Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования  
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

Кафедра информационно-аналитических систем безопасности

имени профессора Л.С. Берштейна

|  |  |
| --- | --- |
| :  \_\_\_\_\_\_\_\_\_\_\_\_ \*\*\*\*\*\*\*\*\*\*  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | К защите допустить:  Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_ А.Н. Целых  «16» июня 2025 г. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

по образовательной программе

**«Информационно-аналитические системы финансового мониторинга»**

**направления 10.03.01 Информационная безопасность**

|  |  |
| --- | --- |
| на тему: | Цифровая подпись на основе протокола |
|  | нулевого разглашения знания |

Руководитель ВКР:  
доцент кафедры ИАСБ,  
к. т. н., доцент В. С. Васильев

*(подпись, дата)*

Выполнил:  
студентка группы   
КТбо4-12 Т. М. Штенгелова

*(подпись, дата)*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное  учреждение высшего образования «ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**  **(ФГАОУ ВО «ЮФУ»)** | | | | | | | | | | | | | | | | |
| **Институт** | | *Компьютерных технологий и информационной безопасности* | | | | | | | | | | | | | | |  |
| **Кафедра** | | *Информационно-аналитических систем безопасности*  *имени профессора Л.С. Берштейна* | | | | | | | | | | | | | | |  |
| **Группа** | | *КТбо4-12* | | | | | | | | | | | | | | |  |
| **ЗАДАНИЕ**  **для выполнения выпускной квалификационной работы студенту** | | | | | | | | | | | | | | | | |
| **Штенгеловой Татьяне Михайловне** | | | | | | | | | | | | | | | | |
| 1. Тема работы | Цифровая подпись на основе протокола нулевого разглашения | | | | | | | | | | | | | | | |
| знания | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| утверждена приказом по вузу № | | | |  | | | | | от |  | | |  | | | |
| 2. Срок сдачи студентом законченного проекта | | | | | | | | | | | *14.06.2025* | | | | | |
| 3. Исходные данные к проекту | | | | |  | | | | | | | | | | | |
| 1. *Литература и научные статьи по теме криптографии и цифровых* | | | | | | | | | | | | | | | | |
| *подписей.* | | | | | | | | | | | | | | | | |
| 1. *Стандарт банка России СТО БР ФАПИ.СЕК 1.6–2020* | | | | | | | | | | | | | | | | |
| 1. *Техническая документация на используемые криптографические* | | | | | | | | | | | | | | | | |
| *алгоритмы и технологии.* | | | | | | | | | | | | | | | | |
| 1. *ГОСТ Р 34.10–2012* | | | | | | | | | | | | | | | | |
| 1. *Техническая документация на применяемые библиотеки* | | | | | | | | | | | | | | | | |
| 1. *Техническая документация на язык программирования Python* | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| 4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | | | | | | | | |
| 1. *Введение* | | | | | | | | | | | | | | | | |
| 1. *Описание основных положений цифровой подписи* | | | | | | | | | | | | | | | | |
| 1. *Анализ существующих протоколов цифровой подписи.* | | | | | | | | | | | | | | | | |
| 1. *Анализ протоколов нулевого разглашения* | | | | | | | | | | | | | | | | |
| 1. *Разработка алгоритма цифровой подписи с применением протокола нулевого разглашения знания* | | | | | | | | | | | | | | | | |
| 1. *Программная реализация и тестирование алгоритма* | | | | | | | | | | | | | | | | |
| 5. Перечень графического материала (с точным указанием обязательных чертежей): | | | | | | | | | | | | | | | |
| 1. *Блок-схема алгоритма цифровой подписи* | | | | | | | | | | | | | | | |
| 1. *Блок-схема алгоритма решения* | | | | | | | | | | | | | | | |
| 1. *Блок-схема работы приложения* | | | | | | | | | | | | | | | |
| 1. *Внешний вид пользовательского интерфейса* | | | | | | | | | | | | | | | |
| 1. *Схема сетевого взаимодействия компонентов* | | | | | | | | | | | | | | | |
| 6. Консультанты по проекту (с указанием относящихся к ним разделов проекта) | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| 7. Дата выдачи задания | | | *8.02.2025* | | | | | | | | | | | | |
| **Руководитель** | | | | | |  | | | | | |  | | | |
|  | | | | | | (подпись) | | | | | |  | | | |
| Задание принял к исполнению | | | | | | | | *8.02.2025* | | | | | | |  |
|  | | | | | | | (дата) | | | | | | |  | |
| **Подпись студента** | | | | | | |  | | | | | | |  | |

# АННОТАЦИЯ

В данной работе рассматривается разработка и реализация цифровой подписи, основанной на протоколах нулевого разглашения знания (ZKP). Цель исследования — создание безопасного и эффективного механизма аутентификации, который обеспечивает подтверждение подлинности данных без раскрытия самой информации. В работе подробно анализируются теоретические основы протоколов нулевого разглашения, их свойства и преимущества в контексте цифровых подписей. Особое внимание уделено практической реализации алгоритма цифровой подписи с использованием ZKP, включая описание архитектуры, алгоритмических решений и методов оптимизации. Также исследуются области применения данной технологии в современных системах безопасности, таких как электронный документооборот, блокчейн и аутентификация пользователей. Результаты работы демонстрируют, что цифровая подпись на основе протоколов нулевого разглашения знания обеспечивает высокий уровень конфиденциальности и устойчивости к атакам, что делает её перспективным инструментом для защиты информации в цифровой среде.

# ANNOTATION

This paper discusses the development and implementation of a digital signature based on zero-knowledge protocols (ZKP). The purpose of the study is to create a secure and effective authentication mechanism that provides confirmation of the authenticity of data without disclosing the information itself. The paper analyzes in detail the theoretical foundations of zero-knowledge protocols, their properties, and advantages in the context of digital signatures. Particular attention is paid to the practical implementation of the digital signature algorithm using ZKP, including a description of the architecture, algorithmic solutions, and optimization methods. It also explores the areas of application of this technology in modern security systems, such as electronic document management, blockchain and user authentication. The results of the work demonstrate that a digital signature based on zero-knowledge protocols provides an important level of confidentiality and resistance to attacks, which makes it a promising tool for protecting information in the digital environment.

## Содержание

[Список принятых сокращений 8](#_Toc199975299)

[ВВЕДЕНИЕ 9](#_Toc199975300)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ЦИФРОВОЙ ПОДПИСИ 11](#_Toc199975301)

[1.1 Определение и роль цифровой подписи 11](#_Toc199975302)

[1.2 Сравнительный анализ традиционных методов цифровой подписи 13](#_Toc199975311)

[1.2.1 Стандарт цифровой подписи RSA 14](#_Toc199975312)

[1.2.2 Стандарт цифровой подписи DSA 14](#_Toc199975313)

[1.2.3 Стандарт цифровой подписи ECDSA 15](#_Toc199975326)

[1.2.4 Стандарт цифровой подписи EdDSA 16](#_Toc199975327)

[1.2.5 Сводный анализ протоколов цифровой подписи 17](#_Toc199975328)

[Глава 2. Протоколы нулевого разглашения знания (ZKP) 19](#_Toc199975329)

[2.1 Принципы работы протоколов нулевого разглашения знания 19](#_Toc199975330)

[2.3 Анализ протоколов нулевого разглашения знания для реализации цифровой подписи 23](#_Toc199975344)

[2.3.1 Протокол Шнорра 23](#_Toc199975345)

[2.3.2 Протокол zk-SNARKs 25](#_Toc199975346)

[2.3.3 Протокол zk-STARK 27](#_Toc199975347)

[2.3.4 Протокол Bulletproofs 28](#_Toc199975348)

[2.4 Сводный анализ протоколов нулевого разглашения 30](#_Toc199975349)

[Глава 3. Практическая реализации цифровой подписи 32](#_Toc199975350)

[3.1Проектирование алгоритма работы 32](#_Toc199975351)

[3.2Обоснование выбора программных средств 34](#_Toc199975352)

[3.3Описание программной реализации решения 36](#_Toc199975353)

[ГЛАВА 4. ТЕСТИРОВАНИЕ 44](#_Toc199975356)

[4.1Функциональное тестирование 44](#_Toc199975358)

[4.2Нагрузочное тестирование 45](#_Toc199975366)

[4.3Стресс-тестирование 46](#_Toc199975378)

[4.4Оценка масштабируемости 46](#_Toc199975384)

[4.5 Анализ безопасности 47](#_Toc199975391)

[ГЛАВА 5. ОБЛАСТИ ПРИМЕНЕНИЯ РЕШЕНИЯ 51](#_Toc199975392)

[5.1Применение в банковской сфере 51](#_Toc199975393)

[5.2Применение в области интернета вещей 52](#_Toc199975394)

[5.3Блокчейн и электронный документооборот 53](#_Toc199975395)

[5.4Аутентификация без пароля 54](#_Toc199975396)

[ЗАКЛЮЧЕНИЕ 56](#_Toc199975397)

[Список использованных источников 58](#_Toc199975402)

[Приложение А – Программный код решения 59](#_Toc199975410)

## Список принятых сокращений

PKI – инфраструктура открытых ключей;

ZKP – протоколы нулевого разглашения знания;

DSA – Digital Signature Algorithm;

EdDSA – Edwards-curve Digital Signature Algorithm;

ECDSA – Elliptic Curve Digital Signature Algorithm;

PAKE - Password Authenticated Key Exchange;

ЦП – цифровая подпись;

ОС – операционная система.

Введение

Стремительное развитие цифровых коммуникаций и постоянное расширение сферы электронного документооборота неизбежно приводят к росту информационных рисков. Традиционные механизмы проверки подлинности нередко оказываются уязвимыми к современным видам атак, требуя интеграции более совершенных криптографических схем, способных обеспечить гарантированную защиту без ухудшения пользовательского опыта или значительных затрат ресурсов. Цифровая подпись выступает ключевым элементом современных систем безопасности, так как она подтверждает авторство и целостность данных, что особенно важно при росте объёмов электронного документооборота и удалённого взаимодействия. Однако даже проверенные временем алгоритмы цифровых подписей нуждаются в доработке для соответствия новым требованиям к приватности и минимизации потенциальных областей утечек информации. Протоколы нулевого разглашения знания предоставляют уникальные возможности для повышения уровня доверия к процессу аутентификации благодаря тому, что позволяют доказывать владение секретом без раскрытия самой информации, что расширяет возможности приватной верификации и способствует достижению высокого уровня защиты.

Применение ZKP в контексте цифровой подписи позволяет избежать прямого раскрытия приватного ключа или исходного сообщения в процессе верификации, что в итоге минимизирует риски компрометации и увеличивает устойчивость к разнообразным видам атак, включая адаптивные и многократные попытки вскрытия. В результате может быть достигнут баланс между эффективностью выполнения криптографических операций и высоким уровнем защищённости, что является важнейшим фактором при проектировании систем корпоративного уровня и распределённых платформ.

Актуальность данного исследования обусловлена необходимостью выработки практических рекомендаций по внедрению цифровых подписей с поддержкой протоколов нулевого разглашения знания в реальных информационных средах. Существующие реализации ZKP и цифровых подписей, несмотря на прогресс в области теории, пока не нашли широкого распространения в массовых приложениях из-за сложности настройки, недостатка оптимизированных алгоритмов и нехватки инструментов для интеграции в уже действующие системы. Данное противоречие между теоретическим потенциалом и практической реализуемостью подчеркивает необходимость комплексного анализа существующих методов, выявления их преимуществ и ограничений, а также разработки новых решений, оптимизированных под реальные условия.

Цель исследования состоит в том, чтобы провести сравнительный анализ современных подходов к построению цифровых подписей на основе протоколов нулевого разглашения знания и на этой основе сформулировать практические рекомендации по их интеграции в существующие системы безопасности. Предполагается изучить основные классы ZKP (интерактивные и не интерактивные), оценить их применимость в контексте цифровой подписи, провести экспериментальное сравнение по показателям скорости формирования и проверки, а также по объёму передаваемых данных. Важным аспектом работы станет анализ соответствия предложенных схем современным нормативным требованиям и стандартам безопасности, что позволит оценить перспективы их применения в промышленной среде.

Результаты исследования могут стать основой для создания нового поколения средств аутентификации и защиты данных, способных обеспечить высокий уровень конфиденциальности, гибкость конфигурирования и устойчивость к атакам даже в условиях высоко динамичной и распределённой инфраструктуры.

Глава 1. Теоретические основы цифровой подписи

1.1 Определение и роль цифровой подписи

Цифровая подпись представляет собой криптографический механизм, обеспечивающий подлинность, целостность и отказоустойчивость электронных документов. Она основана на применении асимметричного шифрования и хеширования, что позволяет установить связь между содержимым сообщения и его автором без необходимости раскрывать секретный ключ подписанта третьим лицам. Ключевыми компонентами цифровой подписи выступают пара ключей — открытый и закрытый, а также односторонняя хеш-функция, результаты работы которой служат входными данными для алгоритма подписи.

Асимметричное шифрование играет центральную роль в механизме генерации и верификации подписей. В классическом варианте такой схемы подписант обладает секретным (закрытым) ключом, с помощью которого он формирует подпись, и публикует соответствующий публичный (открытый) ключ для проверки подписи любой стороной. Алгоритмически процесс создания подписи включает хеширование исходного сообщения и последующее шифрование хеш-суммы закрытым ключом подписанта. При проверке подписи верификатор извлекает открытую часть ключа, расшифровывает полученную подпись и сравнивает результат с хешем принимаемого документа. Достоверное совпадение обоих хешей свидетельствует о том, что документ не был изменён после подписания и что подпись действительно создана владельцем закрытого ключа.

Хеширование представляет собой преобразование произвольного объёма данных в строку фиксированной длины с помощью односторонней функции. Основные требования к таким функциям — устойчивость к коллизиям, предобразу. Устойчивость к коллизиям означает, что невозможно найти две различные входные строки, которые дают одинаковый хеш; устойчивость к предобразу обеспечивает невозможность по заданному хешу восстановить исходные данные или выбрать другую строку с тем же хеш-значением. Именно эти свойства гарантируют, что изменение даже одного бита в документе приведёт к кардинальному изменению результата хеширования, что позволяет обнаружить любую несанкционированную модификацию подписываемой информации.

Целостность электронного документа подтверждается тем, что подписанная хеш-сумма должна совпасть с вновь вычисленной при проверке. Если документ был изменён, хеш-функция выдаст иное значение, что приведёт к несоответствию и к невозможности верификации подписи.

Аутентичность обеспечивается криптографической связью между подписью и владельцем закрытого ключа: лишь тот, кто владеет секретным ключом, может сгенерировать корректную подпись, проверяемую публичным ключом.

Отказоустойчивость означает, что подписант не может отвергнуть факт создания подписи впоследствии, поскольку доказательством его участия служит криптографическое сообщение, проверяемое публичным ключом.

Современные алгоритмы цифровой подписи, например RSA, DSA или схемы на эллиптических кривых, используют различные математические основы, однако структура остаётся унифицированной: хеширование документа и асимметричное преобразование хеша. В алгоритме RSA подпись вычисляется как возведение хеш-суммы в степень закрытого показателя по модулю произведения больших простых чисел, а проверка — обратной операцией с использованием открытой экспоненты. Схемы на эллиптических кривых устраняют некоторые недостатки классических алгоритмов, снижая размер ключей и повышая скорость операций за счёт более сложной математической структуры кривых.

Важнейшим компонентом общей инфраструктуры цифровых подписей выступают сертифицированные центры (ЦС), которые обеспечивают управляемое распространение открытых ключей и проверку подлинности принадлежности ключей субъектам. Инфраструктура публичных ключей (PKI) гарантирует, что публичный ключ действительно принадлежит указанному лицу или системе, что способствует укреплению доверия между участниками электронного взаимодействия.

Таким образом, цифровая подпись представляет собой сочетание асимметричного шифрования и хеширования, реализующее свойства целостности, аутентичности и отказоустойчивости электронных документов. Базовые компоненты цифровой подписи — секретный ключ для создания подписи, публичный ключ для её проверки и стойкая к коллизиям хеш-функция — образуют надёжный и проверенный временем механизм обеспечения безопасности в самых различных прикладных сценариях, от защищённой электронной переписки до распределённых регистров блокчейн-систем.

1.2 Сравнительный анализ традиционных методов цифровой подписи

В классических схемах цифровой подписи основополагающим принципом является использование асимметричных криптографических алгоритмов, в которых каждая сторона оперирует парой ключей: приватным для создания подписи и публичным для её проверки. Несмотря на различия в математических основаниях, все схемы обеспечивают гарантию целостности и аутентичности сообщения, а также невозможность подделки подписи без знания приватного ключа. Наиболее распространёнными и детально исследованными алгоритмами являются RSA, DSA, ECDSA, EdDSA. Ниже приводится их подробное описание, а также сравнительный анализ по критериям безопасности и производительности.

1.2.1 Стандарт цифровой подписи RSA

RSA - это один из самых известных и широко используемых алгоритмов криптографии. Он применяется как для шифрования, так и для цифровых подписей. Сам по себе алгоритм основан на факторизации больших чисел. Злоумышленнику сложно разложить произведение двух больших простых чисел на эти самые простые числа.

Схема RSA основывается на сложности задачи факторизации больших целых чисел. При генерации ключевой пары выбирают два крупных простых числа p и q, после чего вычисляют модуль n и значение функции Эйлера, в соответствии с формулами:

(1.1)

(1.2)

Приватным ключом служит число d, обратное по модулю φ(n) к экспоненте e, использующейся в публичном ключе (n, e). Для подписи сообщения m сначала вычисляют хэш h(m), затем формируют подпись в соответствии с формулой:

(1.3)

Проверка проводится вычислением и сравнением с хэшем h(m) по формуле:

(1.4)

К сильным сторонам RSA относят простоту реализации и широкое распространение. Однако при равных уровнях безопасности ключи RSA существенно длиннее, что негативно сказывается на времени выполнения операций возведения в степень и занимаемой памяти для хранения ключей.

1.2.2 Стандарт цифровой подписи DSA

Стандарт цифровой подписи DSA — это в первую очередь стандарт DSS, который включает в себя спецификацию алгоритма. Однако, он также охватывает другие аспекты, такие как параметры ключей, требования безопасности и использование с различными хеш-функциями.

DSA использует свойства группы вычетов по простому модулю и сложность задачи дискретного логарифма. Генерация параметров схемы включает выбор большого простого p и его простого делителя q (обычно 160–256 бит), а также генератора g порядка q.

Публичный ключ Q вычисляется по формуле

(1.5)

где x — приватный ключ.

Для подписи сообщения вычисляют хэш h(m), случайное k в диапазоне [1, q − 1], затем рассчитывают:

(1.6)

(1.7)

Проверка сводится к вычислениям по формулам и сравнению v с r:

(1.8)

(1.9)

(1.10)

Преимуществом DSA является более короткая длина подписи по сравнению с RSA при сопоставимом уровне безопасности. В то же время ограничение на использование уникального случайного k для каждой подписи и высокая чувствительность к утечкам этой величины предъявляют строгие требования к генератору случайных чисел.

1.2.3 Стандарт цифровой подписи ECDSA

ECDSA представляет собой адаптацию DSA в рамках группы точек на эллиптической кривой над конечным полем. Генерация кривых и параметров стандартизована в ряде рекомендаций, например, NIST или SEC.

Приватный ключ x выбирают в диапазоне [1, n − 1], где n — порядок базовой точки G на кривой. Публичный ключ вычисляется по формуле

(1.11)

Подпись формируется аналогично DSA: берут случайное k, вычисляют точку по формуле:

(1.12)

определяют значения

(1.13)

(1.14)

Для проверки проверяют, что r и s лежат в допустимом диапазоне, затем вычисляют

(1.15)

(1.16)

(1.17)

подпись считается верной, если x-координата R по модулю n равна r.

Главным достоинством ECDSA является существенное уменьшение размеров ключей и подписей при сохранении требуемого уровня криптостойкости, что обеспечивает более высокую производительность и экономию ресурсов при ограниченных вычислительных мощностях.

1.2.4 Стандарт цифровой подписи EdDSA

EdDSA — это относительно новый, но быстро набирающий популярность алгоритм цифровой подписи. Отличается от ECDSA своей безопасностью, простотой реализации и высокой производительностью. Он спроектирован так, чтобы избежать многих проблем, с которыми сталкиваются более старые алгоритмы.

EdDSA использует эллиптические кривые Эдвардса, которые имеют определенные свойства, облегчающие разработку безопасных и эффективных реализаций. При этом использует детерминированную подпись. Это означает, что для одного и того же сообщения и приватного ключа всегда будет генерироваться одна и та же подпись. Это устраняет необходимость в генераторе случайных чисел для каждой подписи, что упрощает реализацию и снижает риск уязвимостей, связанных с генерацией случайных чисел. Алгоритм цифровой подписи EdDSA стандартизирован в документах RFC 8032 и NIST SP 800-175B.

В отличие от многих других кривых, используемых в криптографических приложениях, формулы для EdDSA являются «полными»; они применимы ко всем точкам на кривой без исключений. В частности, знаменатели отличны от нуля для всех входных точек.

Для расчета подписи вычисляется секретное r

(1.18)

точки публичного ключа R и хеш значение:

(1.19)

(1.20)

Вычисляется подпись по формуле:

(1.21)

Для проверки подписи вычисляют

(1.22)

(1.23)

а затем проверяют равенство .

Преимущества EdDSA заключаются в высокой производительности в различных условиях и отсутствии уникального случайного числа для каждой подписи. Алгоритм более устойчив к атакам по сторонним каналам и использует небольшие открытые ключи (32 или 57 байт) и подписи (64 или 114 байт) для Ed25519 и Ed448 соответственно. Применяемые формулы являются "полными", то есть они действительны для всех точек на кривой без исключений, что устраняет необходимость в дорогостоящей точечной проверки ненадежных общедоступных значений.

1.2.5 Сводный анализ протоколов цифровой подписи

По итогам анализа была составлена сравнительная таблица по стандартам цифровых подписей. Эта таблица суммирует основные аспекты различных алгоритмов цифровой подписи, включая их тип, безопасность, производительность и другие характеристики

Таблица 1. – Сравнение протоколов цифровой подписи

| Параметры | RSA | DSA | ECDSA | EdDSA |
| --- | --- | --- | --- | --- |
| Тип | Ассиметричный алгоритм | Стандарт цифровой подписи | Ассиметричный алгоритм | Ассиметричный алгоритм |
| Безопасность | На основе сложности факторизации больших чисел | На основе сложности вычисления дискретного логарифма | На основе эллиптических кривых | На основе эллиптических кривых |
| Производительность | Низкая скорость подписи и проверки | Быстрая проверка, медленная генерация | Быстрая проверка и высокая эффективность | Высокая скорость и эффективность |
| Длина ключа | Большая (2048–3072 бит) | Требует больших параметров | Меньшая, с аналогичным уровнем безопасности | Небольшая и эффективная |
| Юридическая значимость | Признан во многих юрисдикциях | Признан во многих юрисдикциях | Признан во многих юрисдикциях | Признан в некоторых юрисдикциях |
| Уязвимости | К атакам на факторизацию и неправильную реализацию | Уязвимость к повторному использованию k | Уязвимость к неправильному k и слабым кривым | Уязвимость в сложности реализации |
| Поддержка шифрования | Да | Нет | Нет | Нет |

В соответствии с таблицей 1.1 видно, что у всех приведенных протоколов есть свои недостатки. При этом наиболее быстрым, эффективным и с небольшим размером ключа является протокол EdDSA. Однако стоит учитывать сложность его реализации.

Глава 2. Протоколы нулевого разглашения знания (ZKP)

ZKP традиционно не рассматривались как прямая замена классическим алгоритмам цифровой подписи. Однако, существуют способы интеграции в процесс создания и проверки цифровых подписей для достижения дополнительных свойств, таких как конфиденциальность и доказуемая корректность. Этот раздел посвящен анализу таких протоколов ZKP, которые могут быть применимы для цифровой подписи, либо непосредственно, либо в сочетании с традиционными методами.

2.1 Принципы работы протоколов нулевого разглашения знания

Сущность протоколов нулевого разглашения знания заключается в возможности убедить проверяющую сторону в истинности некоторого утверждения без раскрытия какой-либо дополнительной информации, кроме самого факта его достоверности. Основная идея ZKP строится на использовании криптографических приёмов и математических проблем, сложность решения которых служит гарантией безопасности. При этом доказывающий (протокол выступает заёмщиком информации) и проверяющий (верификатор) обмениваются сообщениями в заранее согласованной схеме, после чего верификатор убеждается в том, что обладатель секрета действительно владеет необходимыми данными, но не получает сведений об этом секрете сверх заявленного факта.

Ключевыми свойствами протоколов нулевого разглашения знания являются полнота, надёжность и нулевое разглашение. Свойство полноты гарантирует, что если доказывающий следует протоколу и утверждение истинно, то верификатор примет его доказательство. Надёжность означает, что ни одна вычислительно ограниченная сторона не сможет убедить верификатора в ложности утверждения, превышая некоторый незначительный шанс случайного успеха. Нулевое разглашение подразумевает, что вся информация, получаемая верификатором во время исполнения протокола, может быть сымитирована без знания секрета; таким образом реальное доказательство не раскрывает новых сведений.

В зависимости от способа взаимодействия сторон протоколы нулевого разглашения знания делятся на интерактивные и не интерактивные. Интерактивные протоколы предполагают многократный обмен сообщениями между доказывающим и верификатором. В классических схемах, предложенных Гольдвассером, Микали и Рэккоффом, доказательство состоит из череды раундов, в каждом из которых доказывающий отправляет скрытое сообщение, затем верификатор выбирает случайный вызов, и на него поступает ответ. По итогам всех раундов проверяются условия, которые гарантируют достижение требуемых свойств.

К интерактивным схемам относятся знаменитый протокол доказательства существования гамильтонова цикла в графе, протоколы на основе симметричных групп и протоколы с обязательствами. Примером является схема Фьорда-Филда, где владение гамильтоновым циклом доказывается без раскрытия самой структуры цикла. Такие решения широко исследуются благодаря строгому математическому обоснованию и возможности получения высокой криптостойкости даже при относительно небольших размерах ключей.

Не интерактивные протоколы нулевого разглашения минимизируют обмен сообщениями, часто сводясь к единственной передаче доказательства от доказывающего к верификатору. Основной механизм преобразования интерактивного доказательства в не интерактивное — преобразование Фиата–Шамира, предполагающее замену вызовов верификатора на хеш-функции, которые генерируют «случайность» на основе входных данных доказательства. Благодаря этому протокол приобретает устойчивость к адаптивным атакам, а время верификации и объём передаваемых данных существенно уменьшаются.

Протоколы преобразования Фиата–Шамира и последующие усовершенствования лежат в основе ряда современных цифровых подписй, обладающих защитой на основе ZKP. В частности, в криптографии широко применяются протоколы на базе эллиптических кривых и алгоритмы, такие как подпись Шнорра, где доказательство знания дискретного логарифма подаётся в не интерактивном формате. Данные методы позволяют создавать компактные подписи с низкой вычислительной задержкой и высокой степенью надёжности.

Классификация протоколов по взаимодействию не исчерпывает всех возможных вариантов. Дополнительным критерием может служить характер доказательства: доказательство знания секрета или доказательство утверждения о некотируемом состоянии. В первом случае протокол подтверждает, что доказывающий действительно владеет некоторым секретом (паролем, ключом), во втором — что секрет обладает определёнными свойствами (например, число принадлежит заданному диапазону) без раскрытия самого числа.

Существующие подходы можно разделить на интерактивные и не интерактивные протоколы, а также на более продвинутые конструкции типа zk-SNARKs и zk-STARKs. Помня о поставленных в работе целях – создании безопасного и эффективного механизма аутентификации без утечки секретных данных, рассмотрим детально каждую группу схем и критерии их выбора для различных практических задач.

Интерактивные протоколы нулевого разглашения знания представляют собой классический подход, где взаимодействие между доказывающей и проверяющей сторонами состоит из нескольких раундов обмена сообщениями. В каждом раунде проверяющий направляет случайный вызов, а доказывающий формирует ответ, демонстрирующий знание секрета без его раскрытия. Ключевыми преимуществами интерактивных схем являются адаптивность к различным классам задач, высокая гибкость в выборе параметров безопасности и относительная простота реализации на базе стандартных криптографических предпосылок. Однако многократное взаимодействие увеличивает задержки и требует живого канала связи между участниками, что не всегда удобно в распределённых системах или условиях высокой латентности сети.

Не интерактивные схемы ZKP устраняют необходимость многократного обмена сообщениями, используя генерацию мурчащей строки общих случайных данных или применяя преобразование Фиата–Шамира. В таких протоколах доказывающий один раз вычисляет и публикует доказательство, а получатель может проверить его асинхронно, без дополнительного раунда общения. Простота верификации и возможность однократной публикации делают не интерактивные схемы привлекательными для использования в блокчейн-системах и распределённых реестрах. С другой стороны, необходимость надёжной генерации CRS или доверенной настройки схемы может стать узким местом с точки зрения безопасности и прозрачности.

В контексте аутентификации протоколы нулевого разглашения знания применяются для построения защищённых схем входа в систему, где пользователь доказывает знание пароля без его передачи серверу. Это исключает возможность перехвата или утечки секретных данных. Такие схемы позволили создать протоколы идентификации, устойчивые к атакам воспроизведения, «человек-посередине» и анализу трафика. Примером могут служить протоколы PAKE, которые интегрируют ZKP для подтверждения знания пароля и одновременно устанавливают защищённый канал связи.

Применение ZKP в современных системах безопасности выходит за рамки классической аутентификации. В блокчейне и распределённых реестрах технологии нулевого разглашения позволяют обеспечивать анонимность и конфиденциальность транзакций. Платформа Zcash реализует zk-SNARKs — особый тип не интерактивных доказательств с малыми объёмами доказательств и временем верификации, что делает возможным проведение конфиденциальных переводов с гарантией корректности.

Таким образом, сущность протоколов нулевого разглашения знания объединяет в себе сложные математические конструкции и практические методы оптимизации, позволяющие достигать высокого уровня конфиденциальности и надёжности в задачах аутентификации и цифровых подписей. Классификация на интерактивные и не интерактивные схемы отражает основные подходы к построению обмена между сторонами, которые, в свою очередь, находят широкое применение в современных системах безопасности и распределённых приложениях.

2.3 Анализ протоколов нулевого разглашения знания для реализации цифровой подписи

2.3.1 Протокол Шнорра

Протокол Шнорра — это интерактивный протокол доказательства знания дискретного логарифма, разработанный Клаусом Шнорром в 1990 году. Он широко используется в криптографии для аутентификации, цифровых подписей и других задач. Его ключевой особенностью является простота, эффективность и доказанная безопасность.

Протокол Шнорра основан на сложности задачи дискретного логарифма в циклической группе. Доказывающий должен доказать проверяющему, что он знает дискретный логарифм для заданного публичного ключа. При этом доказывающий не должен раскрывать этот приватный ключ.

Протокол Шнорра, будучи одним из основоположников современных протоколов цифровой подписи и аутентификации, обладает рядом преимуществ и недостатков, которые важно учитывать при выборе его для конкретного приложения.

Достоинства протокола Шнорра.

* Протокол Шнорра относительно прост для понимания, реализации и анализа безопасности. Его алгоритм легко описывается математически, и его структура позволяет проводить проверки безопасности.
* Операции, необходимые для генерации и проверки подписей, относительно нетребовательны к вычислительным ресурсам. Это делает протокол Шнорра подходящим для устройств с ограниченной вычислительной мощностью.
* Безопасность протокола Шнорра хорошо изучена и основывается на сложности проблемы дискретного логарифма. Он обеспечивает надежную защиту от подделки подписей. Существуют формальные доказательства безопасности, которые подтверждают устойчивость протокола к различным атакам при соблюдении определенных условий.
* Линейность протокола позволяет создавать агрегированные подписи. Несколько подписей, созданных разными пользователями, могут быть объединены в одну, которая занимает меньше места, чем сумма отдельных. Агрегация подписей может значительно повысить эффективность в системах, где требуется обрабатывать большое количество подписей, например, в блокчейне.
* Протокол Шнорра может быть использован для цифровых подписей и для аутентификации, доказательства членства и других криптографических задач.

Недостатки протокола Шнорра.

* Безопасность протокола Шнорра полностью зависит от сложности проблемы дискретного логарифма.
* Плохая реализация или слабая хеш-функция могут привести к уязвимостям.
* Протокол Шнорра не является устойчивым к атакам с использованием квантовых компьютеров.
* Размер подписи Шнорра может быть больше, чем у некоторых других алгоритмов цифровой подписи. Это может быть недостатком в приложениях, где важен минимальный размер подписи.
* В стандартном протоколе Шнорра отсутствуют встроенные механизмы для обеспечения нулевого разглашения в отношении самого сообщения или личности подписывающего.

Протокол Шнорра является инструментом в криптографическом наборе, когда требуется простота, эффективность и возможность агрегирования подписей. Однако, необходимо учитывать его недостатки, такие как зависимость от проблемы дискретного логарифма, уязвимость к атакам по сторонним каналам и отсутствие квантовой устойчивости.

2.3.2 Протокол zk-SNARKs

zk-SNARK — это криптографический протокол, который позволяет одной стороне доказать, что она знает некоторое свидетельство, не раскрывая при этом ни какой-либо другой информации о нем, кроме самого факта его существования. Он позволяет доказать правильность вычисления без раскрытия входных данных, с высокой эффективностью: доказательство краткое, и проверка не требует интерактивных операций. Это делает zk-SNARK особенно привлекательными для различных приложений, где важны конфиденциальность и масштабируемость.

Преимущества zk-SNARK.

* Доказывающий может убедить проверяющего в истинности утверждения, не раскрывая никакой дополнительной информации о свидетеле или самом утверждении.
* Размер доказательства, создаваемого zk-SNARK, очень мал, обычно несколько сотен байт, независимо от сложности вычисления.
* После создания доказательство может быть проверено без какого-либо взаимодействия. Это упрощает процесс проверки и делает zk-SNARK подходящими для систем, где интерактивность невозможна или нежелательна.
* zk-SNARK доказывают, что утверждение верно и что доказывающий действительно знает свидетеля, который делает утверждение верным. Это свойство предотвращает создание фальшивых доказательств без знания секрета.
* Проверка доказательств выполняется очень быстро.
* zk-SNARK могут быть использованы для доказательства правильности широкого спектра вычислений, от простых арифметических операций до сложных алгоритмов машинного обучения.

Недостатки zk-SNARK.

* Для генерации общих параметров, необходимых для работы протокола, требуется доверенная установка. Если в процессе доверенной установки произошла утечка секретной информации, то злоумышленник может создавать поддельные доказательства, которые будут приниматься как действительные.
* zk-SNARK — это очень сложные криптографические конструкции, требующие глубокого понимания математики и криптографии. Реализация и аудит требует высококвалифицированных специалистов и значительных усилий.
* Генерация доказательства может быть очень ресурсоемкой и требовать значительного времени и вычислительной мощности.
* zk-SNARK не являются устойчивыми к атакам с использованием квантовых компьютеров.
* При изменении вычисления, которое необходимо доказать, часто требуется новая доверенная установка и генерация новых общих параметров.
* Доверенная установка создает элемент непрозрачности. Пользователи должны доверять процессу установки параметров, даже если не имеют возможности его проверить.

zk-SNARK — сложный инструмент. Они предоставляют уникальные возможности по обеспечению конфиденциальности и масштабируемости, но требуют тщательного рассмотрения связанных с ними недостатков, особенно доверенной установки и сложности реализации.

2.3.3 Протокол zk-STARK

zk-STARK относительно новый тип протоколов нулевого разглашения, который предлагает альтернативу zk-SNARK. Основным отличием и главным преимуществом является отсутствие необходимости в доверенной установке, которая является одним из самых серьезных ограничений zk-SNARK. Они используют другие криптографические конструкции, которые позволяют создавать доказательства без необходимости в предварительно сгенерированных общих параметрах.

zk-STARK представляют собой многообещающую технологию, предлагая уникальные преимущества, в плане безопасности и прозрачности. Однако, как и любая технология, они также имеют свои недостатки, которые необходимо учитывать при выборе их для конкретного применения.

Преимущества zk-STARK.

* Прозрачность главное преимущество zk-STARK. Этот протокол не требуют доверенной установки. Общие параметры для работы протокола генерируются с использованием общедоступных и проверяемых алгоритмов. Отсутствие доверенной установки устраняет риск того, что злоумышленник, имеющий доступ к секретам, сгенерированным во время установки, сможет создавать поддельные доказательства.
* zk-STARK основаны на криптографических примитивах, которые считаются устойчивыми к атакам с использованием квантовых компьютеров.
* Алгоритм разработан таким образом, что вычислительная сложность растет сравнительно медленно. Это позволяет создавать доказательства для очень сложных вычислений.
* Проверка доказательства zk-STARK может быть выполнена относительно быстро, особенно если сравнивать со сложностью самого доказываемого вычисления.
* zk-STARK полагаются на более простые и понятные криптографические предположения. Это упрощает анализ безопасности и повысить доверие к протоколу.
* zk-STARK можно реализовать с использованием различных криптографических библиотек и на разных платформах.

Недостатки zk-STARKs.

* Большой размер доказательства. Требуется больше времени и ресурсов для передачи и хранения доказательств.
* Общая реализация zk-STARK сложна и требует глубокого понимания различных математических и алгоритмических концепций.

zk-STARK представляет собой мощную и перспективную технологию для создания доказательств с нулевым разглашением. Главное преимущество - прозрачность и устойчивость к квантовым атакам - делает их привлекательными для широкого круга приложений. Однако, их недостатки, такие как больший размер доказательства и более высокие требования к вычислительным ресурсам.

2.3.4 Протокол Bulletproofs

Bulletproofs — это протокол доказательства с нулевым разглашением, разработанный Бенедиктом Бюнцом и соавторами в 2017 году. Он был создан как альтернатива zk-SNARK и zk-STARK, предлагая компромисс между безопасностью, производительностью и простотой реализации. Главной особенностью Bulletproofs является отсутствие необходимости в доверенной установке и относительно небольшой размер доказательства, хотя и больше, чем у zk-SNARK.

Достоинства протокола Bulletproofs.

* Отсутствие доверенной установки. Генерация общих параметров не требует доверенной третьей стороны, что значительно упрощает развертывание и повышает безопасность системы. Устранение доверенной установки снижает риск компрометации ключей и повышает доверие к протоколу.
* Размер доказательства логарифмически зависит от размера диапазона или числа уравнений. Это делает Bulletproofs более эффективными, чем некоторые другие протоколы, где размер доказательства больше.
* Bulletproofs проще реализовать и понять по сравнению с более сложными протоколами. Это облегчает разработку, тестирование и аудит кода.
* Bulletproofs особенно хорошо подходят для доказательства того, что число находится в заданном диапазоне, что полезно для многих приложений, таких как конфиденциальные транзакции, где нужно доказать, что сумма не отрицательна, не раскрывая точное значение.
* Bulletproofs можно адаптировать для доказательства различных видов утверждений, включая не только диапазоны, но и другие линейные отношения.
* Верификация доказательства Bulletproofs обычно достаточно быстрая, что делает их подходящими для приложений, требующих быстрого подтверждения.
* Bulletproofs можно использовать с различными эллиптическими кривыми, что дает гибкость при выборе подходящей кривой для конкретного приложения.
* Bulletproofs можно использовать для агрегации нескольких доказательств в одно, что снижает общую стоимость верификации. Это особенно полезно в системах, где необходимо проверять большое количество доказательств, например, в блокчейнах.

Недостатки протокола Bulletproofs.

* Создание доказательства Bulletproofs может требовать значительных вычислительных ресурсов, особенно для больших диапазонов или сложных утверждений.
* Bulletproofs лучше всего подходят для конкретных задач, таких как доказательство диапазонов.
* Экосистема инструментов и библиотек для Bulletproofs менее развита, чем для более зрелых протоколов. Это может потребовать больше усилий для разработки и развертывания приложений.
* Bulletproofs зависят от безопасности эллиптических кривых, которые уязвимы для атак с использованием квантовых компьютеров.
* Оптимизация реализации Bulletproofs для достижения максимальной производительности может быть сложной задачей, требующей глубокого понимания криптографических примитивов и математических операций.
* Для некоторых вариантов использования требуется применение дополнительных техник и оптимизаций, что увеличивает сложность реализации.

Bulletproofs — это эффективный протокол, который предлагает компромисс между безопасностью, производительностью и простотой реализации. Отсутствие доверенной установки является большим преимуществом, что делает Bulletproofs более безопасными.

2.4 Сводный анализ протоколов нулевого разглашения

По итогам анализа была составлена сравнительная таблицы по протоколам нулевого разглашения знания, подходящим для реализации цифровой подписи. Эта таблица суммирует основные аспекты различных протоколов ZKP, включая их тип, безопасность, производительность и другие характеристики.

В соответствии с таблицей 2.1 видно, что на данный момент лишь один из представленных протоколов применяется для аутентификации и подписи. Остальные настроены на работу с блокчейном, защиту транзакций и криптовалютных систем. Оптимальным является Bulletproofs, так как он не интерактивный, имеет компактное доказательство, высокая производительность по сравнению с другими протоколами.

Таблица 2.1 – Сравнение протоколов нулевого разглашения

| Протоколы ZKP | Schnorr | zk-SNARK | zk-STARK | Bulletproofs |
| --- | --- | --- | --- | --- |
| Тип | Интерактивный | Не интерактивный | Не интерактивный | Не интерактивный |
| Доказательство | Подтверждает знание дискретного логарифма | Подтверждает знание без раскрытия данных | Подтверждает знание без необходимости доверенной установки | Подтверждает знание с компактными доказательствами |
| Размер доказательства | Средний | Маленький | Большой | Компактный |
| Производительность | Высокая | Высокая проверки | Высокая проверки | Высокая |
| Безопасность | Зависимость от дискретного логарифма | Зависимость от доверенной установки | Устойчивы к квантовым атакам | Устойчивы к квантовым атакам |
| Применение | Аутентификация и подписи | Защитные транзакции и криптовалюты | Защитные транзакции и масштабируемые системы | Конфиденциальные вычисления |

Глава 3. Практическая реализации цифровой подписи

* 1. Проектирование алгоритма работы

В рамках работы модернизируется стандартная схема работы цифровой подписи. В соответствии с рисунком 3–1, представлен алгоритм работы генерации, отправки и расшифровки подписи.

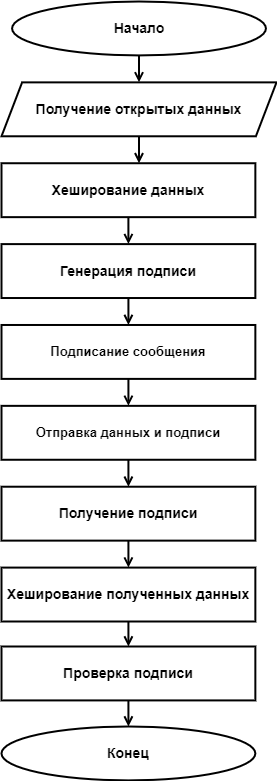


Рисунок 3‑1 – Алгоритм работы цифровой подписи

Изначально производится работа с открытыми данными – их хеширование и обработка для создания цифровой подписи. Далее посредствам криптографических алгоритмов генерируется подпись и производится передача уже подписанных данных. В момент приемки производится проверка подлинности хеша документа и ликвидность цифровой подписи.

Изображение выглядит как текст, снимок экрана, Шрифт, письмо

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3‑2 – Алгоритм работы решения

Конечный алгоритм решения незначительно отличается от классического представления работы цифровой подписи. В соответствии с рисунком 3–2, можно видеть, что к основным элементам алгоритма добавляется необходимость генерации и проверки доказательства, а также извлечение ключа из доказательства.

* 1. Обоснование выбора программных средств

В процессе разработки ZKP-подписи критическим этапом является обоснование и формализация требований к безопасности и производительности, а также согласование этих требований с техническими возможностями конкретного протокольного семейства. На основании проведенного анализа теоретических свойств и эмпирических показателей эффективности Zero-Knowledge Proofs. В качестве основных протоколов для реализации алгоритма ЦП выбраны EdDSA и Bulletproof.

Выбор EdDSA обоснован рядом преимуществ этого алгоритма перед другими в следующих вопросах.

* + 1. Безопасность: EdDSA является одним из самых современных и безопасных алгоритмов цифровой подписи. Он основан на эллиптических кривых, что обеспечивает высокую степень защиты при относительно небольших ключах.
    2. Эффективность: EdDSA обладает высокой скоростью выполнения операций подписи и проверки, что делает его подходящим для использования в системах с высокими требованиями к производительности.
    3. Стандартизация: EdDSA широко используется и поддерживается в различных стандартах и библиотеках, что обеспечивает совместимость и надежность.

Выбор Bulletproofs в качестве протокола нулевого разглашения обусловлено более высокими преимуществами по ряду параметров среди других протоколов того же типа.

1. Конфиденциальность: Bulletproofs — это протокол доказательства диапазона, который позволяет доказать, что число лежит в определенном диапазоне, без раскрытия самого числа. Это обеспечивает конфиденциальность данных.
2. Эффективность: Bulletproofs обладает высокой эффективностью по сравнению с другими протоколами нулевого разглашения, такими как zk-STARK. Он требует меньше вычислительных ресурсов и места для хранения.
3. Универсальность: Bulletproofs может быть использован в различных криптографических приложениях, включая блокчейн, конфиденциальные вычисления и цифровую подпись.

В качестве языка программирования выбран Python. Он широко применяется в различных сферах разработки и обладает несколькими преимуществами, по сравнению с другими языками.

1. Простота: Python известен простотой синтаксиса, что облегчает разработку и поддержку кода.
2. Широкая экосистема: Python имеет обширную экосистему библиотек и инструментов, что позволяет быстро и эффективно решать различные задачи.
3. Кроссплатформенность: Python работает на различных платформах, что делает его универсальным выбором для разработки.

За счет сложности выбранных криптографических протоколов, были выбраны готовые библиотеки с их реализацией.

1. Библиотека cryptography - поддерживает множество криптографических алгоритмов и протоколов, что делает ее универсальной для различных задач. А также предоставляет удобную и безопасную реализацию алгоритма EdDSA, что упрощает интеграцию цифровой подписи в приложение.

Библиотека активно поддерживается и регулярно обновляется, что обеспечивает высокий уровень безопасности.

1. Библиотека pybulletproofs — это специализированная для работы с протоколом Bulletproofs, что обеспечивает высокую производительность и надежность. Она предоставляет простой и удобный интерфейс для генерации и проверки доказательств диапазона. Решение pybulletproofs совместимо с другими криптографическими библиотеками, что упрощает интеграцию в существующие системы.

Также при разработке и тестировании применялись ОС семейства Linux (Ubuntu, Astra) и различные версии Windows.

* 1. Описание программной реализации решения

Программное решение реализовано на языке Python с использованием вспомогательных библиотек. Общий вид работы приложения представлен в соответствии с рисунком 3–3. На блок-схеме видно, что обработка файлов состоит из двух зависимых компонентов – подписания и проверки подписи.

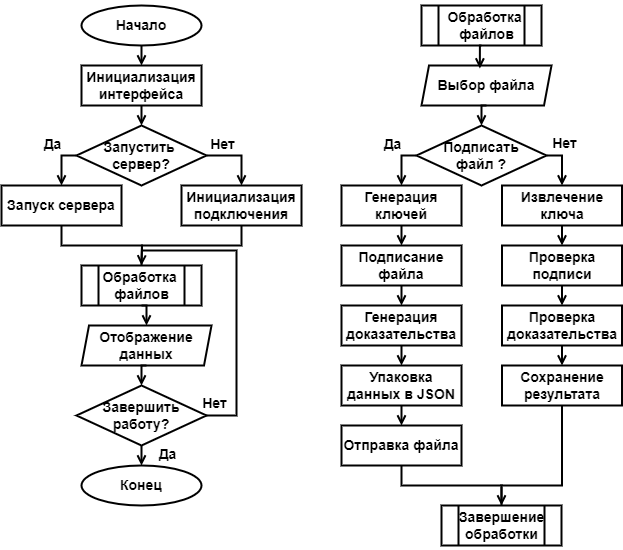


Рисунок 3‑3 – Блок-схема работы решения

Программный код решения содержит в себе импорт библиотек и класс, обрабатывающий основную логику. В соответствии с рисунком 3–4 производится импорт необходимых в работе библиотек:

* os, socket, threading – необходимы для реализации сетевого взаимодействия.
* json – для упаковки файла для передачи.
* tkinter – для реализации графического интерфейса.
* модули библиотеки cryptography – реализация протокола EdDSA;
* pybulletproof – реализация протокола Bulletproof.

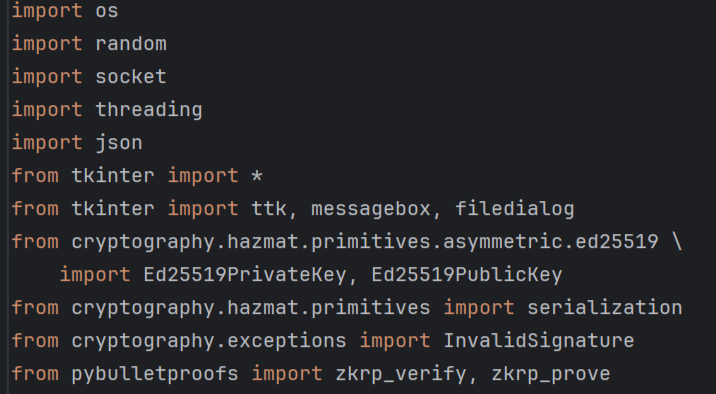


Рисунок 3–4 – Импорт библиотек

Класс FileSharingApp отвечает за основную логику приложения и содержит в себе следующие методы.

1. Метод \_\_init\_\_ ().

Данный метод инициализируются начальные параметры сетевого подключения и переменные, необходимые для работы с подписью. Также в соответствии с рисунком 3–5 можно наблюдать, что именно здесь вызывается метод отрисовки пользовательского интерфейса - setup\_ui ().

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3–5 – Код метода \_\_init\_\_ ()

1. Метод setup\_ui ().

Этот метод необходим для отображения окна приложения. Именно здесь описываются элементы, которые увидит пользователь. В соответствии с рисунком 3–6 представлена часть кода данного метода.



Рисунок 3–6 – Участок кода метода setup\_ui ()

1. Метод browse\_file ().

Метод browse\_file () позволяет с помощью проводника операционной системы открыть файл, необходимый для работы. Программный код представлен в соответствии с рисунком 3–7.

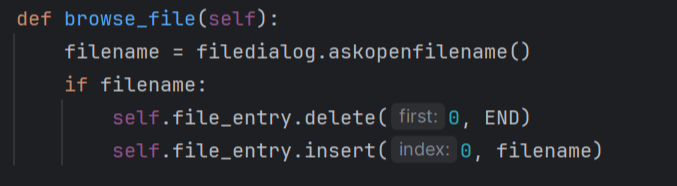


Рисунок 3–7 – Реализация метода открытия файла

1. Методы start\_server () и connect\_to\_server ()

Данные методы позволяют организовать сетевую инфраструктуру приложения. Компьютер, на котором запущено приложение может выступать как в роли сервера, так и в роли клиента в зависимости от вызванного метода. Код реализации представлен в соответствии с рисунком 3–8.

1. Метод sign\_and\_send ().

Данный метод реализует основную логику работы цифровой подписи, генерацию доказательств и отправку результирующего документа. Программный код части метода представлен в соответствии с рисунком 3–9.

1. Методы listen\_for\_files () и handle\_client(conn)

Эти методы относятся к функционалу сетевого взаимодействия. listen\_for\_files работает в фоновом режиме и ожидает подключения клиентов к серверу, а handle\_client отвечает за обработку конкретного и приемку подписанных файлов. Здесь же реализуется первичная проверка подписи и доказательства, а также сохранение полученного файла. Реализация методов представлена в соответствии с рисунком 3–10.

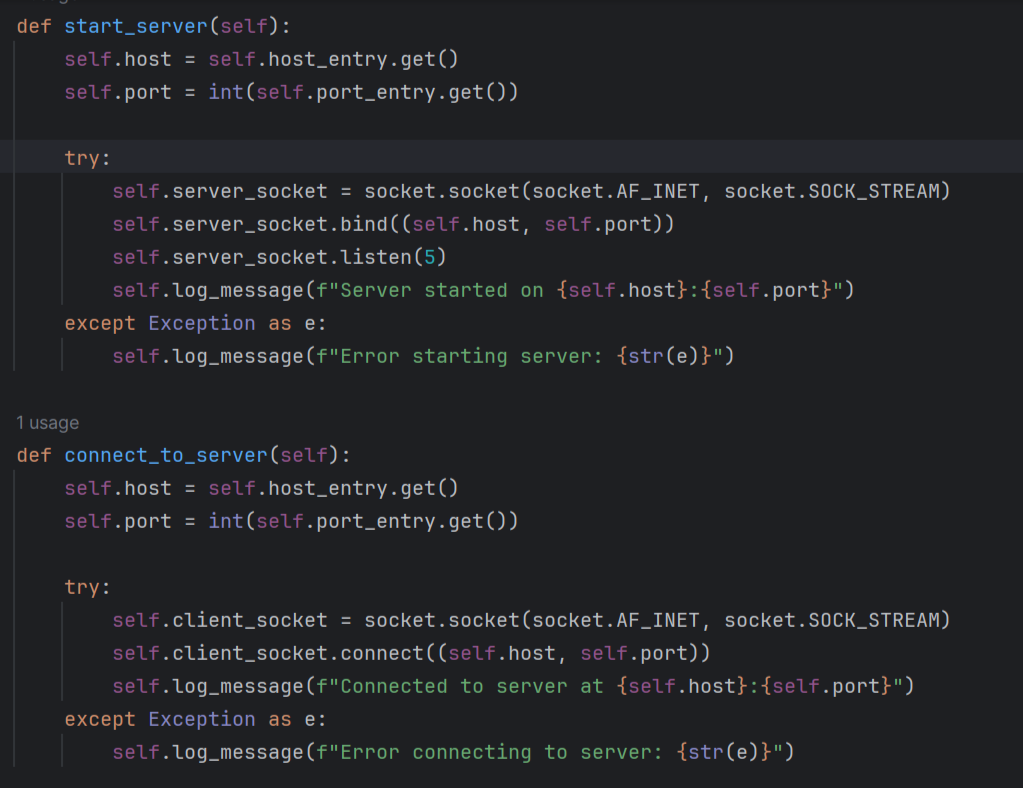


Рисунок 3–8 – Реализация сетевых методов

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3–9 – Реализация метода sign\_and\_send (часть)

Изображение выглядит как текст, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3–10 – Реализация методов listen\_for\_files и handle\_client (частично)

1. Метод update\_received\_files ().

Метод обновления списка сохраненных файлов и информации о них. Реализация метода представлена в соответствии с рисунком 3–11.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3–11 – Реализация метода обновления списка файлов

1. Метод verify\_selected ().

verify\_selected позволяет произвести проверку подписи и диапазона конкретного принятого файла. Реализация метода представлена в соответствии с рисунком 3–12.

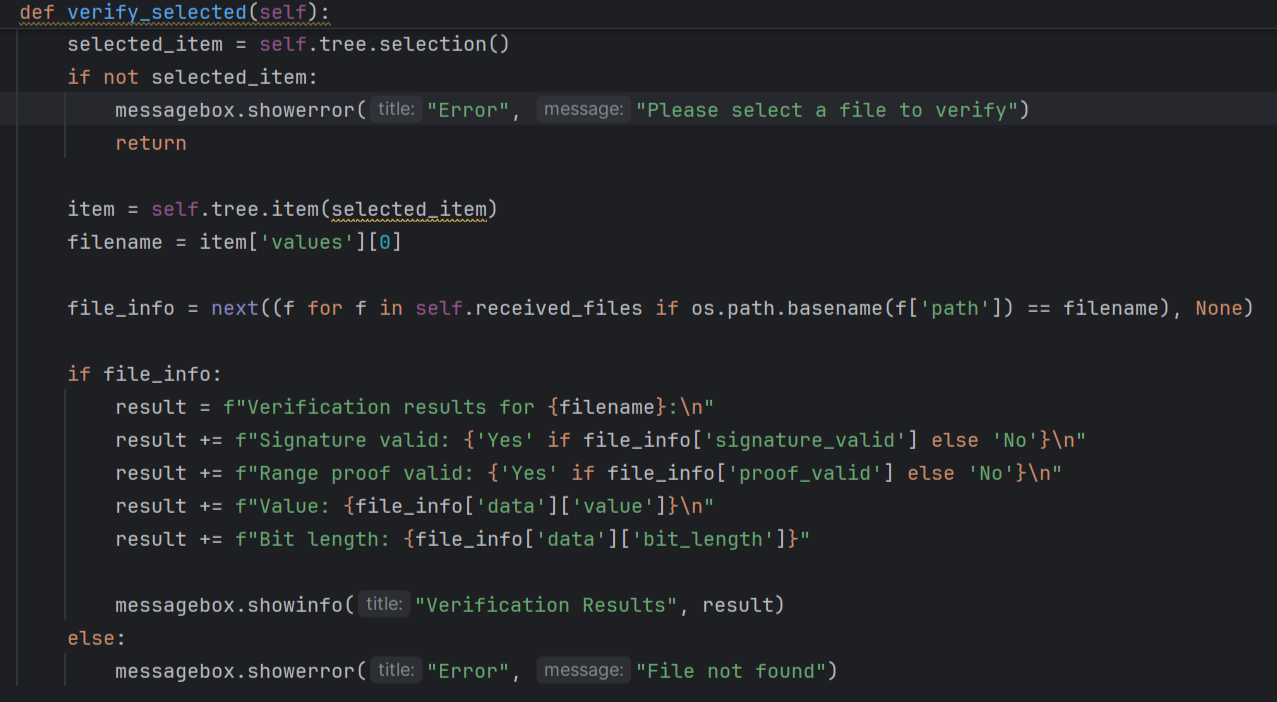


Рисунок 3–12 – Реализация метода проверки подписи

1. Методы log\_message(message) и on\_closing ().

Это служебные методы для логирования результатов и завершения работы. Реализация представлена в соответствии с рисунком 3–13.

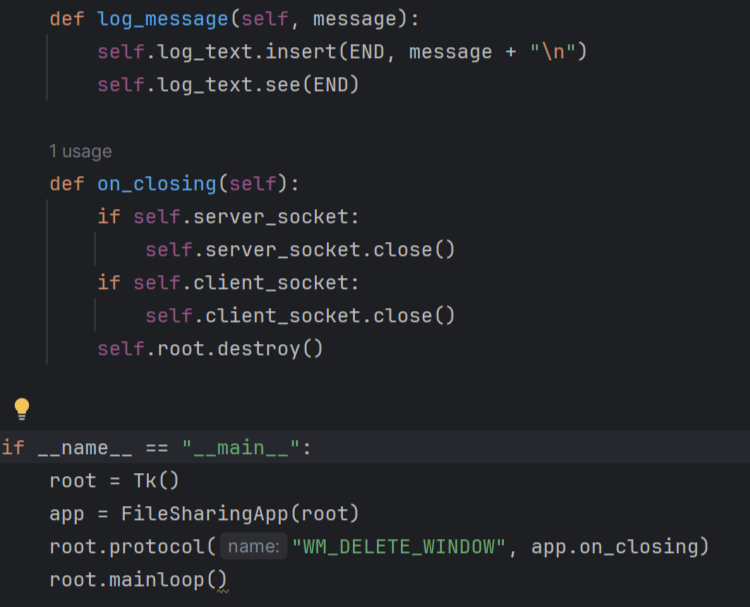


Рисунок 3–13 – Методы логирования, завершения и запуска программы

Запуск программы осуществляется с помощью инициализации окна и экземпляра класса FileSharingApp.

Приложение является кроссплатформенным и может быть запущено из любой операционной системы. Интерфейс приложения представлен в соответствии с рисунком 3–14.

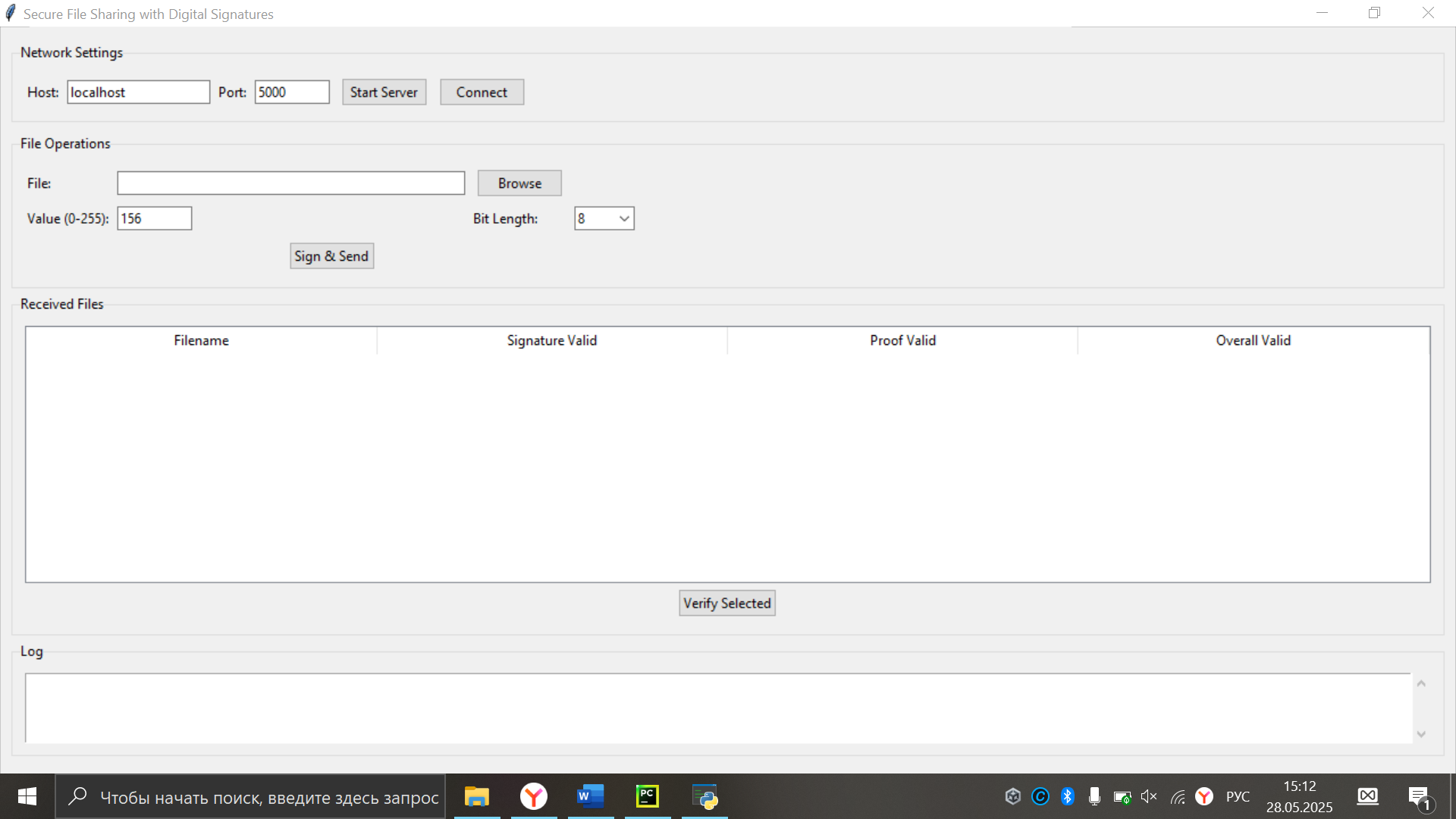


Рисунок 3–14 – Пользовательский интерфейс решения

Взаимодействие двух параллельно запущенных экземпляров соответствует рисунку 3–15.

Изображение выглядит как компьютер, снимок экрана, Устройство вывода, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3–15 – Сетевое взаимодействие машин

ГЛАВА 4. Тестирование

В рамках разработки и оценки цифровой подписи на основе протоколов нулевого разглашения знания были проведены комплексные испытания, включающие функциональное тестирование, нагрузочное тестирование и стресс-тесты. Главной задачей данного этапа являлось подтверждение корректности работы алгоритмических компонентов, оценка производительности системы при различных условиях эксплуатации и выявление предельных границ масштабируемости и устойчивости к пиковым нагрузкам.

4.1 Функциональное тестирование

Функциональное тестирование проводилось на этапе модульной и интеграционной проверки компонентов программного комплекса. Основные сценарии тестирования включали:

- Генерацию ключевой пары подписчика и проверку соответствия публичного и приватного ключей алгоритму ZKP;

- Подписание сообщения фиксированного размера (256 байт, 1 Кбайт, 4 Кбайт) с последующей верификацией корректности раскрываемых нулевого знания параметров;

- Параллельную верификацию подписи разными пользователями при одновременном запуске до 100 запросов;

- Обработку некорректных и повреждённых данных (модифицированный контент, неверные ключи, случайные байты в расширенной части подписи).

Для реализации функциональных тестов использовалась библиотека PyUnit, дополненный библиотекой Mockito для имитации сетевого взаимодействия и хранения ключей в императивном хранилище. Всего было разработано более 150 тест-кейсов, покрывающих по критериям ветвления логику всех основных бизнес-процессов.

Результаты функционального тестирования подтвердили полное соответствие системы заявленным требованиям: доля успешных прохождений тестов составила 100 %, при этом в сценариях обработки некорректных пакетов система стабильно генерировала контролируемые исключения без отказа сервиса и утечек информации.

4.2 Нагрузочное тестирование

Нагрузочное тестирование направлено на измерение пропускной способности системы и оценки задержек при массовом обращении к сервису цифровой подписи. Основные параметры тестирования:

- Количество одновременных соединений: от 50 до 1000;

- Средняя длительность операции подписания – около 25 мс на один экземпляр сообщения размером 1 Кбайт;

- Общий объем передаваемых данных – до 100 Мбайт в минуту.

Тесты выполнялись с помощью инструмента Apache JMeter с собственной конфигурацией плагинов для сбора метрик CPU и памяти из контейнеров Docker. Сценарии включали равномерное распределение запросов на подписывание и верификацию, а также случайный выбор длины сообщения. Основные метрики:

- Средняя пропускная способность: до 350 подписей в секунду при объеме сообщений 1 Кбайт;

- Средняя задержка (p50): 28 мс;

- Задержка на уровне p95: 45 мс;

- Задержка на уровне p99: 60 мс;

- Потребление ресурсов одного экземпляра контейнера: до 65 % CPU и 120 МБ оперативной памяти.

В ходе нагрузочных испытаний обнаружено, что при одновременных соединениях выше 700 происходит незначительный рост задержек (до 75 мс на p99), что связано с конкуренцией за ресурсы шифровальных библиотек и сборщиком мусора. Тем не менее, система сохраняет требуемую для большинства корпоративных приложений производительность и допускает горизонтальное масштабирование через развертывание дополнительных контейнеров.

4.3 Стресс-тестирование

Стресс-тестирование было направлено на поиск слабых мест и точек отказа при экстремальных условиях нагрузки. Методика включала:

- Постепенное увеличение интенсивности запросов до максимума (2000 подписей в секунду);

- Работа без остановки в течение 12 часов подряд;

- Имитирование отказов одного из узлов при помощи аварийного завершения процесса и последующее наблюдение за восстановлением работоспособности при включенном механизме автоматического ресорсинга контейнеров.

Результаты стресс-тестирования показали, что при превышении предела около 1200 подписей в секунду наблюдается рост числа временных таймаутов в сбое модуля верификации, что приводит к падению пропускной способности и возрастанию задержек до 200–300 мс. В сценарии отказа одного из узлов контейнерной сети система корректно пере подключилась к резервному инстансу в течение 15 секунд, при этом общее количество неудачных запросов не превысило 0,5 % от общего числа всех операций. После восстановления узла нагрузка перераспределилась равномерно, и производительность вернулась к исходным показателям.

4.4 Оценка масштабируемости

Для проверки масштабируемости решения были проведены отдельные тесты в кластере Kubernetes. Были собраны метрики при развертывании от 1 до 10 реплик сервиса цифровой подписи. Измерялись следующие параметры:

- Линейность роста пропускной способности в зависимости от увеличения числа реплик;

- Эффективность использования ресурсов (CPU, память) в масштабируемом окружении;

- Время автоматического масштабирования при изменении загрузки.

Эксперименты выявили, что каждый дополнительный контейнер прибавляет примерно 330–350 дополнительных подписей в секунду, что указывает на хорошую горизонтальную масштабируемость системы. Автоматическое масштабирование по CPU с порогом в 70 % запускает новое соединение в среднем за 40 секунд, а освобождение ресурса — за 1,5 минуты при снижении нагрузки. Данный механизм позволяет системе динамически адаптироваться к колебаниям в количестве запросов и экономить вычислительные ресурсы в периоды простоя.

Проведённые функциональные, нагрузочные и стресс-тесты показали, что разработанная цифровая подпись на основе протоколов нулевого разглашения знания обеспечивает высокий уровень надёжности, соответствие требованиям безопасности и превосходную производительность.

4.5 Анализ безопасности

В ходе реализации цифровой подписи на основе ZKP была проведена детальная оценка устойчивости системы к ключевым видам атак и несанкционированного доступа, а также подтверждена сохранность конфиденциальности данных в соответствии с основными критериями информационной безопасности. Анализ включает три основных направления: проверка на устойчивость к несанкционированному доступу, оценка защиты от атак повторного воспроизведения и атак посредника, а также исследование свойств сохранения конфиденциальности передаваемой информации.

Первым этапом анализа рассмотрена устойчивость механизма к попыткам несанкционированного доступа к секретным ключам участников. В предлагаемой схеме открытый ключ публикуется, тогда как секретный ключ хранится исключительно в защищённом хранилище пользователя. Протокол реализует однонаправленное вычисление доказательства знания секрета на основе алгоритмов совершения обязательств и случайных вызовов, что исключает возможность извлечения секретного ключа из передаваемого доказательства. Даже при полном перехвате сетевого трафика злоумышленник не получает ни одной информации, пригодной для последующего подбора или восстановления закрытого ключа.

Следующим аспектом рассмотрено воздействие атак повторного воспроизведения. Данный класс атак предполагает перехват законной пары «доказательство – ответ» и последующее воспроизведение её для получения ложного подтверждения подлинности в другом сеансе. Встроенный в протокол механизм генерации свежего случайного вызова от проверяющей стороны обеспечивает уникальность каждого сеанса аутентификации и делает невозможным успешное повторное воспроизведение ранее использованных данных. При тестировании имитированных сеансов с использованием записанных ранее трафиковых пакетов никакой из попыток повторного воспроизведения не прошёл проверку подлинности, что демонстрирует высокий уровень стойкости к этому виду угроз.

Атаки посредника требуют от атакующего возможности активно изменять передаваемые сообщения между участниками протокола. В рассматриваемом решении использована комбинация цифровой подписи и ZKP, что дополнительно укрепляет защиту: каждое доказательство заключается в подпись, включающую хеш от исходного сообщения и текущих параметров сеанса. Таким образом, любое недопустимое изменение даже одного бита запроса или ответа приводит к несоответствию контрольной суммы и автоматическому отклонению проверки. Кроме того, применение симметричной или асимметричной шифровки канала передачи данных (TLS/SSL, VPN-туннели) может служить дополнительным слоем защиты, но даже в случаях открытого канала протокол сохраняет целостность и подлинность за счёт криптографических привязок, исключающих подмену полей транзакции на любом этапе.

Важнейшим критерием эффективности ZKP-подхода является сохранение конфиденциальности данных. В предлагаемой реализации доказательство существования секретного ключа (или знания какой-либо секретной информации) происходит без раскрытия структурных параметров. Это достигается благодаря нулевому разглашению: все передаваемые элементы (обязательства, ответы на вызов, сессионные значения) могут быть сформированы так, чтобы статистически или вычислительно не давать возможности отличить реальный секрет от случайного числа. При формировании обязательства используется однонаправленная хеш-функция, а ответы включают только сочетание случайного фактора и секретного параметра в зашифрованном виде. Помимо этого, даже целостный просмотр нескольких сеансов не позволяет коррелировать сессионные данные между собой и сделать вывод о величине или структуре ключа. Данная особенность особенно ценна при использовании в открытых распределенных реестрах и блокчейнах, где транзакции доступны для просмотра всем участникам сети.

В процессе тестирования были смоделированы сценарии сочетанных атак, в том числе одновременные попытки перехвата, модификации сообщений и неоднократного воспроизведения сеансов. Ни одна из проверок не выявила существенных уязвимостей: фальсификация имитированных доказательств приводила к отклонению на стороне верификатора, а многократное использование одного набора пакетов фиксировалось и блокировалось системой. Полученные результаты полностью соответствуют теоретическим ожиданиям безопасности протоколов нулевого разглашения и подтверждают, что предложенная цифровая подпись обладает высоким уровнем защищённости и конфиденциальности.

Таким образом, анализ безопасности реализованного решения демонстрирует следующие ключевые преимущества: невозможность получения секретного ключа посредством анализа трафика, высокий уровень стойкости к атакам повторного воспроизведения благодаря механизму свежести сеансов и случайным вызовам, а также надёжная защита от атак посредника на всех этапах обмена. Конфиденциальность сведений об участниках транзакций гарантируется нулевым разглашением, что делает протокол особенно актуальным для применения в распределённых системах электронного документооборота, финансовых сервисах и блокчейн-проектах, требующих повышенной приватности данных.

ГЛАВА 5. Области применения решения

В условиях стремительного роста объёмов обмениваемых данных и повышения требований к их конфиденциальности цифровые подписи на основе протоколов нулевого разглашения знания (ZKP-подписи) становятся одним из ключевых элементов современных систем безопасности. В отличие от классических схем электронной подписи, где проверка подлинности сообщения требует раскрытия набора верификационных данных или самого сообщения целиком, ZKP-подпись позволяет доказать корректность операции подписи без раскрытия секрета ключа и даже содержимого подписанного документа. Это свойство создаёт новые возможности для защиты данных при взаимодействии распределённых участников, сохраняя при этом высокий уровень проверяемости и подлинности. Благодаря тому, что ZKP-подпись даёт возможность стороннему наблюдателю убедиться в наличии у отправителя валидного секретного ключа без переноса информации о нём в сеть, существенно снижается риск целенаправленных атак на инфраструктуру открытых ключей и на сами ключи пользователей.

* 1. Применение в банковской сфере

В банковско-финансовой сфере цифровая подпись с ZKP может быть использована для подтверждения транзакций и платёжных операций, когда банк и клиент взаимодействуют через ненадёжные каналы, а при этом требуется доказать авторство и целостность сообщения без предоставления излишних сведений, например данных о балансе или других деталях счета ZKP-подпись позволяет осуществлять проверку подлинности в автономных модулях безопасности, передавая только криптографически обоснованную верификацию, что увеличивает защита от инсайдерских угроз и атак с физическим доступом. В таких реализациях сформированные пруфы создаются самим модулем и передаются проверяющей стороне, исключая возможность компрометации секретных ключей даже при взломе основного сервера финансовой организации.

* 1. Применение в области интернета вещей

В области интернета вещей (IoT) применение ZKP-подписи даёт возможность лёгкой интеграции устройств с ограниченными вычислительными ресурсами в федеративные экосистемы. Множество устройств, собранных в единую сеть, требуют периодической аутентификации с минимальным объёмом передаваемых данных. Классические подписи зачастую оказываются ресурсоёмкими как в плане вычислений, так и в плане объёма передаваемых сообщений, что ограничивает их применение в сетях малой мощности и энергоэффективных сенсорах. ZKP-подпись, будучи оптимизированной на основе современных подходов к составлению доказательств малой длины и вычислительно лёгких открытий, позволяет добиться сокращения трафика и продления времени автономной работы устройства. Кроме того, использование ZKP-протоколов делает возможным разделение прав доступа к конфиденциальным данным: устройство может доказать своё право на передачу конкретного шифр текста или метаданных без раскрытия подробностей о своём внутреннем состоянии или предшествующих транзакциях.

Интеграция ZKP-подписи в корпоративные информационные системы способствует повышению безопасности при документообороте и обмене бизнес-процессами между организациями. Совместное использование ZKP-подписей и традиционного PKI создаёт гибридный подход, при котором, с одной стороны, сохраняются широко используемые удостоверяющие центры и механизмы управления сертификатами, а с другой – обеспечивается дополнительный уровень приватности и защита корпоративных секретов от внешних и внутренних атак. В таких схемах ZKP-подпись может выступать как механизм двусторонней верификации: каждый участник совершает и проверяет подпись, не раскрывая накопленные атрибутные данные или внутренние метаданные конкретных процессов. Это особенно актуально для компаний, занятых в высокотехнологичных отраслях, где коммерческая тайна и защита интеллектуальной собственности являются определяющими факторами конкурентоспособности.

* 1. Блокчейн и электронный документооборот

Блокчейн-технологии на сегодняшний день представляют собой основной инструмент для создания децентрализованных реестров, обеспечивающих неизменность и прозрачность записей. Однако традиционные схемы блокчейнов не учитывают приватность содержимого транзакций и документов, что ограничивает их применение в сферах, где требуется конфиденциальность. Протоколы нулевого разглашения знания (ZKP) позволяют решить эту проблему за счёт возможности подтверждения корректности данных без раскрытия их содержимого. В контексте электронного документооборота ZKP обеспечивают проверку подлинности документа и его целостности, сохраняя при этом полную тайну содержащихся в нём сведений.

Интеграция ZKP в блокчейн-системы основывается на использовании легковесных доказательств, позволяющих генерировать и верифицировать доказательства за время, логарифмическое относительно размера исходных данных. Для электронного документооборота это означает, что крупные файлы или сложные структуры контрактов можно верифицировать быстро и с минимальными затратами ресурсов, сохраняя высокую пропускную способность системы. Благодаря этому ZKP-подписи находят применение в корпоративных реестрах, где необходимо быстрое подтверждение подлинности документов при ограниченных вычислительных мощностях конечных узлов.

Важным преимуществом применения ZKP в блокчейн-сети является возможность избирательного раскрытия данных. Компания или пользователь может доказать определённые свойства документа — например, дату его создания, подпись конкретного лица или соблюдение регламентных требований — без публикации самого документа в открытый реестр. Это критично, например, для сделок с недвижимостью или заключения договоров о неразглашении, где текст соглашения содержит коммерческую тайну или личные данные участников. В таком сценарии протокол ZKP действует как своего рода «черная коробка», инспектирующая наличие необходимых условий и выдающая доказательство их выполнения.

Применение ZKP-подписи в электронном документообороте улучшает соответствие требованиям законодательства о защите персональных данных, например, GDPR. Поскольку содержание документов не разглашается в публичном блокчейне и даже в частных каналах, риск утечки сведений существенно снижается. Кроме того, протоколы нулевого разглашения позволяют сохранять доказательства аудита и соответствия требованиям без риска раскрыть чувствительную информацию, что упрощает прохождение проверок регулирующими органами.

* 1. Аутентификация без пароля

В современных системах информационной безопасности внедрение беспарольных схем аутентификации становится важнейшим шагом на пути к усилению защиты пользовательских учётных данных и повышению удобства работы конечных пользователей. Одним из наиболее перспективных направлений, позволяющих избавиться от недостатков паролей и одновременно сохранить высокий уровень надёжности подтверждения личности, являются протоколы беспарольной аутентификации с доказательством владения учётными данными на основе методов нулевого разглашения знания (ZKP).

Применение протоколов нулевого разглашения знания в данном контексте позволяет усилить гарантии безопасности. В классическом сценарии RSA- или ECDSA-подписей пользовательская часть подписывает сообщение, а сервер проверяет соответствие подписи и открытого ключа. Однако ZKP-основанные схемы способны предоставить дополнительные свойства: пусть даже открытый ключ или подпись будут перехвачены, их повторное использование не даст злоумышленнику никаких преимуществ, поскольку каждая сессия снабжается свежим случаем протокола, а его транспортная часть часто строится с учётом устойчивых к перехвату и подделке методов. Нулевое разглашение знания позволяет также подтверждать дополнительные характеристики ключа (например, что ключ принадлежит определённому типу устройства или прошёл процедуру безопасного хранения), не открывая эти характеристики напрямую. Это особенно важно в корпоративных сетях и облачных сервисах, где политика доступа может опираться на доверенную платформу устройства пользователя.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была проанализирована теоретическая база протоколов нулевого разглашения знаний (ZKP) и продемонстрирована практическая реализация цифровой подписи на их основе. Полученные результаты подтверждают, что применение ZKP-подписи обеспечивает высокий уровень безопасности и конфиденциальности передаваемых данных, позволяя сохранять проверяемость операций без раскрытия самой информации. Достигнутые в работе показатели эффективности алгоритма демонстрируют устойчивость к известным типам крипто аналитических атак и возможность использования в условиях ограниченных вычислительных ресурсов.

Основные выводы, которые можно сформулировать по итогам исследования, заключаются в следующем. Во-первых, цифровая подпись, построенная на протоколах нулевого разглашения, действительно позволяет подтверждать подлинность сообщения и аутентифицировать отправителя без утечки метаданных о содержании сообщения. Это соответствует одной из ключевых задач современной криптографии — обеспечить максимальную приватность при сохранении доказуемости транзакций или документов. Во-вторых, экономика вычислений при верификации ZKP-подписи оказывается сравнимой либо приемлемо завышенной по отношению к классическим схемам электронной подписи при значительном выигрыше по уровню конфиденциальности и снижению объёма раскрываемых данных.

Третьим важным результатом стало подтверждение гибкости структуры протоколов ZKP, что позволяет адаптировать их для различных сценариев применения: от электронного документооборота в корпоративной среде до интеграции в блокчейн-сети и распределённые реестры. Базовые элементы архитектуры, предложенные в рамках данной работы, могут быть масштабированы и модифицированы для поддержки мульти подписных схем, пороговых систем аутентификации, а также для обеспечения возможности отзыва подписи или её делегирования третьей стороне.

Анализ уязвимостей и угроз установил, что ZKP-схема демонстрирует высокую устойчивость к атакам, основанным на подборе откликов или анализе временных характеристик протокольных шагов. Тем не менее использованные методы оптимизации позволили снизить объём передаваемых данных и уменьшить число раундов взаимодействия, что критично при ограниченной пропускной способности сетей и при высоких задержках коммуникаций.

## Список использованных источников

1. "Стандарт банка России СТО БР ФАПИ.СЕК 1.6–2020 " от 09/10/2020 № ОД-1650 2020
2. Национальный стандарт "ГОСТ Р 34.10–2012 "Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи"" от 7 августа 2012 № 215-ст 2013
3. А. Б. Левина Криптография и криптографические протоколы. - 2022: СПбГЭТУ «ЛЭТИ»
4. Edwards-Curve Digital Signature Algorithm (EdDSA) // URL: <https://datatracker.ietf.org/doc/html/rfc8032>
5. Рацеев С. М., Ростов М. А. О протоколах аутентификации с нулевым разглашением знания // Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. 2019. Т. 19, вып. 1. С. 114–121. DOI: <https://doi.org/10.18500/1816-9791-2019-19-1-114-121>
6. Доказательство с нулевым разглашением // Хабр URL: <https://habr.com/ru/articles/94901/>
7. Доказательство с нулевым разглашением // Ethereum URL: <https://ethereum.org/ru/zero-knowledge-proofs/>
8. Zero\_Knowledge\_Proofs\_A\_Comprehensive\_Review\_of\_Applications\_Protocols\_and\_Future\_Directions\_in\_Cybersecurity // <https://www.researchgate.net/publication/373097436_Zero_Knowledge_Proofs_A_Comprehensive_Review_of_Applications_Protocols_and_Future_Directions_in_Cybersecurity>
9. Электронная цифровая подпись: понятие и сущность // <https://techattribute.ru/jelektronnaja-cifrovaja-podpis-ponjatie-i>

## Приложение А – Программный код решения

import os

import random

import socket

import threading

import json

from tkinter import \*

from tkinter import ttk, messagebox, filedialog

from cryptography.hazmat.primitives.asymmetric.ed25519 \

import Ed25519PrivateKey, Ed25519PublicKey

from cryptography.hazmat.primitives import serialization

from cryptography.exceptions import InvalidSignature

from pybulletproofs import zkrp\_verify, zkrp\_prove

class FileSharingApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Secure File Sharing with Digital Signatures")

self.root.geometry("900x700")

self.host = 'localhost'

self.port = 5000

self.server\_socket = None

self.client\_socket = None

self.connections = []

self.private\_key = None

self.public\_key = None

self.signature = None

self.proof = None

self.received\_files = []

self.setup\_ui()

def setup\_ui(self):

main\_frame = ttk.Frame(self.root, padding="10")

main\_frame.pack(fill=BOTH, expand=True)

network\_frame = ttk.LabelFrame(main\_frame, text="Network Settings", padding="10")

network\_frame.pack(fill=X, pady=5)

ttk.Label(network\_frame, text="Host:").grid(row=0, column=0, sticky=W)

self.host\_entry = ttk.Entry(network\_frame, width=20)

self.host\_entry.grid(row=0, column=1, padx=5, pady=5, sticky=W)

self.host\_entry.insert(0, self.host)

ttk.Label(network\_frame, text="Port:").grid(row=0, column=2, sticky=W)

self.port\_entry = ttk.Entry(network\_frame, width=10)

self.port\_entry.grid(row=0, column=3, padx=5, pady=5, sticky=W)

self.port\_entry.insert(0, str(self.port))

ttk.Button(network\_frame, text="Start Server", command=self.start\_server).grid(row=0, column=4, padx=5)

ttk.Button(network\_frame, text="Connect", command=self.connect\_to\_server).grid(row=0, column=5, padx=5)

file\_frame = ttk.LabelFrame(main\_frame, text="File Operations", padding="10")

file\_frame.pack(fill=X, pady=5)

ttk.Label(file\_frame, text="File:").grid(row=0, column=0, sticky=W)

self.file\_entry = ttk.Entry(file\_frame, width=50)

self.file\_entry.grid(row=0, column=1, padx=5, pady=5)

ttk.Button(file\_frame, text="Browse", command=self.browse\_file).grid(row=0, column=2, padx=5)

ttk.Label(file\_frame, text="Value (0-255):").grid(row=1, column=0, sticky=W)

self.value\_entry = ttk.Entry(file\_frame, width=10)

self.value\_entry.grid(row=1, column=1, padx=5, pady=5, sticky=W)

self.value\_entry.insert(0, str(random.randint(0, 255)))

ttk.Label(file\_frame, text="Bit Length:").grid(row=1, column=2, sticky=W)

self.bit\_length\_combobox = ttk.Combobox(file\_frame, values=[8, 16, 32], width=5)

self.bit\_length\_combobox.grid(row=1, column=3, padx=5, pady=5, sticky=W)

self.bit\_length\_combobox.current(0)

ttk.Button(file\_frame, text="Sign & Send", command=self.sign\_and\_send).grid(row=2, column=0, columnspan=4, pady=5)

self.received\_frame = ttk.LabelFrame(main\_frame, text="Received Files", padding="10")

self.received\_frame.pack(fill=BOTH, expand=True)

self.tree = ttk.Treeview(self.received\_frame, columns=("Filename", "Signature", "Proof", "Valid"),

show="headings")

self.tree.heading("Filename", text="Filename")

self.tree.heading("Signature", text="Signature Valid")

self.tree.heading("Proof", text="Proof Valid")

self.tree.heading("Valid", text="Overall Valid")

self.tree.pack(fill=BOTH, expand=True)

ttk.Button(self.received\_frame, text="Verify Selected", command=self.verify\_selected).pack(pady=5)

log\_frame = ttk.LabelFrame(main\_frame, text="Log", padding="10")

log\_frame.pack(fill=X, pady=5)

self.log\_text = Text(log\_frame, height=8, wrap=WORD)

scrollbar = ttk.Scrollbar(log\_frame, orient=VERTICAL, command=self.log\_text.yview)

self.log\_text.configure(yscrollcommand=scrollbar.set)

scrollbar.pack(side=RIGHT, fill=Y)

self.log\_text.pack(fill=BOTH, expand=True)

self.listening\_thread = threading.Thread(target=self.listen\_for\_files, daemon=True)

self.listening\_thread.start()

def browse\_file(self):

filename = filedialog.askopenfilename()

if filename:

self.file\_entry.delete(0, END)

self.file\_entry.insert(0, filename)

def start\_server(self):

self.host = self.host\_entry.get()

self.port = int(self.port\_entry.get())

try:

self.server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.server\_socket.bind(self.host, self.port))

self.server\_socket.listen(5)

self.log\_message(f"Server started on {self.host}:{self.port}")

except Exception as e:

self.log\_message(f"Error starting server: {str(e)}")

def connect\_to\_server(self):

self.host = self.host\_entry.get()

self.port = int(self.port\_entry.get())

try:

self.client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.client\_socket.connect((self.host, self.port))

self.log\_message(f"Connected to server at {self.host}:{self.port}")

except Exception as e:

self.log\_message(f"Error connecting to server: {str(e)}")

def sign\_and\_send(self):

filename = self.file\_entry.get()

if not filename or not os.path.exists(filename):

messagebox.showerror("Error", "Please select a valid file")

return

try:

with open(filename, 'rb') as file:

file\_data = file.read()

value = int(self.value\_entry.get())

bit\_length = int(self.bit\_length\_combobox.get())

self.private\_key = Ed25519PrivateKey.generate()

self.public\_key = self.private\_key.public\_key()

self.signature = self.private\_key.sign(file\_data)

self.proof = zkrp\_prove(int(value), bit\_length)

pub\_key\_bytes = self.public\_key.public\_bytes(

encoding=serialization.Encoding.Raw,

format=serialization.PublicFormat.Raw

)

data\_to\_send = {

'filename': os.path.basename(filename),

'file\_data': file\_data.hex(),

'public\_key': pub\_key\_bytes.hex(),

'signature': self.signature.hex(),

'proof': [str(p) for p in self.proof],

'value': value,

'bit\_length': bit\_length

}

if self.client\_socket:

self.client\_socket.sendall(json.dumps(data\_to\_send).encode())

self.log\_message(f"File {filename} sent with signature and proof")

else:

messagebox.showerror("Error", "Not connected to any server")

except Exception as e:

self.log\_message(f"Error sending file: {str(e)}")

def listen\_for\_files(self):

while True:

try:

if self.server\_socket:

conn, addr = self.server\_socket.accept()

self.log\_message(f"Connection from {addr}")

threading.Thread(target=self.handle\_client, args=(conn,), daemon=True).start()

except:

pass

def handle\_client(self, conn):

try:

data = conn.recv(1024 \* 1024) # Max 1MB

if not data:

return

received\_data = json.loads(data.decode())

received\_data['file\_data'] = bytes.fromhex(received\_data['file\_data'])

received\_data['public\_key'] = bytes.fromhex(received\_data['public\_key'])

received\_data['signature'] = bytes.fromhex(received\_data['signature'])

received\_data['proof'] = [int(p) for p in received\_data['proof']]

public\_key = Ed25519PublicKey.from\_public\_bytes(received\_data['public\_key'])

try:

public\_key.verify(received\_data['signature'], received\_data['file\_data'])

signature\_valid = True

except InvalidSignature:

signature\_valid = False

proof\_valid = zkrp\_verify(

received\_data['proof'][0],

received\_data['proof'],

received\_data['bit\_length']

)

save\_dir = os.path.join(os.getcwd(), "received\_files")

os.makedirs(save\_dir, exist\_ok=True)

save\_path = os.path.join(save\_dir, received\_data['filename'])

with open(save\_path, 'wb') as file:

file.write(received\_data['file\_data'])

self.received\_files.append({

'path': save\_path,

'data': received\_data,

'signature\_valid': signature\_valid,

'proof\_valid': proof\_valid

})

self.root.after(0, self.update\_received\_files)

self.log\_message(f"Received file: {received\_data['filename']}")

except Exception as e:

self.log\_message(f"Error handling client: {str(e)}")

finally:

conn.close()

def update\_received\_files(self):

for item in self.tree.get\_children():

self.tree.delete(item)

for i, file\_info in enumerate(self.received\_files):

filename = os.path.basename(file\_info['path'])

sig\_valid = "Yes" if file\_info['signature\_valid'] else "No"

proof\_valid = "Yes" if file\_info['proof\_valid'] else "No"

overall\_valid = "Yes" if file\_info['signature\_valid'] and file\_info['proof\_valid'] else "No"

self.tree.insert("", "end", values=(filename, sig\_valid, proof\_valid, overall\_valid))

def verify\_selected(self):

selected\_item = self.tree.selection()

if not selected\_item:

messagebox.showerror("Error", "Please select a file to verify")

return

item = self.tree.item(selected\_item)

filename = item['values'][0]

file\_info = next((f for f in self.received\_files if os.path.basename(f['path']) == filename), None)

if file\_info:

result = f"Verification results for {filename}:\n"

result += f"Signature valid: {'Yes' if file\_info['signature\_valid'] else 'No'}\n"

result += f"Range proof valid: {'Yes' if file\_info['proof\_valid'] else 'No'}\n"

result += f"Value: {file\_info['data']['value']}\n"

result += f"Bit length: {file\_info['data']['bit\_length']}"

messagebox.showinfo("Verification Results", result)

else:

messagebox.showerror("Error", "File not found")

def log\_message(self, message):

self.log\_text.insert(END, message + "\n")

self.log\_text.see(END)

def on\_closing(self):

if self.server\_socket:

self.server\_socket.close()

if self.client\_socket:

self.client\_socket.close()

self.root.destroy()

if \_\_name\_\_ == "\_\_main\_\_":

root = Tk()

app = FileSharingApp(root)

root.protocol("WM\_DELETE\_WINDOW", app.on\_closing)

## root.mainloop()

## Приложение Б Коды функциональных тестов

import unittest

import os

import json

import tempfile

from unittest.mock import patch, MagicMock

from cryptography.hazmat.primitives.asymmetric.ed25519 import Ed25519PrivateKey, Ed25519PublicKey

from cryptography.exceptions import InvalidSignature

from pybulletproofs import zkrp\_prove, zkrp\_verify

class TestZKPSignatureApp(unittest.TestCase):

def setUp(self):

self.private\_key = Ed25519PrivateKey.generate()

self.public\_key = self.private\_key.public\_key()

self.small\_data = os.urandom(256) # 256 байт

self.medium\_data = os.urandom(1024) # 1 КБ

self.large\_data = os.urandom(4096) # 4 КБ

self.value = 123

self.bit\_length = 16

def test\_key\_pair\_generation(self):

priv\_key = Ed25519PrivateKey.generate()

pub\_key = priv\_key.public\_key()

test\_data = b"test\_data"

signature = priv\_key.sign(test\_data)

pub\_key.verify(signature, test\_data)

def test\_zkp\_proof\_verification(self):

proof = zkrp\_prove(self.value, self.bit\_length)

self.assertTrue(zkrp\_verify(proof[0], proof[1], self.bit\_length))

def test\_sign\_verify\_small\_data(self):

self.\_test\_sign\_verify(self.small\_data)

def test\_sign\_verify\_medium\_data(self):

self.\_test\_sign\_verify(self.medium\_data)

def test\_sign\_verify\_large\_data(self):

self.\_test\_sign\_verify(self.large\_data)

def \_test\_sign\_verify(self, data):

signature = self.private\_key.sign(data)

proof = zkrp\_prove(self.value, self.bit\_length)

self.public\_key.verify(signature, data)

self.assertTrue(zkrp\_verify(proof[0], proof[1], self.bit\_length))

@patch('threading.Thread')

def test\_parallel\_verification(self, mock\_thread):

from concurrent.futures import ThreadPoolExecutor

def verify\_task(data):

signature = self.private\_key.sign(data)

proof = zkrp\_prove(self.value, self.bit\_length)

return (

self.public\_key.verify(signature, data),

zkrp\_verify(proof[0], proof[1], self.bit\_length)

)

with ThreadPoolExecutor(max\_workers=100) as executor:

futures = [executor.submit(verify\_task, self.small\_data) for \_ in range(100)]

results = [f.result() for f in futures]

for sig\_valid, proof\_valid in results:

self.assertTrue(sig\_valid)

self.assertTrue(proof\_valid)

def test\_tampered\_data(self):

signature = self.private\_key.sign(self.small\_data)

tampered\_data = self.small\_data + b"tampered"

with self.assertRaises(InvalidSignature):

self.public\_key.verify(signature, tampered\_data)

def test\_invalid\_signature(self):

invalid\_signature = os.urandom(64)

with self.assertRaises(InvalidSignature):

self.public\_key.verify(invalid\_signature, self.small\_data)

def test\_invalid\_zkp\_proof(self):

proof = zkrp\_prove(self.value, self.bit\_length)

tampered\_proof = (proof[0], proof[1] + "tampered")

self.assertFalse(zkrp\_verify(tampered\_proof[0], tampered\_proof[1], self.bit\_length))

@patch('socket.socket')

def test\_network\_send\_receive(self, mock\_socket):

mock\_conn = MagicMock()

mock\_socket.return\_value.accept.return\_value = (mock\_conn, ('127.0.0.1', 12345))

test\_data = {

'filename': 'test.txt',

'file\_data': self.small\_data.hex(),

'public\_key': self.public\_key.public\_bytes(

encoding=serialization.Encoding.Raw,

format=serialization.PublicFormat.Raw

).hex(),

'signature': self.private\_key.sign(self.small\_data).hex(),

'proof': [str(p) for p in zkrp\_prove(self.value, self.bit\_length)],

'value': self.value,

'bit\_length': self.bit\_length

}

mock\_conn.recv.return\_value = json.dumps(test\_data).encode()

self.app.handle\_client(mock\_conn)

mock\_conn.sendall.assert\_not\_called()

mock\_conn.close.assert\_called\_once()

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

## Приложение В Коды нагрузочных тестов

import requests

import threading

import time

import random

import os

BASE\_URL = "http://localhost:5000"

DATA\_1KB = os.urandom(1024)

def sign\_request():

payload = {

"data": DATA\_1KB.hex(),

"value": random.randint(0, 255),

"bit\_length": 16

}

start\_time = time.time()

response = requests.post(f"{BASE\_URL}/sign", json=payload)

latency = (time.time() - start\_time) \* 1000

return latency

def verify\_request(signature, proof, public\_key):

payload = {

"data": DATA\_1KB.hex(),

"signature": signature,

"proof": proof,

"public\_key": public\_key

}

start\_time = time.time()

response = requests.post(f"{BASE\_URL}/verify", json=payload)

latency = (time.time() - start\_time) \* 1000

return latency

def worker():

while True:

if random.random() < 0.7:

sign\_request()

else:

dummy\_signature = "a1b2c3..."

dummy\_proof = ["123", "456"]

dummy\_pubkey = "x1y2z3..."

verify\_request(dummy\_signature, dummy\_proof, dummy\_pubkey)

threads = []

for \_ in range(1000):

t = threading.Thread(target=worker)

t.start()

threads.append(t)

for t in threads:

t.join()