема**

Для обеспечения защиты системы необходимо разработать систему криптографической защиты. Криптографическая система будет использовать как симметричные алгоритмы шифрования, так и асимметричные.

В первую очередь всеми участниками распределенной системы согласуется эллиптическая кривая, которая будет использоваться для шифрования данных. Для решения поставленной задачи была выбрана эллиптическая кривая под названием SECP256R1, также известная как P-256.

Эта эллиптическая кривая определяется уравнением , где *a* и *b* – константы, определенные стандартом ANSI X9.62, а *x* и *y* - координаты точки на кривой. Одним из основных достоинств эллиптической кривой SECP256R1 является ее безопасность. Благодаря размеру поля и параметрам кривой, она обеспечивает высокую стойкость к криптоанализу и предотвращает возможность атак с использованием известных алгоритмов. Также эллиптическая кривая SECP256R1 хорошо подходит для использования в различных приложениях благодаря своей эффективности. Она позволяет производить операции шифрования и подписи данных с высокой скоростью и использовать меньшее количество ресурсов, чем при использовании других криптографических алгоритмов.

Для шифрования данных, прежде всего, необходимо создать ключи шифрования. Эллиптическая кривая определяет следующие параметры:

1. *p* – простое число, задающее размер конечного поля;
2. *a* и *b* – коэффициенты уравнения эллиптической кривой;
3. *G* – базовая точка, генерирующая подгруппу;
4. *n* – порядок подгруппы;
5. *h* – кофактор подгруппы.

Алгоритмы эллиптических кривых будут работать в циклической подгруппе эллиптической кривой над конечным полем. В результате параметрами области определения для алгоритмов является шестёрка (p, a, b, G, n, h). В результате будет сгенерировано два ключа – приватный и публичный. Приватный ключ – это случайное большое число, а публичный ключ – точка (X, Y) на эллиптической кривой.

Для обмена ключами используется алгоритм ECDH — это криптографический протокол, используемый для обмена ключами шифрования между участниками общения в сети с целью обеспечения безопасности передачи данных. Он обеспечивает безопасный обмен ключами, так как любой несанкционированный пользователь, который пытается перехватить данные, не сможет получить секретный ключ без знания закрытого ключа. Данный алгоритм основан на криптографических алгоритмах эллиптических кривых и является аналогом алгоритма Диффи-Хеллмана для классической криптографии.

Для шифрования данных используется алгоритм шифрования AES — это симметричный алгоритм блочного шифрования, который шифрует данные блоками размером 128 бит, используя ключи длиной 128, 192 или 256 бит. AES работает в несколько раундов, где каждый раунд состоит из различных этапов, таких как замена байтов, перемешивание столбцов и смещение строк. В результате каждого раунда получается новый блок данных, который используется для следующего раунда. Число раундов зависит от размера ключа - для ключей длиной 128 бит используется 10 раундов, для ключей длиной 192 бит - 12 раундов, а для ключей длиной 256 бит - 14 раундов. В начале процесса шифрования, исходный текст разбивается на блоки по 128 бит, которые затем проходят через начальный раунд шифрования, в котором каждый байт заменяется на другой байт из предопределенной таблицы замен. Затем происходит перемешивание столбцов и смещение строк, чтобы получить новый блок данных, который подвергается следующему раунду шифрования. После нескольких раундов шифрования, получается зашифрованный блок данных, который можно отправлять по сети или сохранять на диске. Для дешифрования данных, процесс применяется в обратном порядке, начиная с последнего раунда и заканчивая начальным раундом. В процессе дешифрования, каждый блок данных проходит через обратные этапы, чтобы получить исходный текст.

Для создания и проверки электронных подписей используется алгоритм ECDSA — это метод создания электронной подписи, который использует эллиптические кривые. Для создания электронной подписи необходимы закрытый и открытый ключи шифрования. Закрытый ключ хранится только у владельца и используется для создания электронной подписи. Открытый ключ может быть распространен публично и используется для проверки электронной подписи. Во время создания электронной подписи сообщения вычисляется хеш-функцию, которая сжимает сообщение до фиксированной длины и в конечном счете возвращает фиксированный результат. Затем используется закрытый ключ шифрования для создания электронной подписи, которая включает информацию о сообщении и публичном ключе. Любой пользователь может убедиться в подлинности подписи, используя открытый ключ и сравнивая результат с хеш-значением исходного сообщения.

## **Выбор программных средств разработки**

## **C#**

C# — это объектно-ориентированный язык программирования, который был разработан Microsoft в 2000 году. Он предназначен для разработки веб-серверов, мобильных приложений и приложений для ПК по управлением системы Windows. Синтаксис и структура языка C# похожи на языки программирования C и C++, но C# является более современным языком с большим количеством возможностей. C# полностью поддерживает объектно-ориентированное программирование. Это означает, что программа состоит из объектов, которые взаимодействуют друг с другом. C# — это строго типизированный язык. Это означает, что переменные должны быть определены с указанием их типа, и тип не может быть изменен во время выполнения программы. C# имеет автоматическую сборку мусора, что означает, что программа автоматически удаляет неиспользуемые объекты и освобождает память. C# поддерживает многопоточность и асинхронное программирование, что позволяет программистам создавать приложения, которые выполняют несколько задач одновременно. C# имеет широкую стандартную библиотеку, которая включает в себя множество функций для работы с файлами, сетью, базами данных, графикой и другими задачами. C# компилируется в промежуточный язык (IL), который может быть выполнен на любой платформе, поддерживающей среду выполнения .NET. Также C# интегрируется с другими языками программирования, которые поддерживают .NET.

Использование C# может быть особенно полезно для разработки приложений для Windows, веб-приложений на платформе .NET и мобильных приложений на платформе Xamarin. C# также используется для создания игр, научных и инженерных приложений, систем управления базами данных и других приложений. Выбор C# обусловлен актуальностью данного языка программирования. Язык программирования C# имеет яркие перспективы развития и применения в различных областях, потому что платформа .NET, для работы с которой используется данный язык программирования, я является одной из самых распространенных платформ для разработки приложений. Развитие .NET-платформы и дальнейшее развитие C# могут привести к созданию новых возможностей для разработчиков и более широкому применению этого языка.

## **ASP.NET**

ASP.NET — это веб-фреймворк, разработанный компанией Microsoft для создания динамических веб-приложений и веб-сайтов. ASP.NET используется для создания серверных приложений на языках программирования C# или Visual Basic .NET, которые работают в рамках платформы .NET Framework. ASP.NET был впервые выпущен в 2002 году компанией Microsoft как часть .NET Framework. Он был разработан как замена для старой технологии ASP (Active Server Pages), которая была выпущена в 1996 году. Одной из целей создания ASP.NET было улучшение производительности и расширяемости веб-приложений.

ASP.NET обеспечивает мощный набор инструментов и библиотек для разработки и развертывания веб-приложений.

1. ASP.NET Web Forms — это модель программирования, которая позволяет создавать веб-страницы, которые могут содержать элементы управления, такие как кнопки, текстовые поля и списки. Web Forms используются для создания сложных пользовательских интерфейсов.
2. ASP.NET MVC (Model-View-Controller) — это модель программирования, которая используется для создания веб-приложений, основанных на шаблоне проектирования Model-View-Controller. MVC разделяет приложение на три компонента: модель, отображение и контроллер. Это делает приложения более легко сопровождаемыми и тестируемыми.
3. ASP.NET Web API — это фреймворк, который позволяет создавать RESTful веб-службы, которые могут использоваться другими приложениями для доступа к данным и функциональности.
4. ASP.NET SignalR — это библиотека, которая позволяет создавать реальном времени веб-приложения, которые могут общаться с браузерами и устройствами клиентов в режиме реального времени.
5. ASP.NET также обеспечивает поддержку для многопоточности, безопасности, управления сеансами и многое другое. Фреймворк интегрируется с другими продуктами Microsoft, такими как Microsoft SQL Server, Visual Studio и IIS (Internet Information Services).

В целом, ASP.NET представляет собой мощный инструмент для разработки и развертывания веб-приложений и веб-сайтов.

## **WPF**

Windows Presentation Foundation (WPF) — это технология разработки графических интерфейсов для приложений Windows, которая была представлена компанией Microsoft в 2006 году. WPF позволяет создавать более интерактивные и эффективные пользовательские интерфейсы с помощью визуальных элементов, таких как кнопки, списки, таблицы, графики и другие элементы управления. WPF разработан на основе технологии XAML (eXtensible Application Markup Language), которая позволяет описывать интерфейс приложения в виде XML-документа. Это делает возможным разделение дизайна и логики приложения, что упрощает его сопровождение и расширение.

Основной особенностью Windows Presentation Foundation (WPF) являются возможность создания приложения с гибким и масштабируемым пользовательским интерфейсом, который может изменяться в зависимости от размеров экрана или других параметров окружающей среды. Помимо этого, WPF интегрируется с другими технологиями .NET Framework, такими как ADO.NET, ASP.NET, WCF и другими, что упрощает создание приложений, которые используют эти технологии. WPF использует аппаратное ускорение для отображения графических элементов, что позволяет создавать более производительные приложения с лучшей графикой, а также имеет встроенную поддержку мультимедиа, что позволяет создавать более интерактивные приложения с использованием звука, видео и других мультимедийных элементов. WPF позволяет создавать стили и шаблоны, которые можно применять к различным элементам интерфейса, что упрощает создание единообразного и современного пользовательского интерфейса.

WPF применяется в различных областях, включая банковское дело, медицину, науку, игровую индустрию и другие. Он используется для создания различных типов приложений, от маленьких утилит до крупных корпоративных систем. WPF также позволяет создавать приложения, которые могут работать на разных устройствах, таких как настольные компьютеры, планшеты и смартфоны.

## **PostgreSQL**

PostgreSQL - это свободная реляционная база данных с открытым исходным кодом, которая была создана в 1986 году в Университете Беркли. PostgreSQL является одной из самых популярных реляционных баз данных в мире и используется многими крупными компаниями, включая Apple, Cisco, Fujitsu, Microsoft и другие.

PostgreSQL поддерживает стандарт SQL, что делает его легко использовать для разработки приложений. SQL — это язык программирования, который используется для работы с реляционными базами данных. SQL позволяет программистам и администраторам баз данных выполнять различные операции с данными, такие как создание, изменение, удаление и извлечение информации из баз данных. SQL является декларативным языком, что означает, что пользователь указывает желаемый результат, а не способ, которым его необходимо достичь. SQL состоит из различных команд, таких как SELECT, INSERT, UPDATE, DELETE, CREATE, DROP и другие, которые позволяют управлять данными в базе данных. Он также поддерживает различные функции агрегирования, операторы соединения и группировки, что делает его мощным инструментом для аналитики данных и получения информации из баз данных.

PostgreSQL является очень расширяемой базой данных и позволяет разработчикам создавать свои собственные типы данных, функции и процедуры. PostgreSQL предоставляет многоуровневую систему безопасности, которая позволяет защитить данные от несанкционированного доступа, а также обеспечивает высокую надежность и стабильность, что делает его идеальным выбором для критически важных приложений.

PostgreSQL может использоваться для различных типов приложений, включая веб-приложения, системы управления контентом, электронную коммерцию, аналитику данных, мобильные приложения и другие. Он также широко используется в различных отраслях, включая финансы, здравоохранение, образование, науку и другие.

## **Docker и Docker Compose**

Docker — это платформа для создания, развертывания и управления приложениями в контейнерах. Он использует технологию виртуализации на уровне операционной системы, чтобы разделить приложения и их зависимости от основной операционной системы, что делает их переносимыми и легко развертываемыми на любой системе. Docker позволяет программистам и администраторам упаковывать приложения и их зависимости в легковесные, портативные контейнеры, которые можно запускать на любой поддерживаемой платформе, без необходимости устанавливать и настраивать дополнительное программное обеспечение.

Docker Compose — это инструмент для описания и запуска многоконтейнерных приложений в Docker. Он позволяет описывать зависимости и конфигурации контейнеров в файле YAML, что делает управление множеством контейнеров проще и более эффективным. Docker Compose автоматически управляет зависимостями между контейнерами, запускает их в правильном порядке и предоставляет легкий способ для управления многоконтейнерными приложениями.

Использование Docker и Docker Compose позволяет создавать, развертывать и масштабировать приложения с высокой производительностью и меньшими затратами на инфраструктуру. Docker также предоставляет возможности для автоматизации их развертывания, мониторинга и обновления, что делает его одним из наиболее популярных инструментов для контейнеризации приложений в индустрии.

## **Описание программной разработки**

## **Распределенная система**

Распределенная система состоит из нескольких компонентов, приведенных на рисунке ниже.

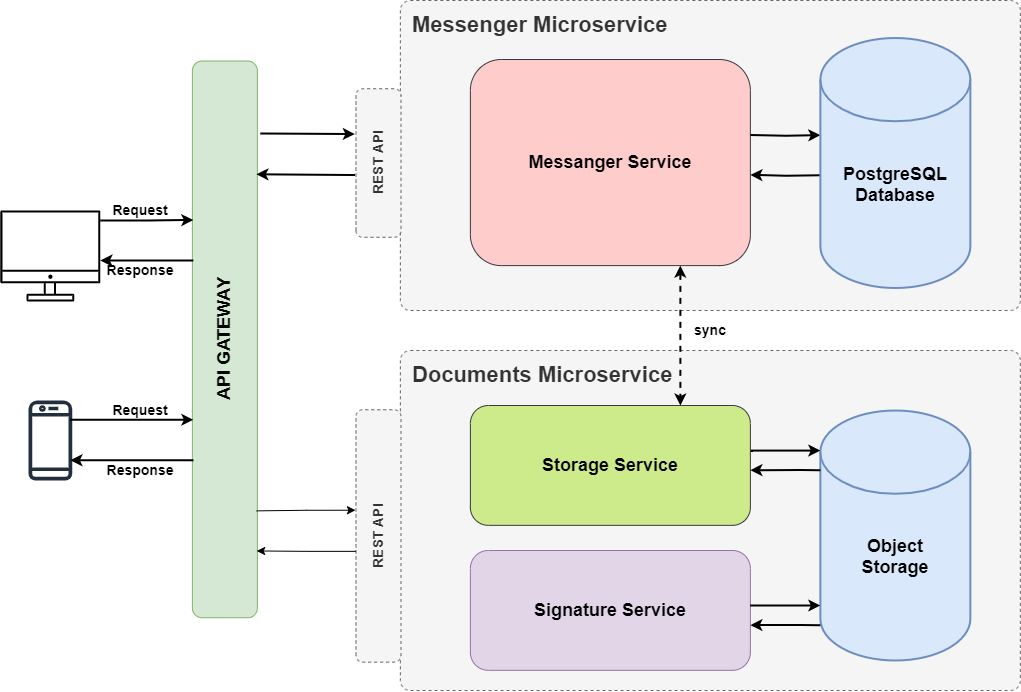


Рисунок 5 – Архитектура приложения.

Для реализации распределенной системы была выбрана клиент-серверная архитектура. Данная разновидность распределенной архитектуры состоит из сервера, который обслуживает несколько клиентов. Приложение представляет из себя мессенджер, использующий разработанную библиотеку для работы с эллиптическими кривыми и алгоритмами для обеспечения надежной защиты данных пользователей. Приложение имеет микросервисную архитектуру. Микросервисная архитектура — это способ разработки программного обеспечения, в котором приложение разбивается на небольшие самостоятельные блоки, называемые микросервисами. Каждый микросервис выполняет свою специфическую функцию и может взаимодействовать с другими микросервисами, используя API или другие методы коммуникации. Этот подход помогает сделать приложение более масштабируемым, гибким и легко изменяемым, что может упростить его разработку и обслуживание. Кроме того, микросервисы могут быть развернуты и масштабированы независимо друг от друга, что позволяет более эффективно использовать вычислительные ресурсы и повышать отказоустойчивость.

В соответствии с микросервисной архитектурой было спроектировано два самостоятельных сервиса: сервис мессенджера, содержащий основную бизнес-логику, и сервис документов, отвечающий за хранение документов и работу с электронными подписями. Для объединения микросервисов в единый компонент используется паттерн API Gateway — это паттерн проектирования, используемый для создания единой точки входа в микросервисную архитектуру, который облегчает доступ и управление множеством микросервисов. Суть этого паттерна заключается в том, что все запросы к микросервисам проходят через единую точку входа.

## **REST API сервис**

REST — это стиль проектирования, используемый для проектирования веб-сервисов и API. Он основан на принципе организации взаимодействия между клиентом и сервером через стандартные HTTP-методы, такие как GET, POST, PUT, DELETE и т.д., и ресурсы, которые идентифицируются уникальными URI (Uniform Resource Identifier). Основной принцип REST — это разделение клиента и сервера, что позволяет улучшить масштабируемость и гибкость приложений. Клиент и сервер обмениваются данными в формате, общеизвестном как JSON или XML, и общий формат сообщений позволяет легко интегрировать приложения, разработанные на разных платформах. REST также поддерживает кэширование, что может улучшить производительность, и не требует состояния сессии на стороне сервера, что упрощает масштабирование и увеличивает надежность приложения. REST является одним из наиболее распространенных и удобных способов взаимодействия между клиентом и сервером в современном веб-программировании. Он широко используется в различных приложениях, включая социальные сети, мобильные приложения и облачные сервисы.

Данный сервис представляет из себя REST API сервис, содержащий основную бизнес-логику приложения и реализующий CRUD операции с базой данных. Приложение имеет трехуровневую архитектуру, состоящую из следующих уровней:

1. уровень бизнес-логики – это слой, который содержит в себе основную логику приложения и является центральным, независимым слоем, вокруг которого выстраивается приложение;
2. уровень доступа к данным – это слой, который определяет, где и в каком виде хранятся данные, с которыми работает приложение;
3. уровень API – это слой, который реализует REST API для взаимодействия микросервиса с внешним миром.

Выбор трехуровневой монолитной архитектуры решает множество архитектурных проблем небольших приложений. Разделение функциональности на три уровня облегчает масштабирование приложения и улучшает его гибкость. Каждый уровень может быть развернут и масштабирован независимо, что упрощает управление ресурсами. Разделение бизнес-логики и уровня данных помогает обеспечить более высокий уровень безопасности. Бизнес-логика не имеет прямого доступа к данным, что позволяет лучше контролировать доступ к ним и уменьшить риски несанкционированного доступа. Как итог, трехуровневая архитектура повышает читаемость и улучшает поддержку кода. Разделение кода на три уровня позволяет разработчикам работать с более четко определенными участками кода, что упрощает его понимание и поддержку.

## **Сервис баз данных**

Хранением данных пользователей в системе занимается сервер баз данных. Сервер обладает единственной ответственностью - хранением данных, поэтому на сервере располагается единственный объект - реляционная база данных. Вынесение базы данных приложения и связанных с ней компонентов на отдельный сервер является преимуществом, поскольку повышает отказоустойчивость системы. Разбиение основного приложения и базы данных вводит такое понятие, как разделение ответственностей. Сервер основного приложения отвечает за обработку запросов и логику приложения, а сервер баз данных отвечает за хранения информации в системе. Это позволяет повысить отказоустойчивость системы, и в случае сбоя того или иного компонента позволяет сохранить работоспособность системы.

Для хранения информации была выбрана реляционная база данных – это тип баз данных, который организует хранение информации в виде таблиц. Каждая таблица представляет собой коллекцию объектов или сущностей, например, пользователей сервиса. В таблице есть столбцы, которые определяют тип данных, хранимых в таблице, и строки, которые содержат фактические данные. Для работы с реляционными базами данных используется язык SQL. Существуют также нереляционные базы данных, именуемые как NoSQL базы данных. Отличие нереляционных баз данных от реляционных заключается в способе хранения информации. Если в реляционных базах данных информация хранится в виде таблиц, то в нереляционных базах данных информация может быть представлена в виде документов, кэша и т.д.

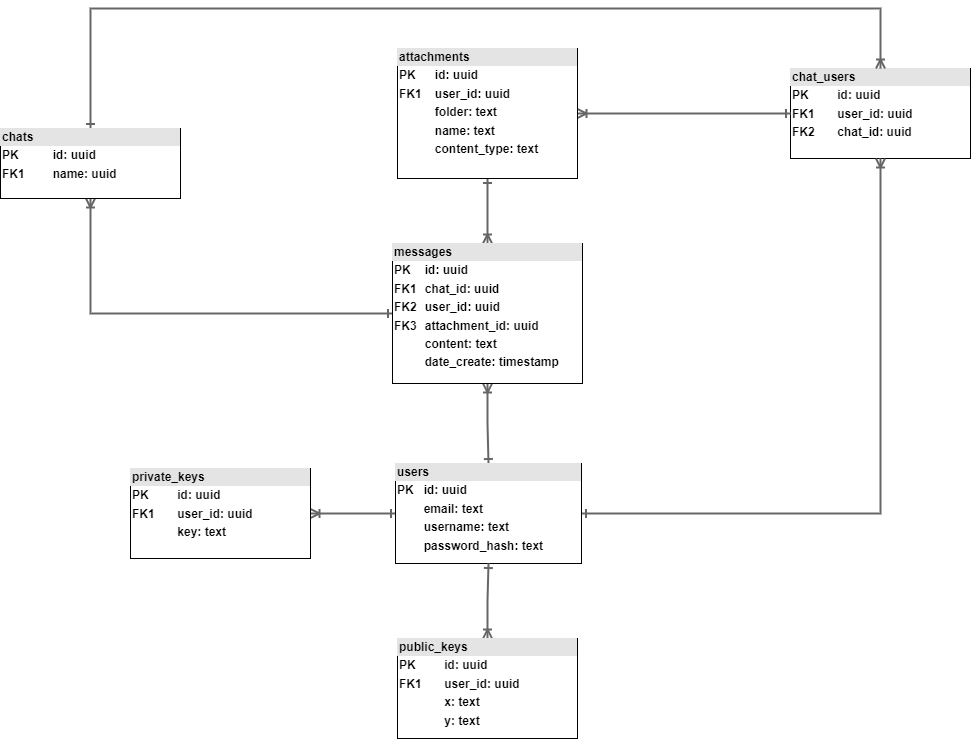


Рисунок 6 – Схема данных приложения.

Выбор такого типы баз данных обоснован простотой использования реляционных баз данных. Потому что они имеют простую и интуитивно понятную структуру, которая позволяет легко добавлять, изменять и удалять данные. Это делает их идеальными для широкого круга пользователей, включая разработчиков и конечных пользователей.

## **Сервис файлового хранилища**

Сервис предоставляет пользователю возможность работать с документами. Документами в данном случае называются любые файлы, например, текстовые файлы, архивы, бинарные файлы и т.д. Для хранения этих документов и последующей их обработки необходимо разработать дополнительный микросервис, который помимо хранения будет выполнять и другие функции.

Главной задачей данного сервиса является хранение документов. Существует множество готовых решения от различных компаний, предоставляющий необходимый функционал, однако практически все эти решения платные, либо избыточные для данной системы. В результате было принято решение разработать собственное файловое хранилище. Данный сервис представляет из себя файловую систему и инструменты для работы с электронной подписью.

Помимо управления файлами, сервис занимается электронными подписями документов. Электронная подпись — это цифровой аналог обычной подписи, которые обеспечивает проверку подлинности документов и сообщений. Она представляет собой уникальную комбинацию символов, которые обеспечивают конфиденциальность, целостность и аутентификацию информации. Для создания и верификации электронной подписи используются алгоритмы из разработанной алгоритмической библиотеки. При верификации электронной подписи любой пользователь системы может убедиться, что документ или сообщение не были изменены после того, как были подписаны, и что они были созданы именно автором, чья подпись была использован.

Разработанный сервис предоставляет API для взаимодействия с внешним миром. В качестве инструмента межпроцессного взаимодействия был выбран REST API. Сервис реализует методы загрузки и скачивания документов, а также отвечает за создание, хранение и проверку электронной подписи документа.

# **РЕЗУЛЬТАТЫ РАБОТЫ**

## **Характеристика условий и места применения разработки**

Разработанное приложение является универсальным средством общения в сети Интернет, поэтому может иметь множество мест применения. Главной особенностью приложения является использование алгоритма на эллиптических кривых, что обеспечивает конфиденциальность пользователей и гарантирует надежную защиту информации, которой обмениваются пользователи, от третьих лиц.

Распределенная система может быть использована в сфере финансовых услуг. Безопасность данных играет важную роль в финансовой индустрии. Доступ к информации, содержащей финансовую тайну, может причинить вред не только одному физическому лицу, но и целым компаниям, потому что эта информация может быть использована злоумышленниками в корыстных целях. Также не стоит забывать и о конфиденциальных данных пользователей финансовых услуг, которые также могут быть украдены злоумышленниками. Для решения вышеупомянутых проблем может использоваться разработанный мессенджер с криптографической защитой данных, который обеспечит защищенный канал обмена финансовыми данными между банками, инвестиционными фирмами, трейдерами и клиентами. Он поможет обеспечить конфиденциальность при передаче чувствительных финансовых сведений, таких как банковские реквизиты, номера кредитных карт и инвестиционные стратегии.

Разработка может быть использована в сфере информационной безопасности. Компании, занимающиеся информационной безопасностью, могут использовать мессенджер с криптографической защитой для обмена данными с клиентами и другими специалистами в области безопасности. Зачастую такие данные могут содержать информацию о незащищенных местах систем, исследуемых специалистами в области кибербезопасности. Попа в руки злоумышленников, эта информация может быть использована для взлома слабых мест различных систем. Использование разработанного мессенджера позволяет защитить ценные информационные активы и предотвратить несанкционированный доступ к ним.

Также приложение может быть использовано в образовательных учреждениях, как в высших учебных заведениях, так и в школах. Мессенджер с криптографической защитой может быть использован для безопасного обмена информацией между студентами и преподавателями, а также для обмена информацией, содержащей научную тайну. Также приложение может обеспечить конфиденциальность при передаче учебных материалов, результатов тестирования и других личных данных.

Наконец, разработка может быть применена в государственных учреждениях. Государственные организации могут использовать мессенджер с криптографической защитой для безопасной коммуникации между государственными служащими, правительственными агентствами и другими структурами. Такая информация содержит наибольшую ценность, поскольку может содержать в себе государственную тайну, поэтому подобного характера нуждаются в особой защите от сторонних лиц. Утечка государственной информации ставит под угрозу государственную безопасность. Для предотвращения утечки данных может быть использовано разработанное приложение, которое обеспечит надежную защиту данных

## **Результаты работы разработанного программного кода**

## **Серверное приложение**

## **Сервис основного приложения**

В качестве основного серверного приложения был разработан REST API сервис на языке программирования C#. Для простоты тестирования и работы с API приложения был использован Swagger – это набор инструментов для разработки API. Преимуществом работы со Swagger является автоматическая генерация документации API приложения, что позволяет видеть все методы API, доступные в приложении. Документация API основного приложения мессенджера приведена на рисунке ниже.

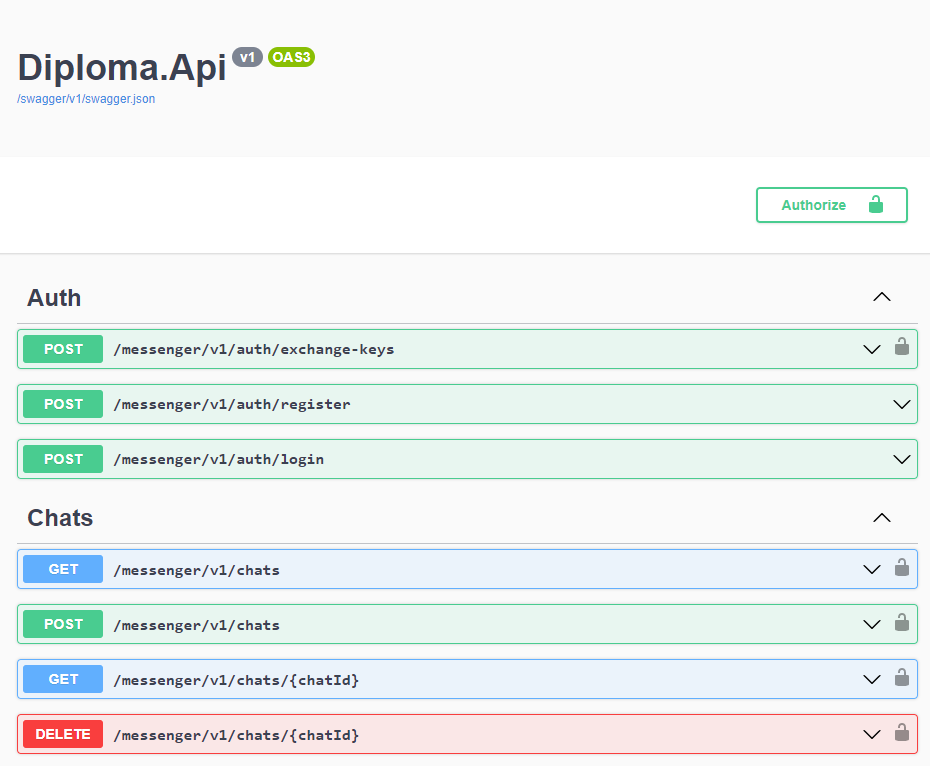


Рисунок 7 – Документация API, сгенерированная с помощью Swagger.

Приложение содержит механизм авторизации и бизнес-логику приложения: общение и работа с документами. Для авторизации была использована аутентификация пользователей на основе JWT токена – это протокол аутентификации, в котором пользователи проверяют свои идентификационные данные в обмен на уникальный токен. Токен добавляется в Headers каждого запроса, после чего на стороне сервера происходит валидация токена. Если токен оказался валидным, пользователю становятся доступны ресурсы приложения, в противном случае авторизация является неудачной. В этом случае пользователю на каждый запрос в систему возвращается ответ «401 Unathorized».

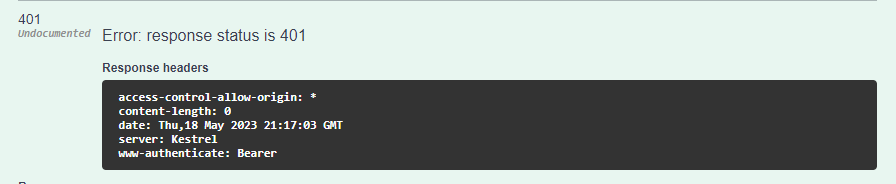


Рисунок 8 – Пример неудачной авторизации в системе.

После успешной авторизации пользователю становятся доступны функции система. Методы API представляют из себя команды и запросы к системе. Запрос – это операция получения данных, в то время как команда – операция изменения данных (создание, редактирование, удаление). В API реализованы запросы и использованием метода HTTP GET и команды с использованием методов HTTP POST, PUT, DELETE. Главная часть приложения – бизнес-логика. Бизнес-логика разделена на возможности, которые система предоставляет пользователя, а именно: возможность общения в чате с другими пользователями, работа с документами и управление профилем. На рисунках ниже приведены примеры методов API для управления чатами и сообщениями.

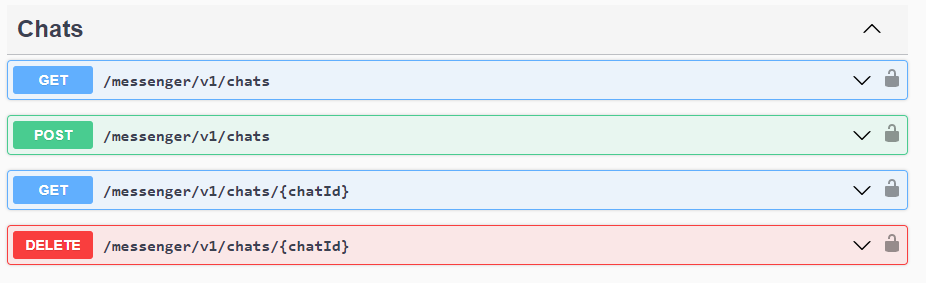


Рисунок 9 – Методы API для управления чатами.

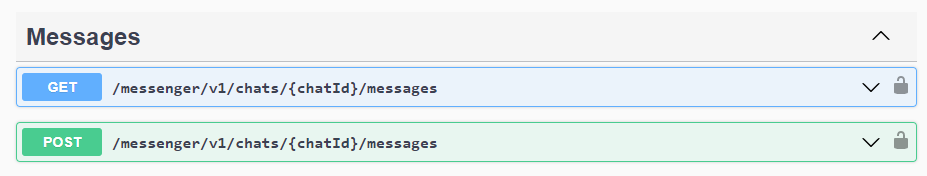


Рисунок 10 – Методы API для работы с сообщениями.

## **Сервис файлового хранилища**

В процессе общения пользователи могут обмениваться не только текстовыми сообщениями, но и файлами. В результате возникает вопрос хранения файлов в системе. Согласно принципу единой ответственности SOLID каждый сервис имеет лишь одну зону ответственности. Сервис мессенджера отвечает за реализацию бизнес-логики, поэтому он не должен отвечать за хранение файлов. В итоге было принято решение разработать отдельный сервис для хранения документов.

Сервис документов представляет из себя REST API приложение на языке программирования C#. Как и основной сервис, сервис документов также использует авторизацию через JWT токен, чтобы обезопасить систему от неавторизованных пользователей. В случае успешной авторизации пользователь сможет воспользоваться методами API, которые предоставляет сервис В противном случаев качестве ответа вернется ошибка авторизации.

Зоной ответственности данного сервиса является хранение файлов. Для этого были разработаны методы для загрузки и скачивания документов, приведенные на рисунке ниже.

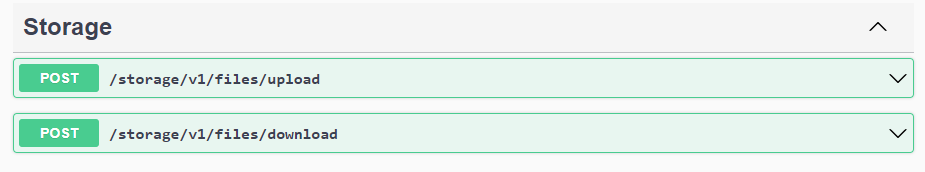


Рисунок 11 – Методы API для работы с файловым хранилищем.

Хранилище представляет из себя файловую систему с загруженными в него документами. Каждый документ хранится в папке, название которой формируется путем нахождения хэш-суммы содержимого файла и преобразования полученного значения в строку. При загрузке нового документа сначала вычисляется хэш-сумма содержимого файла. Далее полученное значение, преобразованное в строку, сравнивается с названиями уже существующих папок. В случае, если в результате поиска была найдена папка, информация о файле в этой папке возвращается пользователю в ответе. Таким образом, разработанное решение позволяет оптимизировать количество расходуемой памяти путем исключения возможности создания дубликатов. Пример структуры хранения файлов приведен на рисунке ниже.

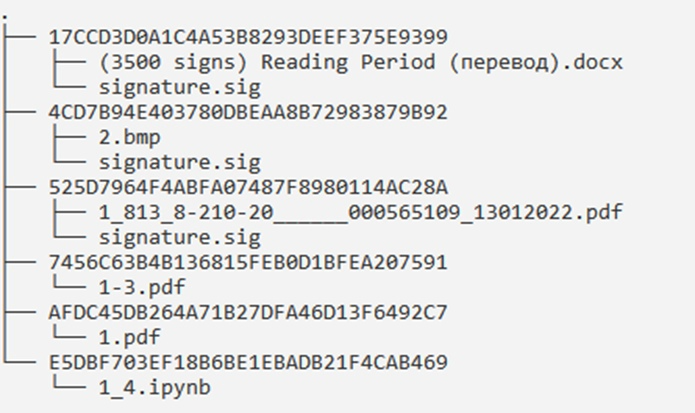


Рисунок 12. Структура хранения документов

Данные системы всегда должны находится в актуальном состоянии. С выделением нового микросервиса возникает проблема синхронизации данных с основным приложением, так как после загрузки файла в хранилище информация об этом файле должна быть известна основному сервису приложения. Для этого был реализован механизм синхронизации информации о новых документах, загруженных в хранилище. При загрузке нового документа сервис документов использует API сервиса мессенджера для выполнения команды создания нового файла. В теле команды содержится информация о местонахождении файла (папка в хранилище, название). В результате выполнения команды в базу данных записывается информация о загруженном файле. Данное решение позволяет содержать данные о файлах в актуальном состоянии.

## **Сервис электронной подписи**

Помимо работы с файлами распределенная система позволяет пользователю работать с электронной подписью документов. Сервис электронной подписи аналогично сервисам мессенджера и документов является REST API приложением с набором методов, доступных пользователю. Сервис поддерживает авторизацию через токен, что обеспечивает защищенность системы от несанкционированного доступа. Данный сервис связан с сервисом документов и предоставляет пользователю подписывать документы электронной подписью и проверять ЭП документов, подписанных другими пользователями. Электронная подпись представляет из себя файл, который хранит набор байтов. Файл ЭП документа хранится в той же папке, что и документ,

Для возможности выполнения этих операций сервис предоставляет открытый API с методами, предназначенными для работы с электронной подписью.

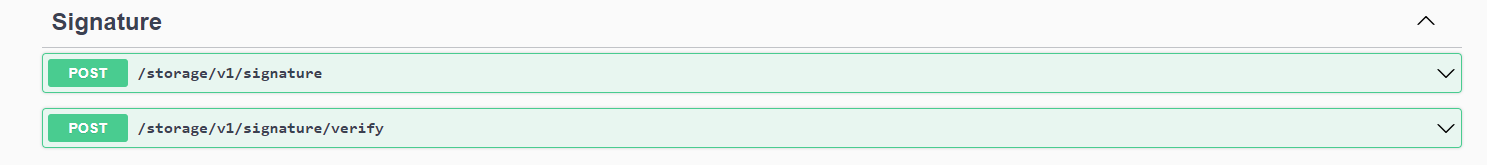


Рисунок 13 – Методы API создания и проверки ЭП.

ЭП позволяет однозначно идентифицировать отправителя и подтвердить, что документ не был подделан или изменен после его подписания. Подписание документов позволяет обеспечить доверие к подписанным документам, а также обеспечивает криптографическую защиту и шифрование данных и целостность документов в процессе их передачи и хранения.

## **Клиентское приложение**

## **Авторизация**

Приложение для ОС Windows представляет из себя мессенджер для общения между пользователями в сети Интернет. При запуске приложения отображается окно авторизации пользователя. Для использования приложения каждый пользователь должен иметь аккаунт в системе. В случае, если аккаунта не существует, пользователь должен зарегистрировать новый аккаунт.

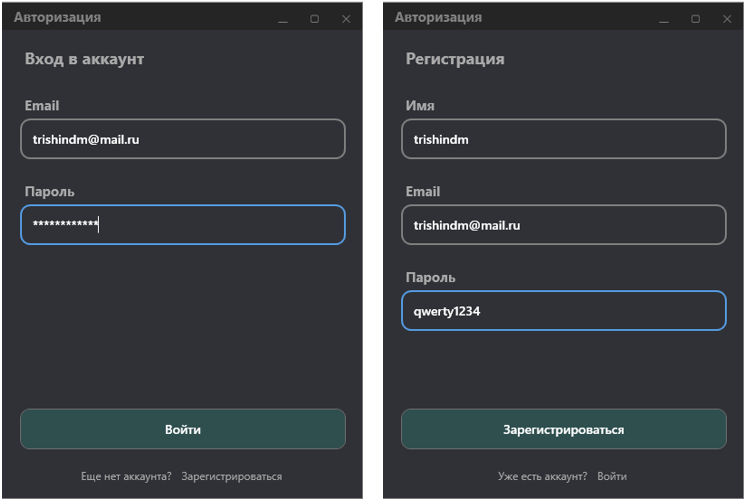


Рисунок 14 – Окна авторизации и регистрации.

Для авторизации пользователю необходимо ввести данные от своей учетной записи:

1. адрес электронной почты, на которую зарегистрирована учетная запись;
2. пароль для входа в учетную запись.

При нажатии на кнопку «Войти» приложение отправляет запрос на удаленный сервер для авторизации пользователя в системе, после чего получает и обрабатывает ответ. В случае, если авторизации была успешной, сервер в качестве ответа вернет JWT токен пользователя, что позволит ему как авторизованному пользователю использовать все возможности приложения.

Для регистрации в системе нового пользователя необходимо ввести следующие данные:

1. уникальное имя пользователя для идентификации в системе;
2. адрес электронной почты для регистрации учетной записи;
3. пароль для входа в созданную учетную запись.

При нажатии на кнопку «Зарегистрироваться» приложение отправляет запрос на удаленный сервер для регистрации пользователя в системе. В случае, если авторизация прошла успешно, то, как и с авторизацией, сервер в качестве ответа вернет JWT токен зарегистрированного пользователя, который далее может быть использован для идентификации пользователя при отправке запросов на сервер.

Перед обработкой каждого запроса авторизации на сервере происходит валидация. Валидация необходима для проверки корректности запроса, то есть тех введённых данных о пользователе, которые клиентское приложение отправило на сервер. В результате валидации могут быть выявлены ошибки, вследствие чего сервер в качестве ответа вернет статус «400 Bad Request» и сообщение, содержащее описание ошибки. Ошибки валидации могут быть использованы для отображения в пользовательском интерфейсе, чтобы дать пользователю возможность увидеть, в каких полях допущены ошибки, и исправить их. Так, к примеру, ошибки валидации во время авторизации могут возникнуть в следующих случаях:

1. пользователь ввел неверный формат адреса электронной почты;
2. пользователь ввел адрес электронной почты, к которой не привязан аккаунт;
3. пользователь ввел неверный пароль для входа в учетную запись.

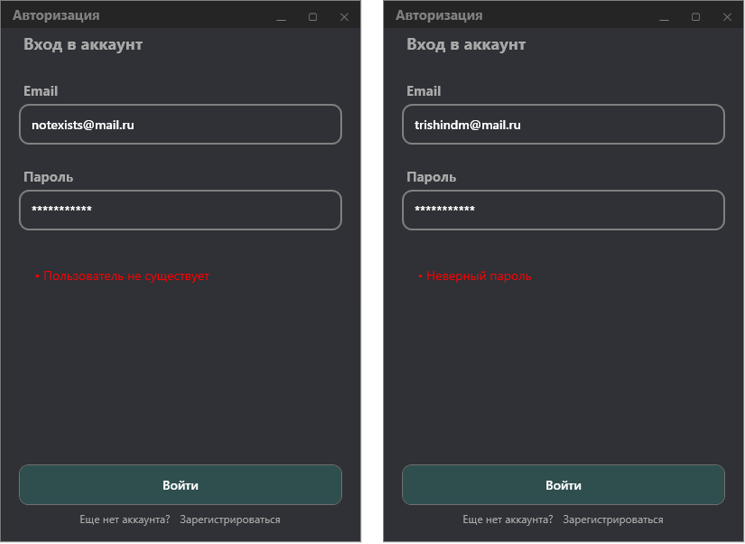


Рисунок 15 – Обработка ошибок в процессе авторизации.

Во время регистрации, как и авторизации, пользователь может допустить ошибку в введённых данных либо использовать недопустимые данные для регистрации. Во время обработки запроса регистрации нового пользователя могут встречаться следующие ошибки:

1. пользователь ввел имя, занятое другим пользователем;
2. пользователь ввел неверный формат адреса электронной почты;
3. пользователь ввел адрес электронной почты, занятой другим пользователем;
4. пользователь ввел недостаточно надежный пароль.

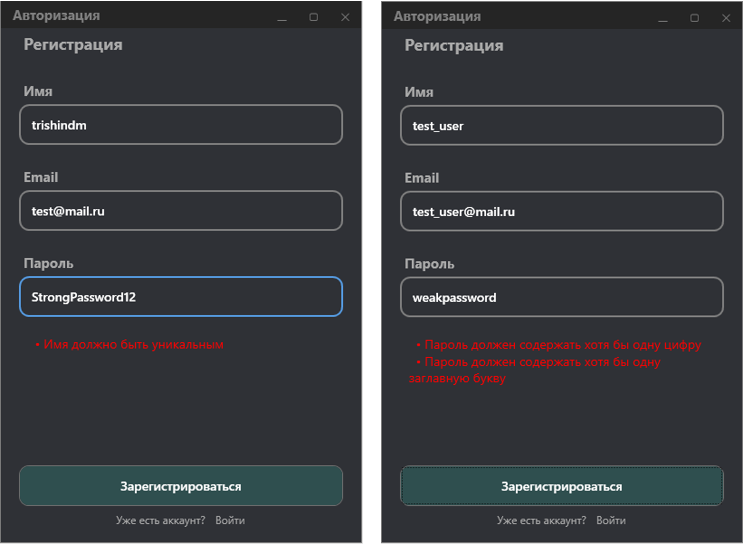


Рисунок 16 – Обработка ошибок в процессе регистрации.

## **Чат**

По завершению авторизации пользователю становится доступно основное приложение. Данное приложение реализует функционал онлайн-чата с другими пользователями, зарегистрированными в системе. Интерфейс главного экрана приложения проиллюстрирован на рисунке ниже.

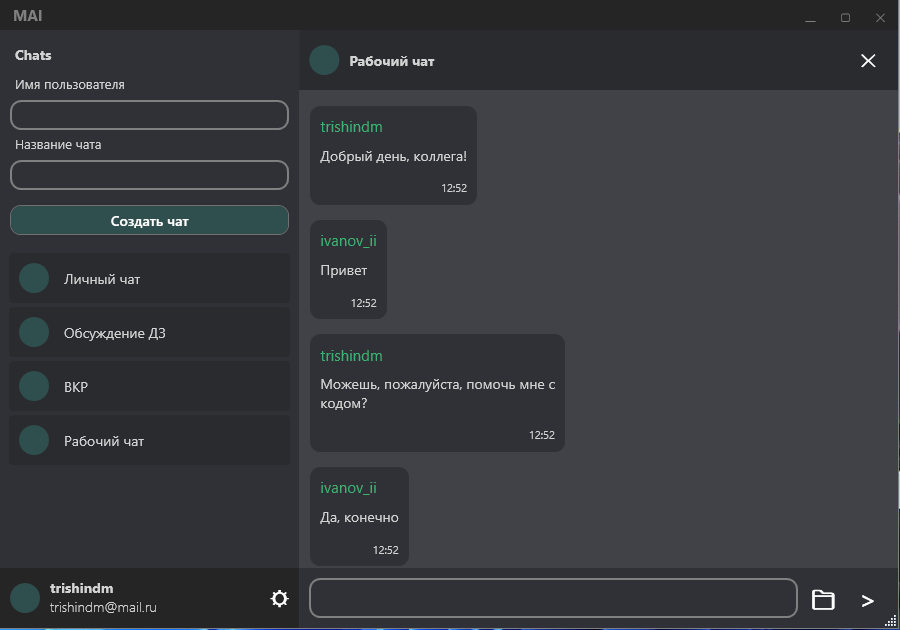


Рисунок 17 – Пользовательский интерфейс приложения.

Пространство пользовательского интерфейса разделено на две части – меню управления приложением и чат. В левой части окна располагается меню управления, которое служит для навигации по приложению и управления функционалом, таким как: создание чатов, просмотр существующих чатов пользователя, информация об авторизованном пользователе, настройки учетной записи.

Для создания нового чата необходимо ввести имя пользователя в поле «Имя пользователя», после чего поисковая строка отобразит список пользователей с похожим именем. Необходимо выбрать одного пользователя, с которым планируется переписка, после чего заполнить поле «Название чата». После выполнения всех действия необходимо нажать кнопку «Создать чат». В результате будет создан новый чат, который будет доступен в списке ниже.

В самом низу пространства меню управления расположена информация об авторизованном пользователе, а также меню настроек, доступное для открытия по кнопке. Настройки позволяют изменить имя пользователя, адрес электронной почты (если, к примеру, доступ к электронной почте был утерян), а также сменить пароль.

В правой части окна располагается открытый чат, который занимает большую область окна. В время общения пользователи могу присылать сообщения и документы. Безопасность общения в чате обеспечена шифрованием данных разработанной криптографической системой на основе эллиптических кривых. Расшифровать содержимое переписки смогут только собеседники, поскольку только они обладают ключами шифрования и дешифрования. В случае, если кому-то из пользователей будет необходимо удалить переписку, можно воспользоваться кнопкой удаления чата в правом верхнем углу экрана.

## **Документы и электронная подпись**

Приложение позволяет обмениваться в чате не только сообщениями, но и документами. Пользователь чата может загрузить документ со своего устройства, после чего загруженный документ будет отправлен на сервер, сохранен в хранилище документов и отправлен в чат в виде сообщения, содержащего документ. По завершению загрузки документа ему присваивается уникальный идентификатор, который закрепляется за пользователем. Этот идентификатор и является электронной подписью документа.

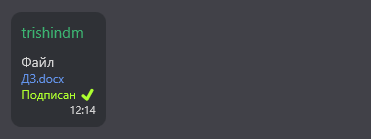


Рисунок 18 – Электронная подпись документа.

Электронная подпись документа доступна всем пользователям и может быть расшифрована для проверки подлинности документа. Электронная подпись документа гарантирует, что документ был подписан человеком, который отправил документ. Если ЭП проверена, под загруженным файлом появляется статус, подтверждающий подлинность электронной подписи.

## **Технические характеристики разработанного приложения**

В результате разработки был создана распределенная система, использующая криптографическую систему шифрования данных на основе эллиптических кривых. Главный технический показатель, определяющий быстродействие системы – это скорость обработки запросов и количество запросов, которые пользователи могут делать в определенный промежуток времени.

Для нагрузочного тестирования веб-сервера был выбран инструмент Apache Bench. Apache Bench – это программа, использующаяся для измерения производительности веб-серверов. Для тестирования был выбран сценарий, при котором 350 пользователей отправляют 20000 запросов на сервер. В результате был получен следующий, результат, приведённый на рисунке ниже.

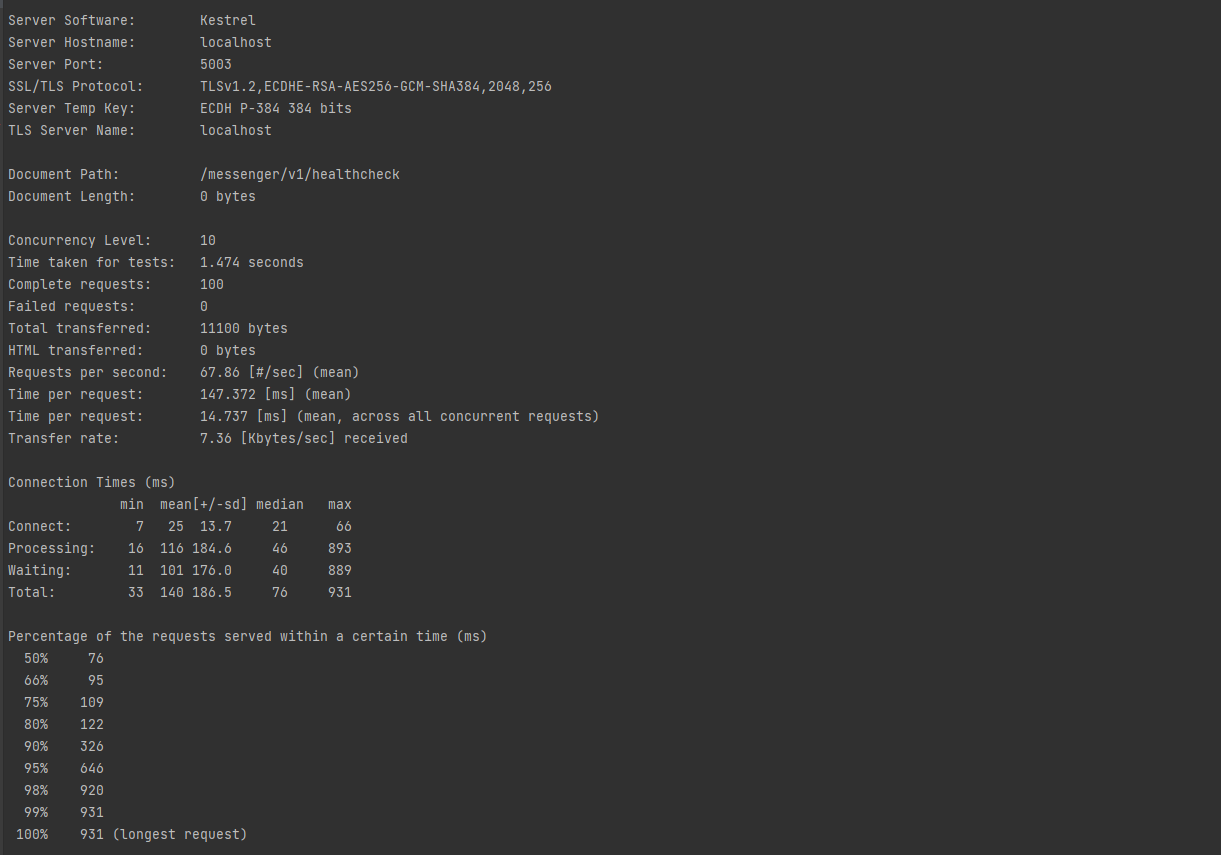


Рисунок 19 – Результат нагрузочного тестирования (количество запросов в секунду и среднее время одного запроса).

Как видно из результатов тестирования, сервер способен выполнить в среднем 67 запросов пользователей в секунду со средним временем выполнения одного запроса 14.737 миллисекунд. Помимо количества запросов в секунду и среднего времени выполнения одного запроса в результате тестирование была получена статистика количества выполненных запросов и времени на их выполнение в процентах. Более 50% запросов, сделанных на веб-сервер, выполнены за минимальное время 76 миллисекунд.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения выпускной квалификационной работы бакалавра были рассмотрены алгоритмы на основе эллиптических кривых над полями конечной характеристики.

В результате проделанной работы анализ различных алгоритмов шифрования на основе эллиптических кривых позволил выбрать оптимальные варианты для разработки распределенной системы. Для работы с эллиптическими кривыми была разработана математическая библиотека исходного кода на языке программирования C#, позволяющая в полной мере работать с используемыми алгоритмами. Была спроектирована и разработана распределенная система на языке программирования C#, использующая математическую библиотеку для работы с алгоритмами шифрования и электронной подписи. Для демонстрации работы распределенной системы было разработано приложения для ПК, реализующее функционал онлайн-чата для безопасного общения пользователей в сети Интернет.

В рамках выполненной работы были решены все поставленные задачи, включая разработку и тестирование распределенной системы, а также анализ результатов. Однако, несмотря на достигнутые цели, есть возможности для дальнейшего улучшения распределенной и увеличения безопасности и быстродействия системы.

Рекомендации по конкретному использованию результатов работы включают внедрение разработанной системы во всех сферы деятельности, где конфиденциальность передаваемых данных играет значимую роль. Так, к примеру, разработанное решение может быть использовано в финансовой сфере либо государственных учреждениях.

Технико-экономическая эффективность внедрения заключается в использовании более устойчивых ко взлому криптографических алгоритмов, которые обеспечивают безопасность данных пользователей в сети Интернет.

Оценка научно-технического уровня выполненной работы в сравнении с лучшими достижениями в этой области показывает, что разработанное решение соответствует современным требованиям безопасности и эффективности.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Moos, D., Peska, P. et al. Comprehensive wellbore stability analysis using quantitative risk assessment// Knowledge Organization – 2017. – Vol. 34, No. 4 . – P. 97–109.
2. Прогноз научно-технологического развития Российской Федерации на период до 2090 года. – URL: [http://government.ru/media /files/41d4b737638891da2184/pdf](http://government.ru/media%20/files/41d4b737638891da2184/pdf) (дата обращения 15.11.2022).
3. Bradbury, A., McLellan, J. Impact Fluid Solutions Biodegradable shale inhibitor developed for water-based muds, Drilling It Safely, Innovating While Drilling, July/August, Onshore Advances Jul 11, 2013. – URL: [http://www.nlm.nih.gov/ pubs/factsheets/umlsmeta.html](http://www.nlm.nih.gov/%20pubs/factsheets/umlsmeta.html%20) (дата обращения 01.02.2023).
4. Двойников, М.В. Технология бурения нефтяных и газовых скважин модернизированными винтовыми забойными двигателями// Дисс. доктора техн. Наук. – Тюмень, 2021. – 360 с.
5. Хегай, В.К. К вопросу оптимизации режимных параметров бурения с учетом крутильных автоколебаний// Известия Коми НЦ УрО РАН. 2020 . № 07–08. – с. 6–10 .
6. Гуреев, В.Н., Мазов, Н.А. Использование библиометрии для оценки значимости журналов в научных библиотеках (обзор)// Научно-техническая информация. Сер. 1. 2019. № 2. – с. 8–19.
7. Статистические показатели российского книгоиздания в 2006 г.: цифры и рейтинги [Электронный ресурс]. - URL: <http://bookhamber.ru/stat_2006.htm> (дата обращения 12.03.2023).

*Рекомендуемое число источников – около 30, в т.ч. не менее 15% на иностранном*