МИНОБРНАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

«Пермский государственный национальный исследовательский университет»

Физико-математический институт

Кафедра радиоэлектроники и защиты информации

КУРСОВАЯ РАБОТА

**Программная реализация протокола электронного голосования с одной Центральной комиссией на базе «слепой» подписи**

|  |  |
| --- | --- |
| К защите допускаю:  заведующий кафедрой  радиоэлектроники и защиты информации  \_\_\_\_\_\_\_\_\_\_\_\_ И.В.Лунегов | Исполнитель  студент 4 курса,  специальность "Информационная безопасность автоматизированных систем"  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.А.Москоков |
|  | Научные руководители:  доцент физико-математического института  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.П.Шкарапута |
|  | доцент кафедры радиоэлектроники и защиты информации  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А.Федоренко |

Пермь 2025

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| 1 ВВЕДЕНИЕ | 4 |
| 2 ИСПОЛЬЗОВАНИЕ КРИПТОГРАФИИ | 6 |
| 2. 1 Основы криптографии | 6 |
| 2.1.1 История криптографии и её роль в защите информации | 6 |
| 2.1.2Введение в симметричное и асимметричное шифрование | 7 |
| 2.1.3 Применение криптографии в защищённых системах | 8 |
| 2.2 Асимметричное шифрование и ЭЦП | 9 |
| 2.2.1 Принципы работы с асимметричным шифрованием: алгоритмы RSA, ЭЦП | 9 |
| 2.2.2 Роль ЭЦП в аутентификации и защите данных | 10 |
| 2.2.3 Слепая подпись в контексте голосования | 11 |
| 2.3 Симметричное шифрование в системах защиты информации | 12 |
| 2.3.1 Принципы работы симметричного шифрования, алгоритмы AES, DES | 12 |
| 2.3.2 Применение симметричного шифрования для хранения кэшированных данных в вашем проекте | 13 |
| 2.3.3 Заключение | 15 |
| 2.4 Анализ существующих решений и протоколов голосования | 15 |
| 2.4.1 Обзор существующих протоколов электронного голосования | 15 |
| 2.4.2 Преимущества и недостатки различных методов | 16 |
| 2.4.3 Сравнение с реальными системами электронного голосования | 18 |
| 2.4.4 Заключение | 19 |
| 3 ПРОЕКТИРОВАНИЕ КОМПЛЕКСА ПРИЛОЖЕНИЙ | 20 |
| 3.1 Описание системы и её архитектуры | 20 |
| 3.1.1 Обоснование выбора языка программирования Python | 20 |
| 3.1.2 Архитектура системы голосования | 20 |
| 3.1.3 Взаимодействие компонентов системы | 20 |
| 3.1.4 Основные принципы функционирования | 21 |
| 3.1.5 Особенности архитектурных решений | 21 |
| 3.2 Протокол голосования с одной Центральной комиссией на основе «слепой» подписи | 22 |
| 3.3 Проектирование баз данных | 23 |
| 3.3.1 Обоснование использования PostgreSQL как основной СУБД | 23 |
| 3.3.2 Анализ типов информации, подлежащей хранению: проектирование таблиц и связей | 24 |
| 3.3.3 База данных центральной избирательная комиссия(CIK\_BD) | 24 |
| 3.3.4 Базы данных клиентов(Сlient\_BD) | 27 |
| 3.3.5. База данных центр сертификации ключей(CSK\_BD) | 28 |
| 3.3.6 Общий вывод по проектированию структуры БД | 29 |
| 3.4 Система обмена сообщениями | 29 |
| 3.5 Первоначальная инициализация приложений | 33 |
| 3.5.1 Механизм создания кэш-файлов при первом запуске | 33 |
| 3.5.2 Применение симметричного шифрования для защиты данных | 33 |
| 3.5.3 Алгоритмы создания, шифрования и проверки хешей | 33 |
| 3.5.4 Вывод | 34 |
| 3.6 Разработка программных модулей | 34 |
| 4 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ | 37 |
| 4.1 Разработанные методы | 37 |
| 4.1.1 Реализован протокол в программной среде | 37 |
| 4.1.2 Разработаны методы для автоматической инициализации структуры баз данных | 41 |
| 4.1.3 Метод реализующий чтение и создания файлов(сообщений) | 42 |
| 4.1.4 Метод первоначальной инициализация приложений чтения кэша | 43 |
| 4.1.5 Реализация программных модулей | 46 |
| 4.2 Разработана криптографическая система для голосования | 48 |
| 4.2.1 Общая информация | 48 |
| 4.2.2 Этапы криптографического протокола | 48 |
| 4.2.3 Заключение | 50 |
| 4.3 Тестирование работоспособности комплекса | 50 |
| 4.3.1 Цель тестирования | 50 |
| 4.3.2 Активация приложений | 51 |
| 4.3.3 Сценарий голосования | 52 |
| 4.3.4 Завершение голосования | 54 |
| 4.3.5. Результат тестирования | 55 |
| 4.4 Сравнение с аналогичными решениями | 55 |
| 4.4.1 Сравнение с классическими протоколами электронного голосования | 55 |
| 4.4.2 Сравнение с блокчейн-системами | 56 |
| 4.4.3 Сравнение с государственными системами (Москва, Эстония) | 57 |
| 4.4.4 Уникальные особенности разработанного решения | 57 |
| 4.4.5 Предложения по улучшению | 57 |
| 4.4.6 Заключение | 58 |
| 5 ЗАКЛЮЧЕНИЕ | 59 |
| 5.1. Основные выводы | 59 |
| 5.2. Предложения по улучшению системы | 59 |
| 5.3. Область применения и перспективы развития | 60 |
| СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ | 62 |
| ПРИЛОЖЕНИЕ | 64 |

1 ВВЕДЕНИЕ

Системы электронного голосования, обеспечивающие анонимность и безопасность, становятся всё более востребованными в различных сферах — от государственных выборов до корпоративных голосований. Одной из главных проблем таких систем является обеспечение защиты от фальсификаций и анонимность голосующих. Для решения этих задач широко используются криптографические методы, такие как асимметричное шифрование и электронная цифровая подпись (ЭЦП) [1, 2]. Эти технологии гарантируют защиту данных, а также позволяют проводить безопасное и анонимное голосование.

Протоколы голосования с использованием «слепой» подписи являются одним из самых защищённых и эффективных способов организации электронных выборов [3], поскольку они позволяют скрывать информацию о голосах от Центральной избирательной комиссии (ЦИК) и других участников системы, обеспечивая полную анонимность. Однако для полноценной реализации таких систем требуется разработка программных компонентов, которые будут обеспечивать работу с базами данных, обработку и обмен сообщениями, а также корректное выполнение протокола.

В данной курсовой работе реализована система, которая полностью воплощает протокол голосования с одной Центральной комиссией на базе «слепой» подписи. Включенные в систему компоненты — база данных PostgreSQL для хранения информации [4], система папок для обмена сообщениями и кэш для безопасного хранения данных — обеспечивают полноценную работу протокола голосования, включая создание, обработку и проверку голосов.

Решение, разработанное в рамках данной работы, находит практическое применение в нескольких областях:

1. Электронное голосование: разработанная система может использоваться для организации анонимных и безопасных голосований в различных сферах, таких как выборы, корпоративные голосования, опросы и другие формы голосования [5].

2. Образовательные и исследовательские цели: система может быть полезной для обучения студентов принципам работы с криптографическими протоколами, а также для проведения исследований в области информационной безопасности и защиты данных [6].

3. Разработка и тестирование протоколов безопасности: система может служить основой для создания и тестирования новых методов защиты данных и обеспечения анонимности в процессе голосования [7].

Цель работы — разработка системы, реализующей протокол голосования с одной Центральной комиссией на базе «слепой» подписи. Система должна обеспечивать полноценную работу протокола, включая создание базы данных для хранения информации, обработку, хранение и создание сообщений, а также использование кэш-системы для хранения данных.

Задачи работы:

1. Обоснование выбора протокола голосования — исследование и обоснование применения протокола голосования с использованием «слепой» подписи, определение его преимуществ и недостатков.

2. Разработка базы данных и кэш-системы — создание базы данных в PostgreSQL для хранения информации о голосах и избирателях, а также разработка системы для безопасного хранения и обработки данных (кэш-система).

3. Разработка структуры сообщений — определение структуры и формата сообщений для обмена данными между компонентами системы, создание механизмов взаимодействия между программами через систему папок.

4. Реализация протокола голосования — разработка всех этапов реализации протокола, включая генерацию ключей, подписание и верификацию меток, а также процесс голосования.

5. Разработка и тестирование программы — создание программных компонентов, которые реализуют протокол голосования, тестирование их функциональности, обеспечение корректной работы с базой данных, сообщениями и кэшированием.

Результаты работы могут быть полезны для различных задач:

1. Организация электронного голосования — разрабатываемая система может быть использована для создания защищённого и анонимного голосования в рамках организаций, выборов, опросов и других процессов, где требуется сохранить конфиденциальность голосующих.

2. Образовательные проекты — система может служить учебным примером для изучения криптографических протоколов, а также методов защиты и анонимности в информационных системах.

3. Тестирование криптографических решений — система может быть использована для анализа и улучшения методов криптографической защиты данных в рамках исследований и разработки новых решений в области безопасности.

2 ИСПОЛЬЗОВАНИЕ КРИПТОГРАФИИ

2.1 Основы криптографии

2.1.1 История криптографии и её роль в защите информации

Криптография, как наука о защите информации, имеет многовековую историю, начиная с древних времён. Её основные задачи — обеспечение конфиденциальности, целостности и подлинности данных. Применение криптографии для защиты информации сыграло важнейшую роль в обеспечении безопасности в разных исторических эпохах, начиная от военных операций до современного интернета.

Древняя криптография. Первые примеры использования криптографии восходят к Древнему Египту и Месопотамии, где использовались различные способы скрытия информации, такие как замены букв и использование секретных символов. Однако широкое применение криптография получила в эпоху Римской империи. Известным примером является использование Цезаревого шифра, в котором каждую букву алфавита заменяли на букву, стоящую на несколько позиций дальше [8].

Средние века. В этот период криптография активно использовалась в дипломатических и военных целях. Одним из значимых событий было введение методов шифрования для защиты военных сообщений, таких как использование перестановочных и подстановочных шифров [9].

Новая эпоха. В XVII-XVIII веках с развитием математики и науки криптография начала развиваться как отдельная область знаний. В этот период появились первые шифры, основанные на математических принципах. Например, шифр Виженера, разработанный в 1586 году, стал значительным шагом в развитии теории криптографии [10].

Современная криптография. В XX веке криптография стала неотъемлемой частью информационных технологий. Ключевыми событиями стали изобретение симметричных алгоритмов (например, DES, AES) и асимметричной криптографии (например, алгоритм RSA). С развитием компьютеров и сетевых технологий криптография приобрела исключительное значение для защиты данных в условиях растущих угроз и кибератак [11].

Роль криптографии в защите информации. Криптография играет ключевую роль в современных информационных системах, обеспечивая защиту данных от несанкционированного доступа. Основные задачи криптографии включают:

1. Конфиденциальность — защита информации от несанкционированного доступа.

2. Целостность — обеспечение того, чтобы информация не была изменена или искажена в процессе хранения или передачи.

3. Аутентификация — проверка подлинности источника информации.

4. Неотказуемость — защита от отказа от действий, таких как отправка сообщения или голосование.

Без криптографических методов защита данных в интернете и на локальных устройствах была бы невозможна, что ставит криптографию в центр информационной безопасности [12].

2.1.2Введение в симметричное и асимметричное шифрование

В криптографии различают два основных типа шифрования: симметричное и асимметричное.

Симметричное шифрование (или криптография с секретным ключом) — это метод шифрования, при котором для шифрования и дешифрования используется один и тот же ключ. Процесс симметричного шифрования основывается на применении математических операций, которые преобразуют исходное сообщение в зашифрованный текст и обратно [13].

Ключевыми особенностями симметричного шифрования являются:

-Быстрота работы. Алгоритмы симметричного шифрования, такие как AES (Advanced Encryption Standard), DES (Data Encryption Standard), RC4, используют простые математические операции, что делает их быстрыми в исполнении.

-Проблема передачи ключа. Главным недостатком является необходимость безопасной передачи ключа между отправителем и получателем. Если ключ окажется в руках злоумышленника, все данные можно будет расшифровать [14].

Примеры симметричных алгоритмов:

-AES (Advanced Encryption Standard) — широко используется в защите данных для шифрования файлов, данных в интернете и в мобильных приложениях.

-DES (Data Encryption Standard) — устаревший алгоритм, который использовался в финансовых и банковских системах.

Асимметричное шифрование (или криптография с открытым ключом) — это метод шифрования, при котором используются два ключа: один для шифрования и другой для дешифрования. Обычно это пара ключей: открытый и закрытый. Открытый ключ доступен всем, а закрытый ключ хранится в секрете [15].

Преимущества асимметричного шифрования:

-Безопасность передачи ключей. Открытый ключ может быть передан по открытому каналу, и его невозможно использовать для расшифровки данных. Только закрытый ключ может расшифровать информацию.

-Цифровая подпись. Асимметричное шифрование позволяет использовать цифровую подпись для обеспечения подлинности сообщений, что невозможно при симметричном шифровании.

Примеры асимметричных алгоритмов:

-RSA (Rivest-Shamir-Adleman) — один из самых известных и широко используемых алгоритмов асимметричного шифрования. Он использует два больших простых числа для создания ключевой пары.

-ECC (Elliptic Curve Cryptography) — более современный алгоритм, который использует эллиптические кривые для создания ключей меньшего размера при одинаковом уровне безопасности.

Применение асимметричного шифрования критически важно для безопасности интернет-коммуникаций, таких как SSL/TLS для HTTPS, и в приложениях, где необходима аутентификация пользователей, например, при электронной подписи.

2.1.3 Применение криптографии в защищённых системах

Криптография используется в широком спектре защищённых систем для обеспечения безопасности данных. Наиболее распространённые области применения криптографии включают:

1. Безопасность коммуникаций. Криптография используется для защиты данных при передаче через публичные сети, такие как интернет. Это включает в себя шифрование электронной почты, чат-сообщений, а также создание защищённых каналов связи, таких как VPN (Virtual Private Network) и SSL/TLS [16].

2. Электронная коммерция. В интернет-торговле криптография используется для обеспечения безопасности платежей, защиты данных о кредитных картах и личной информации. Протокол SSL (Secure Sockets Layer) и его преемник TLS широко применяются для создания защищённых соединений между клиентами и серверами.

3. Электронная подпись. Для защиты документов, подписанных в электронном виде, используется цифровая подпись, которая подтверждает подлинность и целостность документа. Применение цифровых подписей активно используется в юридической и финансовой сфере.

4. Защита данных в базах данных. Криптография применяется для шифрования данных в базах данных, чтобы предотвратить утечку конфиденциальной информации при её хранении или передаче.

5. Электронное голосование. В системах электронного голосования криптография используется для обеспечения анонимности голосующих, защиты данных голосов и предотвращения фальсификаций. Протоколы, основанные на «слепой» подписи, обеспечивают безопасность при голосовании, а также позволяют защищать идентификацию участников голосования [17].

6. Защита персональных данных. С увеличением объёма персональных данных, которые передаются и хранятся в интернете, криптография становится важным инструментом для защиты конфиденциальности данных и соблюдения законодательства, такого как GDPR (General Data Protection Regulation) [18].

Таким образом, криптография является основой для обеспечения безопасности в самых различных сферах, от личных данных до глобальных коммуникационных систем, и играет ключевую роль в защите информации в условиях современных угроз.

2.2 Асимметричное шифрование и ЭЦП

2.2.1 Принципы работы с асимметричным шифрованием: алгоритмы RSA, ЭЦП

Асимметричное шифрование, также известное как криптография с открытым ключом, основывается на использовании двух ключей — открытого и закрытого, которые взаимно связаны. Принцип работы асимметричной криптографии заключается в том, что для шифрования используется один ключ, а для дешифрования — другой, при этом знание одного ключа не даёт возможности вычислить другой. Это делает возможным безопасную передачу данных, даже если каналы связи не защищены [11].

Ключевыми преимуществами асимметричного шифрования являются:

-Безопасность обмена ключами. Поскольку открытый ключ может быть передан по незащищенному каналу, а только закрытый ключ необходим для расшифровки данных, существует минимальный риск того, что ключ будет перехвачен.

-Цифровая подпись. Асимметричное шифрование позволяет использовать цифровую подпись для проверки подлинности данных и обеспечения их целостности.

Одним из наиболее популярных и широко используемых алгоритмов асимметричного шифрования является RSA (Rivest-Shamir-Adleman). Этот алгоритм был разработан в 1977 году и до сих пор является стандартом для многих приложений. В RSA используется два больших простых числа, которые служат основой для формирования открытого и закрытого ключей. Алгоритм RSA может использоваться как для шифрования данных, так и для создания цифровых подписей.

Процесс работы RSA заключается в следующем:

1. Генерация ключей. Для создания пары ключей выбираются два простых числа, их произведение используется как модуль для вычислений. Также выбираются открытый и закрытый экспоненты для шифрования и дешифрования.

2. Шифрование. Сообщение преобразуется в число, которое затем возводится в степень открытого ключа и берется по модулю произведения двух простых чисел.

3. Дешифрование. Полученное зашифрованное сообщение возводится в степень закрытого ключа и снова берется по модулю.

Однако использование RSA требует вычислительных ресурсов, особенно при обработке больших объёмов данных. Поэтому на практике часто используется гибридный подход, при котором для шифрования данных используется симметричный ключ, а сам ключ шифруется с помощью RSA.

Электронная цифровая подпись (ЭЦП) представляет собой криптографический механизм, который используется для обеспечения подлинности и целостности данных, а также для подтверждения их происхождения. ЭЦП является основой для защиты данных в системах, таких как электронная почта, онлайн-банки и системы электронной торговли.

Процесс создания и проверки ЭЦП выглядит следующим образом:

1. Создание подписи. Сначала создается хеш-сумма документа с использованием криптографической хеш-функции. Эта хеш-сумма затем шифруется с использованием закрытого ключа подписанта, создавая подпись.

2. Проверка подписи. Получатель сообщения может дешифровать подпись с использованием открытого ключа отправителя и сравнить расшифрованную хеш-сумму с хеш-суммой полученного документа. Если хеш-суммы совпадают, значит, данные не были изменены, и подпись является подлинной.

Таким образом, ЭЦП позволяет гарантировать, что сообщение не было подделано и было отправлено именно тем лицом, которое указано в подписанном документе [12].

2.2.2 Роль ЭЦП в аутентификации и защите данных

Цифровая подпись играет центральную роль в системах, где необходимо удостовериться в подлинности сообщения и идентификации отправителя. Она является неотъемлемым элементом в процессах аутентификации и защиты данных.

Аутентификация с использованием ЭЦП позволяет убедиться в том, что сообщение поступило от конкретного отправителя. В отличие от традиционных методов аутентификации (например, с использованием пароля), цифровая подпись не может быть подделана без доступа к закрытому ключу отправителя. Этот метод аутентификации используется в широком спектре приложений, включая электронные платежи, авторизацию пользователей в системах, а также в системах электронного голосования.

Кроме того, ЭЦП обеспечивает целостность данных. Даже малейшее изменение в документе приведет к несоответствию между хеш-суммой, подписанной с использованием закрытого ключа, и хеш-суммой, вычисленной при проверке подписи с использованием открытого ключа. Это является важным элементом защиты от внезапных изменений данных и фальсификации [14].

В области электронного голосования, например, ЭЦП используется для обеспечения подлинности и безопасности голосов. Каждый голосующий подписывает свой голос с помощью ЭЦП, что подтверждает его подлинность. Поскольку в этом случае ключи RSA широко используются для создания и проверки подписей, эта технология значительно повышает уровень защиты и предотвращает подделку голосов.

Применение ЭЦП в области защиты данных также включает:

-Защиту от несанкционированного доступа в электронных системах.

-Проверку подлинности документов, таких как контракты, финансовые отчёты и другие юридически важные материалы.

-Обеспечение безопасности в электронных банковских системах и при интернет-платежах.

2.2.3 Слепая подпись в контексте голосования

Слепая подпись является специализированным криптографическим механизмом, который используется в протоколах электронного голосования для обеспечения анонимности и конфиденциальности голосующих. Суть метода заключается в том, что голосующий может подписать свой выбор, не раскрывая его содержимого тому, кто подписывает.

Принцип работы слепой подписи основан на том, что голосующий создает "слепую" версию своего выбора — преобразует данные о голосе в такой вид, что сам выбор скрыт от подписанта. Этот процесс обычно включает несколько этапов:

1. Создание слепого сообщения. Голосующий создаёт сообщение, содержащее его выбор, и преобразует его с помощью криптографической функции так, чтобы подписант не мог узнать, что именно он подписывает.

2. Подпись. После этого голосующий отправляет слепое сообщение на подпись Центральной комиссии, которая подписывает его с помощью своего закрытого ключа.

3. Возвращение подписанного сообщения. Получив подписанное сообщение, голосующий расшифровывает его и отправляет на сервер голосования, где его выбор уже будет защищён с помощью цифровой подписи, подтверждающей его подлинность.

Слепая подпись обеспечивает две основные цели:

1. Анонимность. Голосующий может быть уверен в том, что его выбор останется анонимным, поскольку Централизованная комиссия не видит его голос до момента подачи.

2. Неотказуемость. Подписанное сообщение служит доказательством того, что голос был подан именно этим человеком, и он не может впоследствии отказать от своего выбора [17].

Применение слепой подписи в протоколах голосования является важным шагом в направлении повышения безопасности и анонимности процессов голосования. Этот механизм значительно снижает риски манипуляций с результатами голосования, а также защищает участников от давления и манипуляций со стороны третьих лиц.

Асимметричное шифрование и электронная цифровая подпись являются основой современных методов защиты информации. Протоколы, использующие эти технологии, играют ключевую роль в обеспечении безопасности данных, а также в различных системах, включая электронные голосования, электронные платежи и систему защиты личных данных. В частности, слепая подпись является эффективным инструментом в контексте защищённых систем голосования, обеспечивая анонимность и предотвращая фальсификацию голосов.

2.3 Симметричное шифрование в системах защиты информации

2.3.1 Принципы работы симметричного шифрования, алгоритмы AES, DES

Симметричное шифрование является одним из основных способов защиты информации в криптографии, где для шифрования и дешифрования используется один и тот же ключ. Это означает, что как отправитель, так и получатель должны иметь доступ к секретному ключу, который используется для выполнения обеих операций. Симметричное шифрование является более быстрым и менее ресурсоемким по сравнению с асимметричными методами, что делает его особенно подходящим для шифрования больших объемов данных [13].

Основным элементом симметричного шифрования является алгоритм, который определяет, как именно будет происходить преобразование исходных данных в зашифрованный текст и наоборот. Существуют различные алгоритмы симметричного шифрования, каждый из которых имеет свои особенности и области применения. Наиболее популярными и широко используемыми являются AES (Advanced Encryption Standard) и DES (Data Encryption Standard).

DES был разработан в 1970-х годах и долгое время считался стандартом для защиты данных. Он использует 56-битный ключ и шифрует данные блоками по 64 бита. Однако с развитием вычислительных мощностей, алгоритм DES стал уязвимым для атак на основе полного перебора ключей. Сегодня он уже считается устаревшим и не используется в современных системах, однако является важным историческим этапом в развитии криптографии.

Для преодоления недостатков DES был создан AES. Этот алгоритм использует ключи длиной 128, 192 и 256 бит, что обеспечивает гораздо большую стойкость против атак. AES работает с блоками данных по 128 бит и поддерживает три различных режима шифрования: с ключом длиной 128, 192 и 256 бит. AES стал стандартом для шифрования в государственных и коммерческих приложениях по всему миру, включая банковские системы, протоколы SSL/TLS для Интернета, а также в приложениях для защиты данных на мобильных устройствах [19].

Принцип работы симметричного шифрования обычно включает несколько этапов:

1. Разбиение данных на блоки. Данные, которые необходимо зашифровать, разбиваются на блоки фиксированного размера. В случае с AES это блоки по 128 бит, а для DES — по 64 бита.

2. Применение ключа к каждому блоку. Алгоритм использует ключ для шифрования каждого блока данных. В случае AES используется несколько раундов преобразований, которые включают такие операции, как замена байтов, сдвиги строк, и выполнение операций с ключами.

3. Шифрование и дешифрование. Процесс шифрования заключается в применении математических операций с ключом, которые преобразуют данные в зашифрованный вид. Дешифрование происходит с использованием того же ключа, но в обратном порядке.

При этом стоит отметить, что алгоритмы симметричного шифрования, такие как AES, обеспечивают высокую степень защиты данных, но их безопасность сильно зависит от надежности используемого ключа. Если ключ будет скомпрометирован, вся система шифрования будет уязвима. Поэтому важной частью криптографической системы является также безопасное хранение и управление ключами [17].

2.3.2 Применение симметричного шифрования для хранения кэшированных данных в вашем проекте

В контексте разработки защищенных систем, таких как система голосования, симметричное шифрование может быть использовано для защиты кэшированных данных. Кэш — это временное хранилище, которое ускоряет доступ к данным, часто используемым в приложении. Однако, поскольку данные в кэше могут быть конфиденциальными, их защита с использованием шифрования является важной частью системы безопасности.

Для того чтобы данные в кэше оставались защищёнными даже в случае компрометации серверов или устройств хранения, необходимо шифровать их перед сохранением и дешифровать только при необходимости доступа. В случае с вашим проектом, где используется система для обработки сообщений и работы с базами данных, важным элементом является обеспечение безопасности временных данных, которые могут быть кэшированы для ускорения процессов.

Применение AES для кэширования в вашем проекте может выглядеть следующим образом:

1. Шифрование данных перед кэшированием. Все данные, которые сохраняются в кэш, шифруются с использованием симметричного ключа, например, с использованием алгоритма AES с ключом длиной 256 бит. Это гарантирует, что даже если кэш окажется скомпрометирован, данные будут оставаться защищенными.

2. Дешифрование данных при извлечении. Когда данные из кэша должны быть использованы, они сначала извлекаются и дешифруются с использованием того же ключа, который был использован для их шифрования.

3. Управление ключами. Важно безопасно управлять симметричными ключами, поскольку если ключ будет раскрыт, весь механизм шифрования станет уязвимым. Для этого можно использовать отдельные компоненты системы для хранения и управления ключами, такие как HSM (Hardware Security Module) или системы управления ключами.

Симметричное шифрование для кэширования данных имеет несколько явных преимуществ. Во-первых, оно быстрое и эффективное, что особенно важно для работы с большими объемами данных, как в случае с вашим проектом. Во-вторых, использование AES с длинными ключами обеспечивает высокий уровень безопасности, что делает этот метод подходящим для защиты данных в реальных приложениях.

Одним из важных аспектов является выбор подходящего режима шифрования. Для кэшированных данных часто используется режим CBC (Cipher Block Chaining), который обеспечивает дополнительную безопасность за счет связывания каждого блока с предыдущим. Это предотвращает возможность атаки, при которой одинаковые данные дают одинаковые шифрованные блоки [14].

Кроме того, важно учитывать сроки хранения ключей и периодическую смену ключей. В системах, где данные обновляются часто, использование одного и того же ключа на протяжении долгого времени может повысить риски компрометации. Рекомендуется регулярно обновлять ключи и использовать протоколы безопасной передачи ключей, чтобы минимизировать риски.

2.3.3 Заключение

Симметричное шифрование является неотъемлемой частью защиты данных в современных информационных системах. Его использование в контексте вашего проекта, особенно для защиты кэшированных данных, обеспечивает высокий уровень безопасности при минимальных затратах ресурсов. Алгоритмы AES и DES, несмотря на устаревание последнего, продолжают оставаться основой для многих приложений в области криптографии.

Особое внимание следует уделить управлению ключами, так как безопасность всей системы напрямую зависит от того, как будут защищены ключи, используемые для шифрования. Внедрение симметричного шифрования для кэширования и хранения конфиденциальных данных является важным шагом в направлении создания безопасных и эффективных систем, которые могут противостоять современным угрозам безопасности.

2.4 Анализ существующих решений и протоколов голосования

2.4.1 Обзор существующих протоколов электронного голосования

Электронное голосование представляет собой использование информационных технологий и средств связи для осуществления голосования, обработки и подсчета голосов. В зависимости от сложности, масштабируемости и требуемого уровня безопасности различаются подходы и протоколы, используемые для реализации таких систем. Рассмотрим несколько типов существующих протоколов.

1. Централизованные системы голосования

В централизованных системах, как правило, используется единый сервер для сбора и обработки голосов. Примером такого подхода является система Direct Recording Electronic (DRE), которая позволяет избирателям голосовать через специализированные терминалы. Голоса отправляются на центральный сервер, где они записываются и подсчитываются. Эти системы достаточно просты в использовании и подходят для небольших выборов. Однако они имеют ряд недостатков, таких как высокие риски утечек данных и атаки на сервер, что может поставить под сомнение честность голосования.

2. Децентрализованные системы на основе криптографии

В этом случае используются криптографические методы для защиты голосов и аутентификации избирателей. Одним из наиболее популярных решений является система, использующая слепую подпись, при которой голосующие подтверждают свой голос, но не могут быть идентифицированы. Такая система предотвращает возможность фальсификации данных и сохраняет анонимность голоса. Применение этой технологии требует высокой степени доверия к центральному органу, который предоставляет подписанные удостоверения голосующих [20].

3. Блокчейн-технологии

В последние годы блокчейн стал популярным решением для реализации защищенных систем голосования. В этой модели все голоса фиксируются в распределенной базе данных, которая обладает высокой степенью защищенности и прозрачности. Каждый голос представляет собой запись, которая не может быть изменена или удалена. Это исключает возможность манипуляций с результатами голосования, а также предоставляет внешним наблюдателям возможность проверки. Примеры таких решений включают голосование на основе Ethereum и Bitcoin blockchain, которые гарантируют целостность и безопасность данных [21].

4. Гибридные системы

Некоторые системы комбинируют элементы централизованных и децентрализованных решений. Гибридные системы могут использовать централизованные серверы для аутентификации и сбора голосов, но применяют криптографию и блокчейн для повышения безопасности и прозрачности. Такой подход позволяет добиться более высоких уровней защиты и при этом уменьшить затраты на инфраструктуру.

2.4.2 Преимущества и недостатки различных методов

Каждый из существующих протоколов электронного голосования имеет свои преимущества и недостатки. Для более четкого понимания рассмотрим преимущества и слабые стороны самых популярных технологий.

1. Блокчейн

Преимущества:

-Прозрачность. Каждый голос записывается в блокчейн, что делает результаты доступными для проверки любым участником.

-Неизменность данных. Записи в блокчейне не могут быть изменены или удалены без согласия всех участников сети, что исключает возможность манипуляций.

-Анонимность. Использование криптографических методов позволяет сохранить анонимность голосующих при проверке их подлинности.

Недостатки:

-Высокие затраты на инфраструктуру. Блокчейн требует значительных вычислительных мощностей для обеспечения безопасности сети, что увеличивает стоимость реализации и поддержки.

-Масштабируемость. Для массовых выборов блокчейн-системы могут столкнуться с проблемами производительности, поскольку каждая транзакция требует вычислительных ресурсов.

-Зависимость от консенсуса. Для обеспечения надежности данных блокчейн использует консенсус, что влечет за собой задержки в обработке транзакций и в принятии решений [21].

2. Система на основе слепой подписи

Преимущества:

-Анонимность. Протокол слепой подписи обеспечивает анонимность голосующих, так как они могут подтвердить свой голос, не раскрывая его содержимого.

-Простота внедрения. Эта технология относительно проста в реализации по сравнению с блокчейном.

-Надежность. Протокол слепой подписи гарантирует, что голос будет учтен, и предотвратит его фальсификацию.

Недостатки:

-Зависимость от центральной авторитетной структуры. Доверие к системе основывается на надежности центрального органа, который управляет процессом голосования.

-Отсутствие полной прозрачности. В отличие от блокчейна, система на базе слепой подписи не предоставляет полной прозрачности результатов голосования, что может вызывать сомнения в их подлинности [22].

3. Централизованные системы (DRE)

Преимущества:

-Простота в использовании. Эти системы имеют простой интерфейс и легко воспринимаются пользователями.

-Быстрая обработка данных. Централизованная система может быстро обработать большое количество голосов, что важно для выборов с высокой явкой.

Недостатки:

-Уязвимость к атакам. Централизованные серверы могут стать целью атак, что ставит под сомнение безопасность и честность выборов.

-Отсутствие полной прозрачности. Поскольку голосование происходит через централизованный сервер, существует риск, что результаты могут быть изменены или манипулированы без ведома пользователей.

2.4.3 Сравнение с реальными системами электронного голосования

Современные системы электронного голосования используют различные протоколы для обеспечения безопасности, анонимности и прозрачности. Рассмотрим несколько реальных примеров таких систем.

**1. Estonian E-Voting**. Эстонская система электронного голосования является одной из самых известных в мире. Она использует систему централизованного хранения данных и основана на криптографическом удостоверении личности. Голосующие используют свои электронные ID-карты для подтверждения личности, а затем голосуют через защищенное интернет-приложение. В отличие от системы на основе блокчейна, эстонская система предоставляет высокую степень удобства, но также зависит от надежности центрального органа, который управляет системой.

**2. Voatz**. Voatz — это система, использующая блокчейн для обеспечения безопасности голосования. Эта система была использована для некоторых выборов в США. Преимущество Voatz заключается в использовании блокчейн-технологии для обеспечения целостности и прозрачности данных. Однако существует несколько критических замечаний относительно безопасности системы, в том числе сомнения по поводу уязвимости мобильных приложений и возможных манипуляций с данными [23].

**3. Helios Voting**. Helios является примером системы, которая использует слепые подписи для обеспечения анонимности голосующих. В этой системе каждый голос подтверждается цифровой подписью, но его содержимое остается скрытым. Helios широко используется для выборов в университетах и других организациях. Это решение предоставляет высокую степень безопасности, но ограничено в масштабируемости для крупных выборов.

2.4.4 Заключение

Анализ существующих решений показывает, что выбор подхода к электронному голосованию зависит от множества факторов, включая уровень требуемой безопасности, масштаб выборов и ресурсы, доступные для развертывания системы. Протоколы на базе блокчейна предоставляют высокую степень прозрачности и защиты от манипуляций, но требуют значительных вычислительных мощностей и ресурсов. Системы на основе слепых подписей обеспечивают анонимность и защищенность, но требуют высокого уровня доверия к центральному органу. Централизованные системы являются более удобными и быстрыми, но при этом они уязвимы для атак и манипуляций с данными.

3 ПРОЕКТИРОВАНИЕ КОМПЛЕКСА ПРИЛОЖЕНИЙ

3.1 Описание системы и её архитектуры

3.1.1 Обоснование выбора языка программирования Python

Выбор языка программирования Python для реализации системы электронного голосования обусловлен несколькими важными факторами. Прежде всего, Python обладает высокой читаемостью кода и простотой синтаксиса, что значительно сокращает время разработки и упрощает сопровождение программных решений [13]. Его богатая экосистема библиотек, включая средства для криптографических операций, взаимодействия с базами данных и работы с файловой системой, делает его оптимальным выбором для реализации защищённых приложений в сжатые сроки [14].

Дополнительным преимуществом Python является широкая поддержка со стороны сообщества и наличие подробной документации, что облегчает интеграцию криптографических примитивов и реализацию защищённых протоколов [15].

3.1.2 Архитектура системы голосования

Разработанная система представляет собой программную реализацию протокола электронного голосования с одной Центральной избирательной комиссией (ЦИК) на базе технологии «слепой» подписи. Архитектура системы включает три компонента:

-Программу для работы избирателя;

-Программу ЦИК;

-Программу для сбора и подсчёта голосов.

Каждый компонент реализован как отдельное терминальное приложение, развёртывающееся независимо и взаимодействующее с остальными посредством файловой системы, служащей упрощённой моделью передачи сообщений между компонентами.

3.1.3 Взаимодействие компонентов системы

Обмен данными между участниками осуществляется с помощью сообщений, представляющих собой текстовые файлы, помещаемые в соответствующие папки. Каждое приложение использует папки inbox и outbox для чтения и отправки сообщений соответственно. Эта система обмена сообщениями представляет собой заглушку, моделирующую работу защищённого канала, и может быть в будущем заменена на полноценную реализацию с использованием сокетов, HTTPS или других средств сетевой передачи данных [19].

3.1.4 Основные принципы функционирования

Сценарий использования системы включает несколько этапов:

1. Инициализация программ. При первом запуске каждая программа создаёт локальный кэш, содержащий зашифрованные настройки и параметры, необходимые для дальнейшей работы, в том числе зашифрованный пароль к базе данных PostgreSQL.

2. Создание инфраструктуры хранения данных. Программы автоматически разворачивают структуру базы данных PostgreSQL, используя встроенные SQL-запросы. Это обеспечивает совместимость и снижает требования к подготовке среды [20].

3. Реализация действий протокола. Каждая программа реализует набор методов, соответствующих шагам криптографического протокола на основе слепой подписи: генерация слепых голосов, подпись и проверка, подсчёт голосов, публикация результатов. Эти действия выполняются в рамках консольного интерфейса, в котором пользователь взаимодействует с системой через меню.

4. Обработка сообщений. Специальные методы реализуют чтение и создание сообщений, их структуру и внутреннее содержание, что обеспечивает корректное прохождение протокола.

3.1.5 Особенности архитектурных решений

Подход к архитектуре системы базируется на разделении ответственности: каждая программа содержит свой набор функций, обрабатывающих только те этапы, которые ей доступны по логике протокола. Использование файловой системы в качестве канала обмена упростило разработку и дало возможность сосредоточиться на реализации самого протокола и механизмов защиты данных, а не на сетевом взаимодействии [21].

Также важным элементом архитектуры является наличие внутреннего кэша, который используется программами для хранения промежуточных и служебных данных. Эти данные защищаются с помощью симметричного шифрования, что обеспечивает конфиденциальность даже в случае несанкционированного доступа к кэшу.

3.2 Протокол голосования с одной Центральной комиссией на основе «слепой» подписи

Данный протокол реализует электронное голосование с одной Центральной избирательной комиссией (ЦИК), обеспечивая анонимность избирателя за счёт применения технологии «слепой» цифровой подписи. Схема голосования изображена на рис 1.Основная идея заключается в том, чтобы подписывать не сам бюллетень, а уникальную неперсонифицированную метку, позволяющую доказать факт голосования и его корректность, не раскрывая при этом личность избирателя и его выбор.

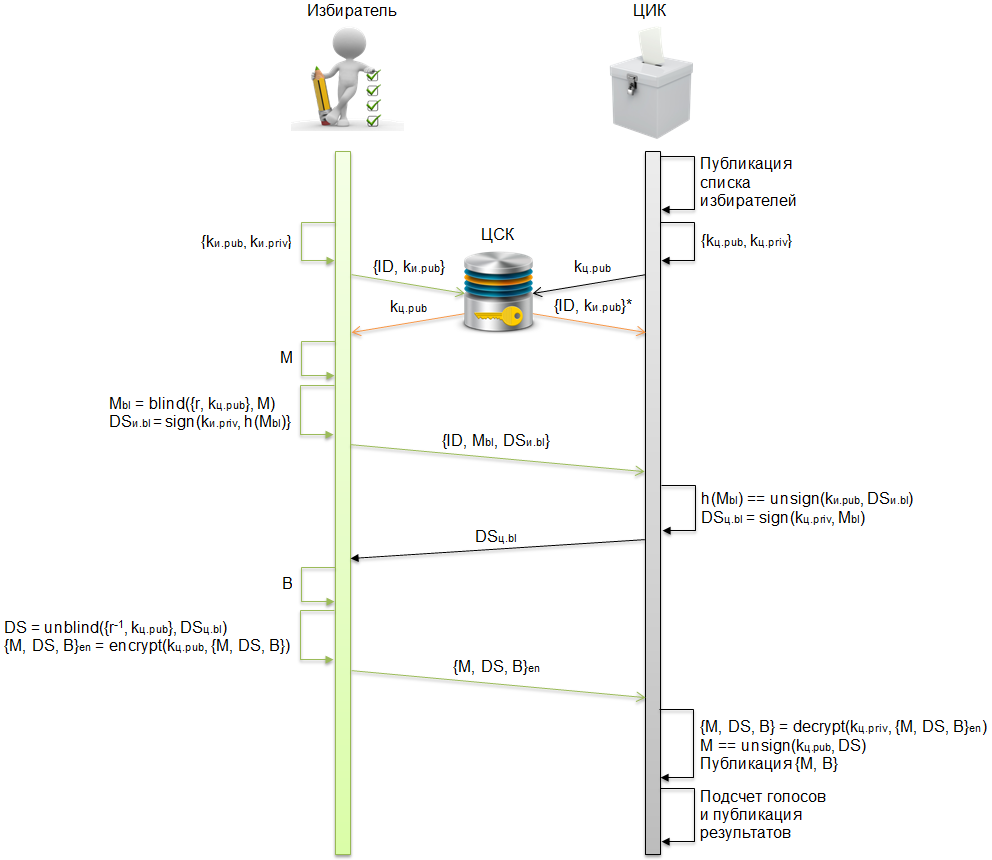


Рис. 1. Схема протокола голосования с одной Центральной комиссией на базе «слепой» подписи

Последовательность действий участников протокола:

1. Действия Центральной избирательной комиссии (ЦИК):

-Публикует список всех правомочных избирателей.

-Генерирует пару ключей для асимметричного шифрования: {k\_ц.pub, k\_ц.priv}.

-Публикует открытый ключ ЦИК k\_ц.pub через Центр сертификации ключей (ЦСК).

2. Действия избирателя:

-Генерирует собственную пару ключей {k\_и.pub, k\_и.priv} и публикует открытый ключ через ЦСК.

-Создаёт случайную метку M (уникальную для каждого голосования).

-Выполняет слепление метки M с помощью «закрывающего множителя» r, рассчитывая слепую метку Mbl = M \* r^e mod n.

-Подписывает хеш метки Mbl своей ЭЦП: DSи.bl = sign(k\_и.priv, h(Mbl)).

-Отправляет в ЦИК идентификатор (ID), слепую метку Mbl и подпись DSи.bl.

3. Действия ЦИК:

-Проверяет корректность подписи избирателя DSи.bl на Mbl с использованием открытого ключа k\_и.pub.

-Подписывает Mbl своей ЭЦП: DSц.bl = sign(k\_ц.priv, Mbl).

-Отправляет избирателю свою слепую подпись DSц.bl.

4. Действия избирателя:

-Снимает слепление с подписи DSц.bl, получая нормальную подпись ЦИК на метке: DS = unblind(DSц.bl, r^{-1}).

-Формирует бюллетень B, указывающий свой выбор.

-Шифрует совокупность {M, DS, B} с использованием k\_ц.pub.

-Анонимно отправляет зашифрованный бюллетень в ЦИК.

5. Действия ЦИК:

-Расшифровывает поступившее сообщение {M, DS, B}.

-Проверяет подпись DS к метке M с использованием k\_ц.pub.

-Публикует {M, B} на портале для контроля со стороны избирателей.

-Выполняет подсчёт голосов и публикует результаты.

Протокол описан в открытом доступе [24], откуда он и был реализован в рамках данного дипломного проекта.

3.3 Проектирование баз данных

3.3.1 Обоснование использования PostgreSQL как основной СУБД

В качестве основной системы управления базами данных (СУБД) для распределённой системы учёта голосов была выбрана PostgreSQL. Этот выбор обусловлен следующими ключевыми факторами:

1. Надёжность и устойчивость к сбоям. PostgreSQL поддерживает полную ACID-семантику, что критически важно для задач с высокой степенью ответственности, таких как выборы.

2. Поддержка расширенных типов данных. Возможность хранения бинарных данных (тип BYTEA) позволяет эффективно сохранять зашифрованные сообщения, электронные подписи, закрытые и открытые ключи.

3. Высокий уровень безопасности. PostgreSQL предоставляет широкие возможности по разграничению прав доступа и шифрованию соединений.

4. Расширяемость. Поддержка пользовательских функций, типов данных, хранимых процедур — важный фактор для внедрения криптографической логики.

5. Совместимость и кроссплатформенность. Поддерживается как на Linux, так и на Windows, что обеспечивает гибкость в развертывании.

3.3.2 Анализ типов информации, подлежащей хранению: проектирование таблиц и связей

В рамках системы реализованы **три обособленные базы данных:**

-CIK\_BD — база Центральной избирательной комиссии (ЦИК);

-CSK\_BD — база Центральной службы ключей (ЦСК);

-Сlient\_BD — отдельные базы клиентов, участвующих в голосовании.

Введение в анализ типов информации, подлежащей хранению  
Проектирование базы данных для системы электронного голосования требует тщательного анализа типов информации, которые необходимо хранить. Важнейшими аспектами являются обеспечение безопасности, конфиденциальности данных, а также эффективное управление и хранение информации о выборах, клиентах и ключах. В системе предусмотрены три основные базы данных, каждая из которых выполняет свою уникальную роль: база Центральной избирательной комиссии (ЦИК), база Центральной службы ключей (ЦСК) и база клиентов, участвующих в голосовании. Каждая из этих баз хранит специфическую информацию, важную для корректной работы системы.

Перед описанием структуры баз данных каждой из составляющих системы необходимо рассмотреть, какие типы данных следует хранить для обеспечения надёжности системы и её криптографической защиты. Эти данные включают в себя как информацию о выборах (название, дата, опции), так и данные о пользователях, их голосах, а также криптографические ключи и подписи для обеспечения анонимности и защиты данных.

3.3.3 **База данных** центральной избирательная комиссия(CIK\_BD)

База данных Центральной избирательной комиссии (ЦИК) представляет собой центральное хранилище для информации, связанной с выборами. В первую очередь, здесь хранится информация о самих выборах: название выборов, дата их проведения, количество опций, публичные и приватные ключи для шифрования и подписи, а также метки времени для отслеживания этапов выборов. Кроме того, для обеспечения безопасности голосования, важно хранить анонимизированные данные о голосах и информацию о клиентских операциях с ключами.

### Описание связей:

### 1. elections - основная таблица, описывающая выборы. Поля: id, title, option\_count, public\_key, private\_key, date\_create, date\_active, date\_completion.

**Связи**:

-id используется как внешний ключ в таблицах:

-election\_options.id\_election

-voice.id\_election

-bilutens.id\_election

2. election\_options - описывает доступные опции голосования для каждой выборной кампании. Поля: id, id\_election, option\_number, option\_text, result.

**Связь**:

-По полю id\_election используется elections.id

3. сlient - Учет зарегистрированных клиентов (избирателей). Поля: id, name, date.

**Связи**:

-id используется в voice.id\_client

4. voice – содержит анонимные данные о голосовании, включая ослепленное сообщение и подпись. Поля: id, id\_client, id\_election, key\_public\_client, ds\_client\_blind, m\_client\_blind, date.

**Связи**:

-id\_client → client.id

-id\_election → elections.id

5. bilutens - таблица с бюллетенями, прошедшими обработку. Поля: id, ds, m, b, id\_election, date.

**Связь**:

-По полю id\_election используется elections.id

### Содержание типов полей таблиц:

1. elections:

-id SERIAL PRIMARY KEY

-title TEXT

-option\_count INTEGER

-date\_create TIMESTAMP

-date\_active TIMESTAMP

-date\_ completion TIMESTAMP

-public\_key BYTEA

-private\_key BYTEA

2. election\_options:

-id SERIAL PRIMARY KEY

-id\_election INTEGER

-option\_number INTEGER

-option\_text TEXT

-result INTEGER

3. client:

-id SERIAL PRIMARY KEY

-name TEXT

-date TIMESTAMP

4. voice:

-id SERIAL PRIMARY KEY

-id\_client INTEGER

-key\_public\_client BYTEA

-id\_election INTEGER

-ds\_client\_blind BYTEA

-m\_client\_blind BYTEA

-date TIMESTAMP

5. bilutens:

-id SERIAL PRIMARY KEY

-ds BYTEA

-m BYTEA

-b INTEGER

-id\_election INTEGER

-date TIMESTAMP

Структура таблиц базы данных ЦИК должна обеспечивать надёжное хранение данных, минимизируя их избыточность и обеспечивая простоту в манипуляциях с данными, такими как подсчёт голосов и обработка выборов.

3.3.4 **Базы данных клиентов(**Сlient\_BD**)**

### Каждый клиент системы создает собственную базу данных, в которой хранится информация, касающаяся его участия в выборах. Здесь важным аспектом является хранение данных о публичных и приватных ключах, а также слепых подписях, используемых для голосования. Эта информация должна быть защищена, а доступ к ней строго контролироваться, чтобы обеспечить анонимность избирателей. Каждая клиентская база содержит информацию об опциях голосования, а также ссылки на данные ЦИК, что позволяет осуществлять проверку на стороне клиента без раскрытия его выбора.

### Описание связей:

### 1. elections - локальное представление выборов, в которых участвует клиент. Поля: id, id\_election\_cik, title, option\_count, public\_key\_cik, public\_key\_my, private\_key\_my, ds\_cik\_blind, metka, closing\_multiplier\_r, voting\_b, date.

**Связи**:

-id — первичный ключ, используется в voting\_options.election\_id

-id\_election\_cik — внешний ключ-связь с CIK\_BD.elections.id (сопоставление с выборами на сервере ЦИК)

2. voting\_options - опции голосования, доступные клиенту. Поля: id, election\_id, option\_number, option\_text, result.

**Связь**:

-election\_id → elections.id (в клиентской базе)

### Содержание типов полей таблиц:

1. elections:

-id SERIAL PRIMARY KEY

-id\_election\_cik INTEGER

-title TEXT

-option\_count INTEGER

-public\_key\_cik BYTEA

-public\_key\_my BYTEA

-private\_key\_my BYTEA

-date TEXT

-metka BYTEA

-сlosing\_multiplier\_r INTEGER

-voting\_b INTEGER

-ds\_cik\_blind BYTEA

2. voting\_options:

-id SERIAL PRIMARY KEY

-election\_id INTEGER

-option\_number INTEGER

-option\_text TEXT

-result INTEGER

3.3.5. База данных центр сертификации ключей(CSK\_BD)

Центр сертификации ключей (ЦСК) играет ключевую роль в обеспечении безопасности системы, так как он управляет публичными ключами, используемыми для шифрования и аутентификации голосов. В базе данных ЦСК хранится информация о зарегистрированных ключах, а также данные о выборах, для которых эти ключи были выданы. База должна обеспечивать связь с выборными данными ЦИК, а также с клиентскими данными для проверки подлинности ключей.

1. election*-* представляет выборы, зарегистрированные в ЦСК. Поля: id, title, id\_election\_cik, sending, date.

**Связи**:

-id\_election\_cik → CIK\_BD.elections.id

-id может использоваться для фильтрации соответствующих ключей

2. key\_public - публичные ключи клиентов, участвующих в голосованиях. Поля: id, id\_client, name, key\_pub, id\_election, date/

**Связи**:

-id\_election → election.id

-id\_client — потенциальная связь с CIK\_BD.client.id (но формально это необязательно, может быть просто идентификатор участника)

### Содержание полей таблиц и связи:

1. election:

-id SERIAL PRIMARY KEY

-title TEXT

-id\_election\_cik INT

-sending BOOLEAN

-date TIMESTAMP

2. key\_public:

-id SERIAL PRIMARY KEY

-id\_client INT

-name TEXT

-key\_pub BYTEA

-id\_election INT

-date TIMESTAMP

3.3.6 **Общий вывод по проектированию структуры БД**

Разработка базы данных для системы электронного голосования требует комплексного подхода, который обеспечит не только правильное хранение информации, но и её безопасность, конфиденциальность и целостность. Проектирование базы данных для ЦИК, ЦСК и клиентских баз основано на принципах минимизации избыточности, разделении ролей и надёжной защите данных. Каждая база данных выполняет свою задачу и имеет чёткие связи между таблицами, что способствует прослеживаемости данных, поддержанию целостности и предотвращению ошибок в процессе голосования.

Таким образом, структура БД учитывает все аспекты:

-Разделение ролей и данных;

-Минимизация избыточности;

-Сохранение всех криптографических данных в бинарном формате;

-Обеспечение прослеживаемости и связности данных через внешние ключи.

3.4 Система обмена сообщениями

В рамках разработанной распределённой системы электронного голосования была спроектирована система обмена сообщениями, обеспечивающая надёжную и формализованную коммуникацию между ключевыми компонентами: Центральной избирательной комиссией (ЦИК), Центром сбора ключей (ЦСК) и клиентами (пользователями). Обмен сообщениями реализован посредством текстовых файлов формата .txt, содержащих сериализованные данные в формате JSON, что обеспечивает совместимость, читаемость и простоту обработки.

Каждое сообщение имеет уникальное имя, содержащее:

-Информативную приставку (например, reg, vote, key и др.);

-Идентификатор сущности (например, title, name, cik, csk);

-Случайный salt, повышающий уникальность сообщений и защищающий от подмены;

-Расширение .txt.

Файлы сохраняются в заранее определённые директории, сгруппированные по типу компонента и направлению взаимодействия (например, входящие от ЦИК к пользователю, от пользователя к ЦИК, от ЦСК к ЦИК и т.д.). Это позволяет легко фильтровать, отслеживать и валидировать сообщения в процессе обмена.

Для каждого сообщения была разработана своя структура, включающая необходимые атрибуты, согласованные между сторонами. Ниже приводится описание каждого типа сообщения, его предназначения, состава и логики валидации.

1. Регистрация пользователя от ЦИК для пользователя

Файл: [name]\_start\_client\_[salt.random].txt

Назначение: Инициализация клиента и его учётная запись в системе.

Это сообщение используется ЦИК для генерации начального профиля пользователя. В него входят:

-name — имя пользователя, необходимое для идентификации;

-id\_client — уникальный идентификатор, назначенный системой;

-date — дата создания, фиксирующая момент регистрации.

2. Добавление/обновление ключа и голосования для ЦСК от ЦИК

Файл: cik\_reg\_csk\_[salt.random].txt

Назначение: Передача от ЦИК в ЦСК информации о новом голосовании и клиентских данных.

Структура включает:

-title — наименование голосования;

-id\_election — уникальный идентификатор голосования;

-date — дата отправки данных;

-key\_csk — открытый ключ для ЦСК;

-count\_name — список клиентов с полями name и id\_client.

Это сообщение необходимо для корректного формирования структуры данных и ключей в ЦСК.

3. Приглашение на голосование от ЦИК для пользователя

Файл: [title]\_invitation\_to\_vote\_[salt.random].txt

Назначение: Информирование пользователя о голосовании и предоставление списка опций.

Содержит:

-title, id\_election, date — основные метаданные голосования;

-count\_number — перечень вариантов голосования, содержащих option\_number и option\_text.

Это сообщение необходимо для отображения опроса пользователю.

4. Слепая подпись для голосования от ЦИК для пользователя

Файл: [name]\_blind\_signature\_[salt.random].txt

Назначение: Подтверждение слепой подписи от ЦИК.

Поля:

-id\_client — идентификатор клиента;

-date — дата подписи;

-id\_election — ID голосования;

-ds\_cik\_blind — цифровая слепая подпись.

Эта подпись используется для валидации анонимного голосования.

5. Результат голосования от ЦИК для пользователя

Файл: [title]\_result\_[salt.random].txt

Назначение: Представление итогов голосования.

Поля:

-title, id\_election, date — идентификация голосования;

-option\_count — результат по каждому варианту: option\_number, option\_text, option\_result;

-count\_mb — блок меток: label\_m и voting\_b, подтверждающие корректность процедуры.

Это сообщение закрывает цикл голосования для пользователя, представляя итог.

6. Ответ от ЦСК для ЦИК

Файл: cik\_key\_csk\_[salt.random].txt

Назначение: Обратная передача информации о публичных ключах.

Поля:

-id\_election, date — информация об ID голосования;

-count\_key — количество ключей;

-data\_key — массив с парами id\_client и key.

Это сообщение обеспечивает проверку целостности ключевой инфраструктуры.

7. Скрытая метка от пользователя для ЦИК

Файл: [name]\_hidden\_label\_[salt.random].txt

Назначение: Передача слепой метки от клиента для подготовки голосования.

Содержит:

-id\_client, id\_election, date;

-m\_blind — слепая метка;

-ds\_client\_blind — её подпись.

Эта информация используется ЦИК для выпуска биллютеня и его валидации.

8. Голос от пользователя для ЦИК

Файл: voting\_[salt.random].txt

Назначение: Передача зашифрованного голоса.

Поля:

-id\_election, date;

-Вложенный блок: m, ds, voting\_b — анонимизированный голос;

-public\_key\_cik — открытый ключ ЦИК.

Сообщение реализует передачу зашифрованного и подписанного выбора.

9. Ключ для голосования для ЦСК от пользователя

Файл: [name]\_reg\_csk\_[salt.random].txt

Назначение: Регистрация пользователем ключа в ЦСК.

Содержит:

-name, id\_client, date;

-id\_election, key.

Позволяет ЦСК обработать ключ, привязанный к конкретному пользователю и голосованию.

10. Ключ для голосования от ЦСК для пользователя

Файл: [name]\_key\_csk\_[salt.random].txt

Назначение: Обратная передача ключа от ЦСК пользователю.

Содержит те же поля, что и при регистрации:

-name, id\_client, date, id\_election, key.

Это сообщение подтверждает успешную обработку и передачу ключа.

Для эффективного функционирования системы электронного голосования важно, чтобы структура обмена сообщениями была организована так, чтобы обеспечивать безопасность, целостность и анонимность данных, передаваемых между компонентами. Процесс обмена осуществляется с использованием текстовых файлов формата .txt, где данные сериализуются в JSON. Это позволяет гарантировать совместимость между различными компонентами системы и удобство их обработки. Каждый файл сообщения имеет уникальное имя, содержащее метки, которые обеспечивают безопасность данных и предотвращают их подмену. В результате такой системы обмена, сообщения, связанные с регистрацией, ключами, голосованием и результатами, могут быть эффективно отслеживаемы, валидируемы и обработаны соответствующими системами, обеспечивая прозрачность и защиту всей процедуры электронного голосования.

3.5 Первоначальная инициализация приложений

Для обеспечения безопасной и корректной работы всех компонентов системы голосования предусмотрен механизм первоначальной инициализации приложений. Эта процедура проводится при первом запуске каждого из компонентов: Центральной избирательной комиссии (ЦИК), Центрального сервера ключей (ЦСК) и клиентского приложения избирателя. Инициализация включает в себя создание необходимых структур каталогов, генерацию и защиту файла кэша, содержащего критически важную информацию, а также базовую настройку подключения к базе данных.

3.5.1 Механизм создания кэш-файлов при первом запуске

Основной задачей кэш-файла является сохранение конфиденциальной информации, необходимой для дальнейшей работы приложения. В частности, в него сохраняются:

-Пароль для подключения к базе данных,

-Имя и идентификатор пользователя (для клиента),

Другие настройки, используемые в рамках работы приложения.

При первом запуске приложение проверяет наличие кэш-файла (kesh\_\*.txt). Если файл найден, пользователю предлагается ввести пароль для расшифровки содержимого. При успешной верификации происходит доступ к хранимым данным, в противном случае — пользователю дается несколько попыток ввода пароля. Если же кэш-файл отсутствует, запускается процедура активации, включающая:

-Выбор или подтверждение данных пользователя;

-Установку пароля;

-Проверку пароля базы данных;

-Создание необходимых директорий для сообщений;

-Генерацию структуры базы данных и таблиц;

-Шифрование данных и сохранение в кэш-файл.

3.5.2 Применение симметричного шифрования для защиты данных

В целях безопасности данные, сохраняемые в кэш, шифруются с использованием алгоритма симметричного шифрования (например, AES). Ключ шифрования генерируется из введенного пользователем пароля. Таким образом, без знания пароля доступ к информации, хранящейся в кэше, невозможен. Это особенно важно, поскольку кэш может содержать пароль к PostgreSQL или другие чувствительные данные, необходимые для установления защищенного соединения с базой данных.

3.5.3 Алгоритмы создания, шифрования и проверки хешей

Процедура шифрования реализована следующим образом:

-Из пароля с помощью функции ключевыведения (например, PBKDF2 или SHA256) создается ключ.

-Данные сериализуются и шифруются.

-Защищенные данные сохраняются в файл на диске.

-При последующем запуске приложения используется тот же пароль для расшифровки содержимого.

Это обеспечивает не только сохранность, но и возможность верификации правильности пароля без дополнительного хранилища открытых хешей — проверка осуществляется попыткой корректной расшифровки содержимого.

3.5.4 Вывод

Такая архитектура инициализации позволяет одновременно обеспечить:

-безопасность чувствительной информации;

-защиту от несанкционированного доступа;

-упрощение последующих запусков приложения;

-строгую верификацию правомочности запуска (в случае клиента — только при наличии регистрационного файла).

3.6 Разработка программных модулей

Все три приложения (клиентское, ЦСК и ЦИК) построены по единому принципу: они используют терминальный интерфейс для взаимодействия с пользователем, работают с базой данных и файловой системой, а также реализуют криптографические методы для обеспечения безопасности голосования. Основные компоненты каждого приложения:

1. Терминальный интерфейс: иерархическое меню с подменю для навигации.

2. Работа с БД: подключение, выполнение запросов, закрытие соединения.

3. Обработка файлов: чтение и создание файлов с ключами, метками, бюллетенями.

4. Криптографические методы: генерация ключей, слепая подпись, шифрование.

5. Аутентификация: проверка пароля приложения и чтение данных КЭШа

Каждое приложение выполняет свою роль в протоколе:

1. Клиентское приложение: участие в голосовании, отправка ключей и меток.

2. Приложение ЦСК: сбор и распределение ключей между участниками.

3. Приложение ЦИК: управление голосованиями, проверка подписей, подсчет результатов.

Клиентское приложение

Структура меню:

1. Главное меню:

-Чтение сообщений.

-Управление голосованиями (просмотр, удаление).

-Выход.

2. Подменю:

-При обработке сообщений: выбор режима (автоматический/ручной), действия с файлами (принять/отклонить/отложить).

-В управлении голосованиями: просмотр статусов, удаление записей.

Пример сценария:

1. Пользователь выбирает "Чтение сообщений".

2. Приложение сканирует папку read\_message, сортирует файлы по типам.

3. Для каждого файла запрашивается действие (например, принять голосование → сгенерировать ключ → отправить ЦСК).

4. Данные сохраняются в БД, файлы перемещаются/удаляются.

Приложение ЦСК

Структура меню:

1. Главное меню:

-Чтение сообщений (ключи от клиентов).

-Управление базой ключей.

-Выход.

2. Подменю базы ключей:

-Просмотр голосований, добавление/удаление участников.

-Запуск рассылки ключей.

Пример сценария:

1. ЦСК получает файл cik\_reg\_csk\_\*.txt с данными голосования.

2. Добавляет запись в БД (election), сохраняет ключи ЦИК.

3. Рассылает ключи клиентам через файлы в sent\_message.

Приложение ЦИК

Структура меню:

1. Главное меню:

-Чтение сообщений (метки, бюллетени).

-Управление пользователями.

-Управление голосованиями (создание, запуск, завершение).

-Выход.

2. Подменю голосований:

-Создание/редактирование голосований.

-Просмотр активных/завершенных голосований.

-Подсчет результатов.

Пример сценария:

1. ЦИК получает файл \_hidden\_label\_\*.txt с слепой меткой.

2. Проверяет подпись клиента, подписывает метку, отправляет ответ.

3. При получении бюллетеня (voting\_\*.txt) расшифровывает его, проверяет метку, сохраняет голос.

4. После завершения голосования считает результаты и рассылает файлы \_result\_\*.txt.

Программные модули реализуют протокол голосования с разделением ролей:

-Клиенты участвуют в голосовании, не раскрывая свой выбор.

-ЦСК обеспечивает распределение ключей.

-ЦИК проверяет подлинность данных и подсчитывает результаты.

4 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

4.1 Разработанные методы

4.1.1 Реализован протокол в программной среде

Для реализации вышеописанного протокола были использованны следующие бибиотеки:

1. PyCryptodome (Crypto) **-** библиотека, реализующая множество криптографических алгоритмов. Это "выпадающий" модуль для PyCrypto, улучшенный и поддерживаемый.

-Crypto.PublicKey.RSA – генерация и работа с RSA-ключами.

-Crypto.Random.get\_random\_bytes – получение криптографически стойкой случайной строки байтов.

-Crypto.Cipher.PKCS1\_OAEP – RSA-шифрование с padding PKCS#1 OAEP (защита от атак на шифротекст).

-Crypto.Signature.pkcs1\_15 – алгоритм подписи по стандарту PKCS#1 v1.5 (RSA).

-Crypto.Hash.SHA256 – криптографическая хеш-функция SHA-256.

-Crypto.Util.number – утилиты для работы с числами, например:

-getPrime – генерация простого числа;

-inverse – нахождение обратного элемента по модулю;

-bytes\_to\_long, long\_to\_bytes – преобразование между байтами и целыми числами (важно для RSA).

2. json - модуль для кодирования/декодирования JSON – используется для упаковки/распаковки бюллетеня (структуры, содержащей M, DS, B).

3. psycopg2 - библиотека для работы с PostgreSQL из Python.

-psycopg2.Binary() – используется для корректной вставки бинарных данных (BYTEA) в таблицы PostgreSQL.

Для реализации вышеописанного протокола были разработаны следующие методы и выбраны соответствующие структуры данных:

**1. Генерация RSA ключей**

def generate\_rsa\_key\_pair(bits=2048):

key = RSA.generate(bits)

return key.publickey().export\_key(format='DER'), key.export\_key(format='DER')

Описание: Этот метод генерирует пару RSA-ключей — публичный и приватный. Он использует библиотеку Crypto.PublicKey.RSA для создания ключей. Вы можете задать длину ключа (в битах), по умолчанию — 2048 бит, что является стандартом для безопасности.

**-**key.publickey().export\_key(format='DER'): Экспортирует публичный ключ в формате DER (формат бинарного представления).

**-**key.export\_key(format='DER'): Экспортирует приватный ключ в формате DER.

Зачем это нужно**:** Публичный ключ используется для шифрования данных, а приватный — для расшифровки и подписи.

**2. Создание метки M**

def generate\_m():

return get\_random\_bytes(32) # 256-битная случайная строка

Описание: Этот метод генерирует случайную метку M, которая используется как уникальный идентификатор для каждого избирателя. Она представляет собой 256-битную строку, которая будет использоваться в процессе слепой подписи.

**-**get\_random\_bytes(32**)**: Генерирует 32 случайных байта (256 бит), используя криптографически стойкий генератор случайных чисел.

Зачем это нужно: Метка M является основным элементом в протоколе, и она будет скрыта и подписана различными участниками протокола для обеспечения анонимности избирателя.

**3. Слепление метки (M) в Mbl**

def blind\_message(M: bytes, r: int, pub\_key\_der: bytes) -> bytes:

pub\_key = import\_rsa\_key(pub\_key\_der)

M\_int = bytes\_to\_long(M)

Mbl = (M\_int \* pow(r, pub\_key.e, pub\_key.n)) % pub\_key.n

return long\_to\_bytes(Mbl)

Описание: Этот метод реализует процесс слепления метки M, с целью скрыть её содержимое от других участников, включая Центральную Избирательную Комиссию (ЦИК). Используется закрывающее множитель r, который применяется для создания "слепой" версии метки Mbl.

-bytes\_to\_long(M): Преобразует байты метки M в целое число.

-pow(r, pub\_key.e, pub\_key.n): Использует публичный ключ для возведения r в степень e по модулю n (где e и n — это компоненты публичного ключа ЦИК).

-Mbl: Это и есть "слепая" метка, которую избиратель отправит ЦИК для дальнейшего подписания.

Зачем это нужно: Слепление метки обеспечивает анонимность: ЦИК не видит оригинальную метку M, и, следовательно, не может привязать её к конкретному избирателю.

**4. Подпись слепой метки (Mbl)**

def sign\_message\_blinded(priv\_key\_der: bytes, message\_bytes: bytes) -> bytes:

key = import\_rsa\_key(priv\_key\_der)

h = SHA256.new(message\_bytes)

return pkcs1\_15.new(key).sign(h)

Описание**:** Этот метод подписывает "слепую" метку с использованием приватного ключа ЦИК. Используется алгоритм RSA с SHA256 хешированием для создания цифровой подписи.

**-**SHA256.new(message\_bytes): Хеширует сообщение (в данном случае, слепленную метку).

-pkcs1\_15.new(key).sign(h): Создает цифровую подпись с использованием приватного ключа ЦИК по хешу сообщения.

Зачем это нужно**:** ЦИК подписывает слепую метку, чтобы доказать, что она была признана действительной и подтверждена Центральной Комиссией.

**5. Проверка подписи**

def verify\_signature(pub\_key\_der: bytes, signature: bytes, message\_bytes: bytes) -> bool:

key = import\_rsa\_key(pub\_key\_der)

h = SHA256.new(message\_bytes)

try:

pkcs1\_15.new(key).verify(h, signature)

return True

except (ValueError, TypeError):

return False

Описание: Этот метод проверяет цифровую подпись на предмет её подлинности. Используется публичный ключ избирателя для проверки подписи, которая была произведена его приватным ключом.

pkcs1\_15.new(key).verify(h, signature): Проверка подписи с использованием публичного ключа.

Зачем это нужно**:** Этот метод позволяет убедиться, что подпись действительна и что сообщение действительно было подписано указанным избирателем.

**6. Снятие слепоты подписи**

def unblind\_signature(blinded\_signature: bytes, r: int, pub\_key\_der: bytes) -> bytes:

key = import\_rsa\_key(pub\_key\_der)

r\_inv = inverse(r, key.n)

sig\_int = bytes\_to\_long(blinded\_signature)

return long\_to\_bytes((sig\_int \* r\_inv) % key.n)

Описание: Этот метод снимает слепоту с подписи, предоставленной ЦИК. Используется закрывающее множитель r, чтобы вернуть оригинальную подпись.

**-**inverse(r, key.n): Находит мультипликативную инверсию числа r по модулю n.

-sig\_int \* r\_inv % key.n: Вычисляет исходную подпись после снятия слепоты.

Зачем это нужно**:** Снятие слепоты позволяет избирателю получить реальную подпись от ЦИК, которая затем будет использована для дальнейшего шифрования бюллетеня.

**7. Шифрование бюллетеня**

def encrypt\_ballot(pub\_key\_der: bytes, M: bytes, DS: bytes, B: int) -> bytes:

key = import\_rsa\_key(pub\_key\_der)

cipher = PKCS1\_OAEP.new(key)

ballot = json.dumps({

"M": M.hex(),

"DS": DS.hex(),

"B": B

}).encode()

return cipher.encrypt(ballot)

Описание: Этот метод шифрует бюллетень, который включает метку M, подпись ЦИК (DS) и выбор избирателя B (например, номер выбранной партии).

-PKCS1\_OAEP.new(key): Создает объект шифрования с использованием алгоритма RSA и режима OAEP (Optimal Asymmetric Encryption Padding).

-json.dumps(): Преобразует данные в формат JSON для удобства передачи.

-cipher.encrypt(): Шифрует данные с использованием публичного ключа ЦИК.

Зачем это нужно: Шифрование бюллетеня гарантирует, что никто не сможет изменить данные до того, как они будут расшифрованы ЦИК.

**8. Расшифровка бюллетеня**

def decrypt\_ballot(priv\_key\_der: bytes, encrypted: bytes):

key = import\_rsa\_key(priv\_key\_der)

cipher = PKCS1\_OAEP.new(key)

ballot = json.loads(cipher.decrypt(encrypted).decode())

return {

"M": bytes.fromhex(ballot["M"]),

"DS": bytes.fromhex(ballot["DS"]),

"B": ballot["B"]

}

Описание: Этот метод расшифровывает зашифрованный бюллетень, чтобы извлечь оригинальные данные (метку M, подпись ЦИК DS и выбор избирателя B).

-cipher.decrypt(): Дешифрует данные с использованием приватного ключа ЦИК.

-json.loads(): Преобразует JSON-строку обратно в Python-словарь.

Зачем это нужно: ЦИК расшифровывает бюллетени для проверки подписей и подсчета голосов.

**9. Конвертация данных для PostgreSQL**

def bytes\_to\_psql\_bytea(data: bytes):

"""Готовит данные для вставки в поле BYTEA PostgreSQL."""

return psycopg2.Binary(data)

def psql\_bytea\_to\_bytes(bytea\_field) -> bytes:

"""При извлечении BYTEA из базы он уже bytes, но эта функция — для ясности."""

return bytes(bytea\_field)

PostgreSQL использует специальный тип данных BYTEA для хранения бинарной информации (например, ключей, хэшей, подписей и шифрованных сообщений). Однако при работе с библиотеками криптографии (например, PyCryptodome) данные часто представлены как байты (bytes в Python). Чтобы корректно сохранять и извлекать такие данные из базы данных, нужна правильная конвертация.

Данные методы используют стандартные криптографические алгоритмы:

-RSA для шифрования и подписи;

-SHA-256 для хеширования сообщений;

-Слепую подпись по схеме Chaum RSA Blind Signature [32].

Для сериализации и десериализации сложных структур используется модуль json. При шифровании бюллетеня применяется гибридный подход — объект сериализуется в JSON и затем шифруется RSA-алгоритмом (PKCS1\_OAEP).

4.1.2 Разработаны методы для автоматической инициализации структуры баз данных

Инициализация БД реализована через Python с использованием библиотеки psycopg2. В функции включены:

-Подключение к серверу PostgreSQL через дефолтную БД postgres;

-Попытка создания целевой БД (CREATE DATABASE);

-Подключение к вновь созданной БД и создание таблиц с проверкой на существование (CREATE TABLE IF NOT EXISTS).

Для каждой таблицы логика одинакова:

-Уникальный идентификатор (SERIAL PRIMARY KEY);

-Необходимые поля с типами TEXT, INTEGER, TIMESTAMP, BYTEA;

-В случае бинарных данных — тип BYTEA, обеспечивающий сохранение ключей, подписей и сообщений.

Также реализованы функции connect\_to\_db, connect\_to\_client\_db, connect\_to\_cik\_db, позволяющие переиспользовать логику подключения, повышая модульность кода.

Использование конструкции try...except psycopg2.errors.DuplicateDatabase предотвращает ошибки при повторных инициализациях — это важно при автоматизированном развёртывании в продакшене или CI/CD.

4.1.3 Метод реализующий чтение и создания файлов(сообщений)

Методы для создания, записи, чтения и валидации сообщений в распределённой системе обмена сообщениями являются однотипными для всех типов сообщений. Это позволяет стандартизировать процесс обмена и обеспечить универсальность работы с различными типами данных, передаваемых между компонентами системы.

Каждое сообщение создаётся с использованием данных, полученных от отправителя, и записывается в текстовый файл в формате JSON. После того как сообщение записано, оно может быть прочитано и обработано получателем с помощью метода чтения, который извлекает и валидирует информацию.

Пример для создания и чтения файла с результатами:

Метод создания сообщения:

Например, для отправки сообщения о результатах голосования:

def created\_res\_client\_messgae(file\_path, title, id\_election, option\_count, data\_nember\_text\_result, count\_mb, data\_mb):

data = {

"title": title,

"id\_election": id\_election,

"date": str(datetime.now()),

"option\_count": option\_count,

"data\_nember\_text\_result": data\_nember\_text\_result,

"count\_mb": count\_mb,

"data\_mb": data\_mb

}

try:

with open(file\_path, 'w') as f:

json.dump(data, f)

return True

except Exception as e:

print(f"Error creating message: {e}")

return False

Здесь используется стандартная структура данных, включающая заголовок, идентификатор голосования, дату и результаты голосования. Данные сериализуются в формат JSON и сохраняются в файл.

Метод чтения сообщения:

Для чтения сообщения о результатах голосования из файла, можно использовать метод, аналогичный приведённому ниже:

def read\_res\_cik\_messge(file\_path):

try:

with open(file\_path, 'r') as f:

data = json.load(f)

return (

data["title"], data["id\_election"], data["date"],

data["option\_count"], data["data\_nember\_text\_result"],

data["count\_mb"], data["data\_mb"]

)

except Exception as e:

print(f"Error reading message: {e}")

return None

Здесь метод открывает файл, парсит его содержимое, а затем возвращает необходимые данные, такие как название голосования, идентификатор, дата, а также результаты голосования.

Эти методы являются примером универсального подхода для работы с любыми типами сообщений, используемыми в системе. Все методы обработки сообщений реализуют схожие процессы:

-Создание и запись данных в файл.

-Чтение данных из файла.

-Валидация содержимого.

Этот подход позволяет эффективно управлять обменом сообщениями в рамках системы, обеспечивая корректность и безопасность данных.

Все методы учитывают возможность ошибок (например, недоступность диска или некорректный формат) и возвращают диагностическую информацию в случае неудачи.

4.1.4 Метод первоначальной инициализация приложений чтения кэша

В рамках работы были реализованы три метода инициализации для компонентов системы голосования: для Центральной избирательной комиссии (ЦИК), Центрального сервера ключей (ЦСК) и клиентского приложения. Все три метода имеют схожие принципы работы, но с небольшими различиями, обусловленными особенностями хранимой информации и проверок.

Для компонента ЦИК была разработана процедура, которая выполняет проверку наличия кэш-файла и, при его отсутствии, активирует систему. Приложение проверяет наличие файла kesh\_cik.txt. Если файл существует, происходит попытка расшифровки с использованием введенного пароля. В случае успешной верификации происходит доступ к сохраненным данным. Если пароль введен неверно трижды, приложение завершает работу. Если файл не найден, то запускается процесс активации, включающий создание базы данных и установку пароля для доступа к PostgreSQL, а также запись зашифрованной информации в новый кэш-файл.

def handle\_password():

# Проверка, существует ли файл с данными

clear\_console()

if os.path.exists("kesh\_cik.txt"):

for \_ in range(3):

clear\_console()

password = input("Введите пароль для входа в приложение: ")

try:

# Попытка дешифровать данные

with open("kesh\_cik.txt", "r") as f:

encrypted\_data = f.read()

data = decrypt\_data(password, encrypted\_data)

return data

except Exception as e:

print("Неверный пароль. Попробуйте снова.")

print("Пароль введен неправильно 3 раза.")

wait\_for\_user()

return None

else:

clear\_console()

print("Приложение не активировано")

password = input("Придумайте пароль для входа в приложение: ")

data = []

while True:

chice=input("Введите пароль для работы с psql:")

if not check\_db\_password(chice):

print("Пароль не подходит для подключения к psql, введите корректный пароль")

data.append(chice)

create\_database\_and\_tables(data[0])

# Шифрование и запись в файл

encrypted\_data = encrypt\_data(password, data)

with open("kesh\_cik.txt", "w") as f:

f.write(encrypted\_data)

nested\_folder\_path = os.path.join("sent\_message", "CIK")

os.makedirs(nested\_folder\_path, exist\_ok=True)

nested\_folder\_path = os.path.join("read\_message")

os.makedirs(nested\_folder\_path, exist\_ok=True)

print("Приложение активировано")

wait\_for\_user()

return data

Для клиентского приложения процесс активации немного сложнее. Кроме проверки кэш-файла kesh\_client.txt, приложение проверяет наличие регистрационного файла, выданного ЦИК. Если файл найден, пользователю предлагается активировать приложение, и только после этого осуществляется создание и защита данных. Важно, что в этом случае клиенту необходимо ввести пароль для шифрования данных, а также пароль для подключения к базе данных PostgreSQL.

def handle\_password():

clear\_console()

# Проверка, существует ли файл с данными

if os.path.exists("kesh\_client.txt"):

for \_ in range(3):

clear\_console()

password = input("Введите пароль: ")

try:

# Попытка дешифровать данные

with open("kesh\_client.txt", "r") as f:

encrypted\_data = f.read()

data = decrypt\_data(password, encrypted\_data)

return data

except Exception as e:

print("Неверный пароль. Попробуйте снова.")

print("Пароль введен неправильно 3 раза.")

wait\_for\_user()

return None

else:

clear\_console()

input("Приложение не активировано. Для активации убедитесь, что у вас есть регистрационный файл в диретории приложения")

wait\_for\_user()

filename = None # по умолчанию, если не найден

for file in os.listdir('.'):

if fnmatch.fnmatch(file, '\*\_start\_client\_\*.txt'):

filename = file

break # если нужен только первый найденный файл

if not filename:

clear\_console()

print("Регистрационный файл не найден")

wait\_for\_user()

return None

name, id\_client, date = read\_start\_client\_message(filename)

while True:

print("Файл регистрации найден для ",id\_client ," " ,name ," выпуск файла:"date)

choice = input("Активировать приложение этим файлом?(1-yes/2-no)")

if choice == "1":

clear\_console()

password = input("Придумайте пароль для входа в приложение: ")

data = []

while True:

chice=input("Введите пароль для работы с psql:")

if not check\_db\_password(chice):

print("Пароль не подходит для подключения к psql, введите корректный пароль")

data.append(chice)

data.append(name)

data.append(id\_client)

create\_client\_database\_and\_tables(data[0], data[1])

# Шифрование и запись в файл

encrypted\_data = encrypt\_data(password, data)

with open("kesh\_client.txt", "w") as f:

f.write(encrypted\_data)

nested\_folder\_path = os.path.join("sent\_message", "CSK")

os.makedirs(nested\_folder\_path, exist\_ok=True)

nested\_folder\_path = os.path.join("sent\_message", "CIK")

os.makedirs(nested\_folder\_path, exist\_ok=True)

nested\_folder\_path = os.path.join("read\_message")

os.makedirs(nested\_folder\_path, exist\_ok=True)

print("Приложение активировано")

wait\_for\_user()

return data

elif choice == "2":

print("Приложение осталось не активировано")

wait\_for\_user()

return None

else:

print("Некорректный ввод")

wait\_for\_user()

Реализация методов инициализации для ЦИК, ЦСК и клиентского приложения показывает гибкость и универсальность предложенной архитектуры. Все три метода используют схожие подходы, включая проверку наличия кэш-файлов и использование пароля для шифрования и дешифрования данных. Однако для ЦСК и клиентского приложения добавлены дополнительные шаги, такие как подтверждение регистрации и учет специфической информации, необходимой для каждого компонента.

Такая унифицированная схема инициализации позволяет:

-Обеспечить высокий уровень безопасности, шифруя данные и проверяя пароли;

-Снизить вероятность несанкционированного доступа;

-Упростить дальнейшее масштабирование и модификации системы.

Таким образом, все три компонента — ЦИК, ЦСК и клиентское приложение — могут работать в едином безопасном и эффективном контексте.

4.1.5 Реализация программных модулей

Общие компоненты

1. Аутентификация:

-Все приложения используют handle\_password() для входа. Пароль проверяется перед доступом к функционалу.

2. Работа с файлами:

-Чтение: read\_\*\_message (например, read\_csk\_message).

-Создание: create\_\*\_message (например, create\_blind\_cik\_message).

-Файлы хранятся в структурированных папках (read\_message, sent\_message).

3. Безопасность:

-Генерация ключей RSA (generate\_rsa\_key\_pair).

-Слепая подпись (blind\_message, sign\_message\_blinded).

-Шифрование бюллетеней (encrypt\_ballot).

4. Утилиты:

-clear\_console: очистка терминала.

-wait\_for\_user: пауза для удобства пользователя.

-gen\_salt: генерация уникальных имен файлов.

Клиентское приложение:

-Работа с БД:

-Подключение к БД при запуске (connect\_to\_db).

-Запросы для:

-Регистрации голосований (INSERT INTO elections).

-Управления вариантами ответов (voting\_options).

-Обновления статусов (UPDATE elections).

-Закрытие соединения после завершения операций.

-Ключевые методы:

-process\_invitation\_to\_vote\_files: обработка приглашений, генерация ключей RSA.

-process\_key\_csk\_files: работа с ключами ЦСК, отправка слепых меток.

-process\_blind\_signature\_files: голосование с использованием слепой подписи.

-process\_result\_files: проверка результатов голосования.

Приложение ЦСК:

-Работа с БД:

-Хранение ключей клиентов (key\_public).

-Управление голосованиями (election).

-Фиксация изменений после критических операций (conn.commit()).

-Ключевые методы:

-read\_message: обработка файлов \_reg\_csk\_\*.txt и \_key\_csk\_\*.txt.

-menu\_key: управление ключами, формирование файлов для рассылки.

-create\_client\_message: создание файлов с ключами для клиентов.

Приложение ЦИК:

-Работа с БД:

-Хранение голосований (elections), вариантов ответов (election\_options).

-Проверка подлинности меток (bilutens).

-Обновление результатов (UPDATE election\_options SET result).

-Ключевые методы:

-process\_hidden\_label\_files: проверка слепых меток, генерация подписей.

-process\_voting\_files: расшифровка бюллетеней, проверка подписей.

-end\_election: завершение голосования, рассылка результатов.

-created\_election: создание нового голосования с ключами RSA.

4.2 Разработана криптографическая система для голосования

4.2.1 Общая информация

Криптографическая система реализует протокол анонимного и безопасного голосования, основанный на технологии слепой подписи и асимметричном шифровании (RSA). Все взаимодействие между сторонами осуществляется через файловую систему (передача сообщений в виде структурированных текстовых файлов) и сопровождается строгой аутентификацией пользователей.

Система построена на взаимодействии трех сторон:

-ЦИК (Центральная избирательная комиссия) — организатор и арбитр голосования.

-ЦСК (Центр Сбора Ключей) — посредник, обеспечивающий анонимность и передачу ключей.

-Клиенты (Избиратели) — участники голосования, передающие зашифрованные голоса.

4.2.2 Этапы криптографического протокола

**1. Авторизация всех приложений:**

-Все приложения используют метод handle\_password() для доступа к функционалу.

-ЦИК проходит авторизацию первым и создаёт голосование.

-Клиент может авторизоваться только после получения **файла регистрации** от ЦИК: title\_invitation\_to\_vote\_[salt].txt.

**2. Создание голосования (ЦИК)**

-После авторизации ЦИК создаёт новое голосование.

-Автоматически формируются:

-Файл для ЦСК: cik\_reg\_csk\_[salt].txt.

-Приглашения для клиентов: title\_invitation\_to\_vote\_[salt].txt.

-ЦСК получает описание голосования и может начинать сбор ключей.

**3. Сбор клиентских ключей (ЦСК)**

-ЦСК принимает **регистрационные файлы от клиентов**: [name]\_reg\_csk\_[salt].txt, содержащие публичные ключи.

-Каждый файл регистрируется в БД, связываясь с конкретным голосованием.

**4. Завершение сбора ключей и их рассылка**

-После получения всех клиентских ключей:

-ЦСК создаёт файл с ключами для ЦИК: cik\_key\_csk\_[salt].txt.

-Также каждому клиенту рассылается файл с ключом ЦСК: [name]\_key\_csk\_[salt].txt.

**5. Отправка скрытых меток клиентами**

-Клиенты генерируют **слепую метку** (hash) и отправляют её ЦИК: [name]\_hidden\_label.txt.

-Метка слепится с помощью ключа ЦСК, что обеспечивает анонимность.

**6. Получение слепой подписи**

-ЦИК подписывает слепую метку и отправляет подписанный файл клиенту: [name]\_blind\_signature\_[salt].txt.

-Эта подпись потребуется для формирования корректного бюллетеня.

**7. Создание и отправка голосов**

-Клиент:

-Расшифровывает слепую подпись.

-Шифрует бюллетень с выбором (вариант голосования).

-Добавляет слепую подпись и отправляет голос ЦИК: voting\_[salt].txt.

-Благодаря применению слепой подписи, ЦИК не может соотнести выбор с конкретным избирателем.

**8. Завершение голосования**

-После завершения голосования ЦИК:

-Расшифровывает и проверяет бюллетени.

-Подсчитывает голоса.

-Генерирует файл с результатами: [title]\_result\_[salt].txt и рассылает участникам.

Особенности реализации

**-Слепая подпись**:

-Используется для отделения личности избирателя от содержимого голоса.

-Реализована через методы blind\_message, sign\_message\_blinded.

**-RSA-шифрование**:

-Все ключи и бюллетени зашифрованы с помощью алгоритма RSA.

-Генерация ключей — generate\_rsa\_key\_pair, encrypt\_ballot.

**-Файловая передача**:

-Коммуникация осуществляется через файловые сообщения.

-Все файлы имеют стандартизированные имена и хранятся в папках:

-read\_message (входящие).

-sent\_message (исходящие).

**-Асинхронность**:

-Этапы 5, 6 и 7 выполняются клиентами по порядку, но в системе — параллельно.

-Система рассчитана на независимую работу участников.

**-Защита данных**:

-Каждый участник имеет свою пару RSA-ключей.

-Слепые подписи исключают возможность слежки за выбором.

4.2.3 Заключение

Криптографическая система обеспечивает:

**-Анонимность голосования** — за счёт технологии слепой подписи.

**-Достоверность** — через криптографическую проверку меток и подписей.

**-Безопасность** — благодаря обязательной авторизации, шифрованию и контролю целостности файлов.

Такой подход позволяет реализовать безопасное электронное голосование с разделением ролей между компонентами и минимальным доверием между сторонами.

### 4.3 Тестирование работоспособности комплекса

4.3.1 Цель тестирования

Проверка корректности работы всех компонентов системы:

-Активация приложений (клиент, ЦСК, ЦИК).

-Регистрация участников и создание голосования.

-Работа с криптографическими методами (ключи, слепые подписи).

-Подсчет результатов и их верификация.

4.3.2 Активация приложений

Приложение ЦСК:

-Действие: Первый запуск CSK\_app.py.  
-Ожидаемый результат:

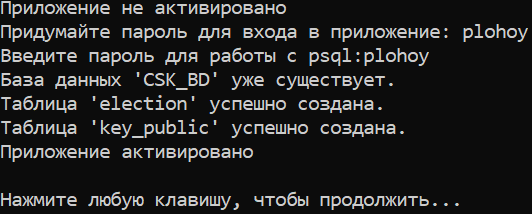
-Создается файл kesh\_csk.txt, папки для сообщений.

Рисунок 2. Скриншот терминального окна при первом запуске приложения ЦСК.

Приложение ЦИК:

-Действие: Запуск CIK\_app\_1.py и ввод пароля.

-Ожидаемый результат:

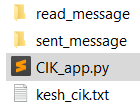
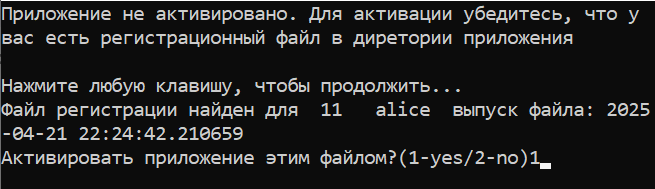
-Появление файла kesh\_cik.txt и структуры папок.

Рисунок 3.Скриншот папки с приложением ЦИК.

Клиентское приложение:

-Действие: Запуск client\_app\_1.py при отсутствии файла kesh\_client.txt.  
-Ожидаемый результат:

-Приложение запрашивает регистрационный файл (\*\_start\_client\_\*.txt).

Рисунок 4. Скриншот терминального окна при первом запуске приложения клиента.

-После ввода паролей (для приложения и PostgreSQL) создаются папки sent\_message, read\_message и файл kesh\_client.txt.

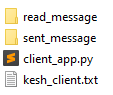


Рисунок 5.Скриншот папки с приложением клиента.

4.3.3 Сценарий голосования

Шаг 1. Создание голосования (ЦИК)

-Действие:

-В меню ЦИК выбрать "Создать голосование".

-Ввести параметры: название, варианты ответов ("Да"/"Нет").

-Ожидаемый результат:

-В БД появляется запись в таблице elections.

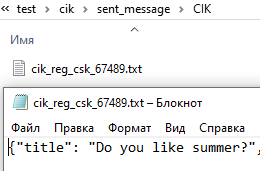
-В папке sent\_message/CIK создается файл cik\_reg\_csk\_\*.txt.  


Рисунок 6.Скриншот папки ЦИК с созданным сообщением.

Шаг 2. Рассылка приглашений (ЦСК)

-Действие:

-ЦСК обрабатывает файл cik\_reg\_csk\_\*.txt, рассылает ключи клиентам.

-Ожидаемый результат:

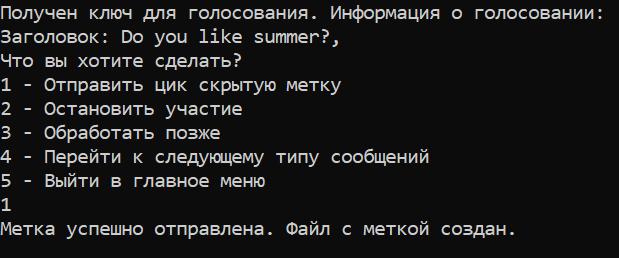
-Клиентское приложение получает ключ для голосования из файла [имя\_клиента]\_key\_csk\_\*.txt.

Рисунок 7. Скриншот терминального окна при получении ключа приложением клиента.

Шаг 3. Участие клиента

-Действие:

-Клиент обрабатывает файл \*\_invitation\_to\_vote\_\*.txt, отправляет слепую метку.

-Ожидаемый результат:

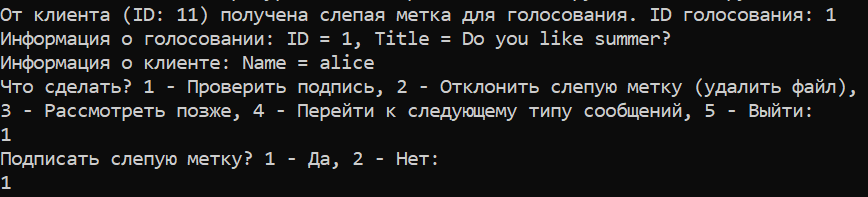
-Приложение ЦИК получает слепую метку в виде файла [имя]\_hidden\_label\_\*.txt.

Рисунок 8. Скриншот терминального окна при получении слепой метки приложением ЦИК.

Шаг 4. Проверка меток (ЦИК)

-Действие:

-ЦИК проверяет подпись клиента, отправляет Слепую подпсиь.

-Ожидаемый результат:

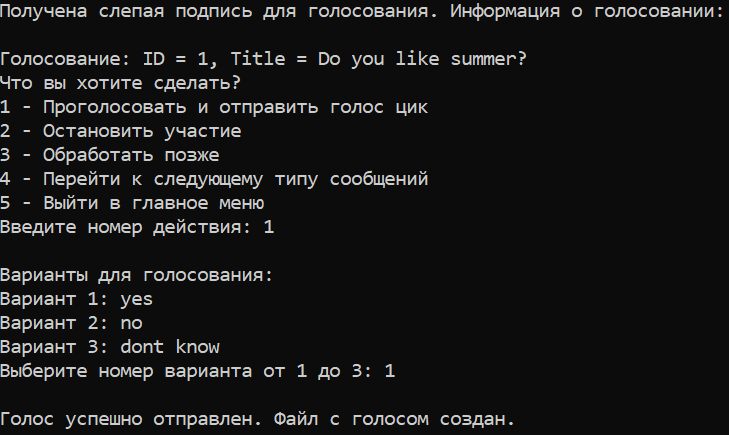
-Выполнение операции создания голоса в клиентском приложении.

Рисунок 9. Скриншот терминального окна при получении слепой подписи приложением клиента и создание голоса

Шаг 5. Голосование

-Действие:

-Клиент выбирает вариант ("yes"), отправляет зашифрованный бюллетень.

-Ожидаемый результат:

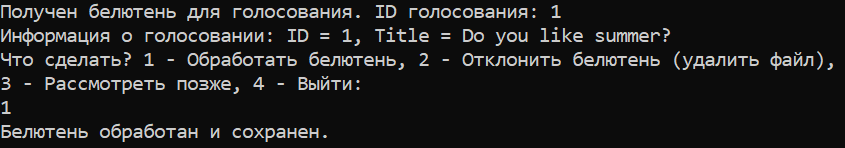
-Файл voting\_\*.txt в папке read\_message ЦИК.

Рисунок 10. Скриншот терминального окна при получении голоса приложением ЦИК

4.3.4 Завершение голосования

Шаг 1. Подсчет результатов (ЦИК)

-Действие:

-ЦИК завершает голосование, проверяет подлинность бюллетеней.

-Ожидаемый результат:

-В БД обновляются результаты в таблице election\_options.

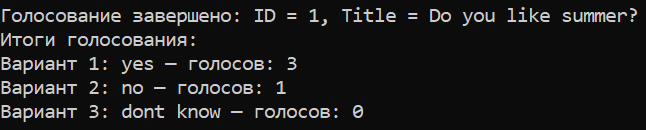
-Рассылаются файлы \*\_result\_\*.txt.

Рисунок 11. Скриншот терминального окна при завершении голосования приложением ЦИК

Шаг 2. Проверка клиентом

-Действие:

-Клиент открывает файл \*\_result\_\*.txt.

-Ожидаемый результат:

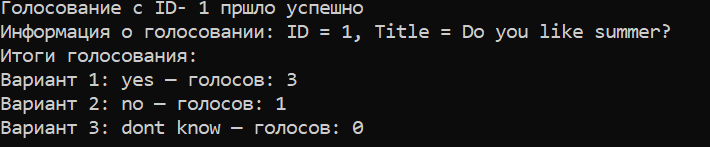
-Отображение корректных результатов

Рисунок 11. Скриншот терминального окна при чтении результатов приложением клиента

4.3.5. Результат тестирования

Успешная активация: Все приложения создают файлы kesh\_\*.txt и структуру папок.

Корректное голосование: Данные передаются без искажений, подписи верифицируются.

Итоги: Результаты совпадают с поданными голосами, ошибки отсутствуют.

4.4 Сравнение с аналогичными решениями

4.4.1 Сравнение с классическими протоколами электронного голосования

Разработанная система основана на модифицированном протоколе слепой подписи, который обеспечивает:

-Анонимность через механизм слепых меток (M\_bl).

-Верифицируемость за счет публикации пар {M, B}.

-Защиту от подмены с помощью ЭЦП ЦИК (DS\_ц.bl).

Отличия от традиционных систем (например, Helios, Prêt-à-Voter):

|  |  |  |
| --- | --- | --- |
| Критерий | Наше решение | Классические системы |
| Анонимность | Полная (blind/unblind) | Частичная (псевдонимы) |
| Роль ЦИК | Подписывает метки, но не видит B | Полный контроль над данными |
| Проверяемость | Каждый избиратель может проверить {M, B} | Требует доверия к серверу |
| Инфраструктура | Файловая система + БД | Централизованные серверы |

Таблица 1. Сравнение классической систем с нашим решением

Преимущества:

-Децентрализованная верификация – избиратели независимо проверяют свои голоса.

-Минимальные требования – не нужны сложные криптографические схемы (в отличие от гомоморфного шифрования в Helios).

Недостатки:

-Зависимость от ЦИК – если ЦИК откажется публиковать {M, B}, проверка невозможна.

-Ручная передача файлов – требует автоматизации (например, через API).

4.4.2 Сравнение с блокчейн-системами

Общие черты с блокчейном:

-Неизменяемость: После публикации {M, B} данные нельзя изменить.

-Псевдопрозрачность: Участники могут проверить, что их голос учтен.

Ключевые отличия:

|  |  |  |
| --- | --- | --- |
| Параметр | Наше решение | Блокчейн (например, Polyas) |
| Консенсус | Не требуется | Нужен (PoW/PoS) |
| Скорость | Быстрее (нет майнинга) | Медленнее (подтверждение блоков) |
| Анонимность | Выше (нет связки ID B) | Зависит от реализации |
| Масштабируемость | Легко масштабируется | Ограничена пропускной способностью сети |

Таблица 2. Сравнение блокчейн-систем с нашим решением

Блокчейн избыточен для систем с доверенной третьей стороной (ЦИК). Наше решение эффективнее для:

-Закрытых голосований (корпоративные решения).

-Систем с низкими latency-требованиями.

4.4.3 Сравнение с государственными системами (Москва, Эстония)

Сходства:

-Использование ЭЦП для аутентификации.

-Публикация результатов с возможностью проверки.

Различия:

|  |  |  |
| --- | --- | --- |
| Критерий | Наше решение | Гос. системы (e-Voting) |
| Анонимность | Да (blind-подпись) | Нет (привязка к ID в Госуслугах) |
| Децентрализация | Частичная (ЦИК + ЦСК) | Полная централизация |
| Прозрачность | Ограниченная (только для участников) | Полная (открытый аудит) |
| Юридическая сила | Требует доработки | Уже легализована |

Таблица 3. Сравнение гос. ситемы с нашим решением

Вывод: Государственные системы удобны для массовых выборов, но наше решение лучше подходит для:

-Конфиденциальных опросов (например, врачебные комиссии).

-Систем с повышенными требованиями к анонимности.

4.4.4 Уникальные особенности разработанного решения

1. Гибридная анонимность: Избиратель скрывает метку (M\_bl), но может доказать, что его голос учтен через {M, B}.

2. Минимальная инфраструктура: Не требует сложных серверов – достаточно обмена файлами.

3. Защита от coercion: Так как B шифруется на ключе ЦИК, принуждение к раскрытию выбора бесполезно.

4.4.5 Предложения по улучшению

1. Автоматизация обмена сообщениями: Замена файлового обмена на REST API или WebSockets.
2. Добавление zk-SNARKs: Для доказательства корректности подсчета без раскрытия M.
3. Интеграция с государственными системами: Использование ГОСТ-ключей для соответствия российским стандартам.

4.4.6 Заключение

Разработанный протокол сочетает анонимность, простоту и проверяемость, превосходя аналоги по:

-Безопасности: Нет единой точки отказа.

-Гибкости: Может работать как в локальной сети, так и в распределенной среде.

-Соответствию стандартам: Криптография на основе RSA/ЭЦП совместима с ГОСТ Р 34.10-2018.

Перспективы: Внедрение в корпоративные системы голосования и медицинские комиссии, где критична конфиденциальность.

5. ЗАКЛЮЧЕНИЕ

5.1. Основные выводы

В ходе выполнения курсовой работы была разработана система электронного голосования на основе протокола слепой подписи, включающая три ключевых компонента:

-Клиентское приложение – для участия избирателей.

-Приложение ЦСК – для генерации и распределения ключей.

-Приложение ЦИК – для организации голосования и подсчета результатов.

Результаты работы:

-Реализация криптографических методов – успешно применены алгоритмы RSA, слепой подписи, шифрования и верификации данных.

-Разработка структуры БД – созданы таблицы для хранения голосований, ключей и результатов с использованием PostgreSQL.

-Разработка системы обмена сообщениями – передача данных организована через файловую систему (папки read\_message, sent\_message).

-Соответствие протоколу – система обеспечивает анонимность, верифицируемость и защиту от подделки голосов.

Оценка эффективности протокола:

-Преимущества:

-Анонимность – ЦИК не знает, как проголосовал конкретный избиратель.

-Защита от фальсификаций – бюллетени подписаны криптографически.

-Децентрализованная проверка – ЦСК и ЦИК работают независимо.

-Недостатки:

-Требует доверия к ЦИК – если ЦИК скомпрометирован, возможна подмена результатов.

-Нет публичного реестра – в отличие от блокчейна, нельзя провести полностью прозрачный аудит.

5.2. Предложения по улучшению системы

Для повышения надежности и функциональности системы можно внести следующие изменения:

1. Внедрение мультиподписи (Multi-Signature) для ЦИК. В настоящий момент существует проблема: ЦИК – единый центр контроля. Возможным решением будет разделить ключи между несколькими доверенными лицами, чтобы подпись требовала согласия большинства.

2. Добавление верифицируемого микширования (Mixnet). Проблема в том, что теоретически ЦИК может связать бюллетени с избирателями. Решением может быть использовать алгоритмы микширования для дополнительной анонимизации.

3. Интеграция с блокчейном для публичного аудита. Причина - нет возможности независимой проверки без участия ЦИК. Решением может служить идея о том, что хеши бюллетеней можно записывать в блокчейн (например, Ethereum или Hyperledger).

4. Улучшение пользовательского интерфейса. Так как текущая версия работает через терминал, можно разработать веб-интерфейс или мобильное приложение.

5. Оптимизация работы с БД необходима при большом числе участников, так как возможны задержки. Решить проблему можно, если добавить индексацию, кэширование запросов.

5.3. Область применения и перспективы развития

Где может применяться система:

-Корпоративные голосования (выборы руководства, опросы сотрудников).

-Академические выборы (студенческие советы, защита проектов).

-Локальные референдумы (например, ТСЖ, муниципальные вопросы).

Перспективы масштабирования:

-Поддержка большего числа участников (оптимизация БД, распределенные серверы).

-Интеграция с государственными системами (например, авторизация через Госуслуги).

-Добавление квантово-устойчивых алгоритмов (на случай развития квантовых вычислений).

Вывод: Разработанная система соответствует современным требованиям к безопасности и анонимности и может быть адаптирована для различных сценариев голосования. Дальнейшее развитие проекта должно быть направлено на усиление децентрализации и упрощение взаимодействия с пользователями.

Рекомендации для будущих работ:

-Провести стресс-тестирование системы при большом числе участников.

-Реализовать веб-интерфейс для удобства голосования.

-Исследовать возможность использования zk-SNARKs для доказательства корректности подсчета без раскрытия данных.

Эта работа подтвердила, что криптографические методы могут обеспечить честные и анонимные выборы, а также показала направления для дальнейшего совершенствования системы.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. ГОСТ Р 34.10-2012. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. — Введ. 2013-07-01. — М. : Стандартинформ, 2013. — 14 с.
2. ГОСТ Р 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хэширования. — Введ. 2013-07-01. — М. : Стандартинформ, 2013. — 18 с.
3. Chaum D. Blind Signatures for Untraceable Payments // Advances in Cryptology : Proceedings of CRYPTO 82. — New York : Springer, 1983. — P. 199–203.
4. Официальная документация PostgreSQL [Электронный ресурс]. — Режим доступа: <https://www.postgresql.org> (дата обращения: 20.04.2025).
5. Бобровский С. В., Кузнецов Н. П. Электронное голосование и правовые основы его применения // Информационное право. — 2020. — № 3. — С. 47–52.
6. Курейчик В. В. Криптография и защита информации : учебник для вузов. — М. : Горячая линия – Телеком, 2021. — 432 с.
7. Гущин А. С. Методы защиты информации : учебное пособие. — СПб. : Питер, 2022. — 256 с.
8. Фейтман И. Л., Кравцов В. М. Криптография : теория и практика. — М. : МИФ, 2019.
9. Шифр Р. Секреты криптографии. — СПб. : БХВ-Петербург, 2018.
10. Кауфман Ч., Павлович М. Безопасность сети : от шифрования до криптографии. — М. : Вильямс, 2017.
11. Ривест Р., Шамир А., Адельман Л. Основы криптографии с открытым ключом. — М. : Инфра-М, 2015.
12. Абельсон Х., Суссман Дж. Криптография и безопасность данных. — СПб. : Питер, 2020.
13. Stallings W. Cryptography and Network Security. — 7th ed. — Pearson, 2017.
14. Menezes A. J., Van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. — Boca Raton : CRC Press, 1997.
15. Diffie W., Hellman M. E. New Directions in Cryptography // IEEE Transactions on Information Theory. — 1976.
16. Anderson R. Security Engineering : A Guide to Building Dependable Distributed Systems. — 3rd ed. — Wiley, 2020.
17. Schneier B. Applied Cryptography. — 2nd ed. — Wiley, 1996.
18. GDPR Guidelines. General Data Protection Regulation (EU) 2016/679 [Электронный ресурс]. — Оfficial EU website.
19. Ferguson N., Schneier B., Kohno T. Cryptography Engineering : Design Principles and Practical Applications. — Wiley, 2010.
20. Shestakov V. I. Modern Methods of Electronic Voting : Challenges and Solutions. — Moscow : Academy of Sciences, 2020.
21. Nakamoto H. Bitcoin : A Peer-to-Peer Electronic Cash System [Электронный ресурс]. — 2008. — URL: https://bitcoin.org/bitcoin.pdf
22. Wright L., Atkeson P. Blind Signatures for Digital Payments. — 2006.
23. Brennan T. H. Blockchain Technology and Voting : The Rise of Voatz. — 2018.
24. Анисимов К. В. Протокол голосования с одной Центральной комиссией на основе «слепой» подписи [Электронный ресурс]. — URL: <https://sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema15/tema15_3#p1537> (дата обращения: 20.04.2025).

ПРИЛОЖЕНИЕ

### Руководство пользователя для клиентского приложения (client\_app\_1.py)

#### 1. **Активация приложения**

* **Первый запуск**:
  + Убедитесь, что в директории приложения присутствует регистрационный файл формата \*\_start\_client\_\*.txt.
  + Приложение автоматически обнаружит файл и предложит активироваться.
  + Выберите 1 для активации, затем:
    - Придумайте пароль для входа в приложение.
    - Введите пароль для подключения к PostgreSQL (проверяется корректность).
  + Приложение создаст необходимые папки (sent\_message/CSK, sent\_message/CIK, read\_message) и зашифрует данные в файл kesh\_client.txt.
* **Последующие запуски**:
  + Введите пароль, созданный при активации. После 3 неудачных попыток приложение завершит работу.

#### 2. **Работа с приложением**

* **Главное меню**:
  + **1. Чтение сообщений**:
    - Автоматически обрабатывает файлы в папке read\_message:
      * Приглашения на голосование (\*\_invitation\_to\_vote\_\*.txt).
      * Ключи от ЦСК ([ваше\_имя]\_key\_csk\_\*.txt).
      * Слепые подписи ([ваше\_имя]\_blind\_signature\_\*.txt).
      * Результаты голосования (\*\_result\_\*.txt).
    - Выберите режим: автоматический (все действия подтверждаются) или ручной (запрос на каждое действие).
  + **2. База данных голосований**:
    - Просмотр текущих голосований, их статусов (ожидание ключей, завершено и т.д.).
    - Удаление голосований.
  + **3. Выход**.

#### 3. **Важные моменты**

* **Безопасность**:
  + Пароль для PostgreSQL должен совпадать с паролем, используемым в СУБД.
  + Файл kesh\_client.txt хранит зашифрованные данные — не удаляйте его.
* **Ошибки**:
  + При отсутствии регистрационного файла или неверном пароле приложение сообщит об этом.

### Руководство пользователя для приложения ЦСК (CSK\_app\_1.py)

#### 1. **Активация приложения**

* **Первый запуск**:
  + Введите пароль для входа в приложение и пароль для PostgreSQL.
  + Приложение создаст БД, папки (sent\_message/CIK, read\_message) и файл kesh\_csk.txt.
* **Последующие запуски**:
  + Авторизуйтесь с помощью пароля. После 3 ошибок приложение закроется.

#### 2. **Работа с приложением**

* **Главное меню**:
  + **1. Чтение сообщений**:
    - Обработка файлов от клиентов (cik\_reg\_csk\_\*.txt) и ЦИК.
    - Подтверждение добавления ключей или их отклонение.
  + **2. База данных ключей**:
    - Просмотр голосований, управление участниками (добавление/удаление).
    - Запуск рассылки ключей клиентам.
  + **3. Выход**.

#### 3. **Особенности**

* **Рассылка ключей**:
  + После подтверждения голосования ЦСК генерирует файлы с ключами для каждого клиента в sent\_message/CIK.
* **Логирование**:
  + Все операции с ключами фиксируются в БД.

### Руководство пользователя для приложения ЦИК (CIK\_app\_1.py)

#### 1. **Активация приложения**

* **Первый запуск**:
  + Задайте пароль для приложения и пароль PostgreSQL.
  + Приложение создаст БД, папки (sent\_message/CIK, read\_message) и файл kesh\_cik.txt.
* **Последующие запуски**:
  + Введите пароль. Неверный ввод 3 раза приведет к закрытию.

#### 2. **Работа с приложением**

* **Главное меню**:
  + **1. Чтение сообщений**:
    - Обработка слепых меток (\*\_hidden\_label\_\*.txt) и бюллетеней (voting\_\*.txt).
    - Проверка подписей, расшифровка голосов.
  + **2. Пользователи**:
    - Добавление/удаление пользователей в системе.
  + **3. Голосования**:
    - Создание/редактирование голосований.
    - Просмотр активных и завершенных голосований.
    - Подсчет результатов (end\_election).
  + **4. Выход**.

#### 3. **Критические операции**

* **Завершение голосования**:
  + ЦИК проверяет подлинность всех бюллетеней, после чего рассылает результаты клиентам.
* **Безопасность**:
  + Все данные шифруются. Убедитесь, что файл kesh\_cik.txt защищен.

### Общие рекомендации для всех приложений

1. **Пароли**:
   * Используйте сложные пароли для приложений и PostgreSQL.
   * Не передавайте файлы kesh\_\*.txt третьим лицам.
2. **Файловая система**:
   * Не удаляйте папки sent\_message и read\_message — они создаются автоматически.
3. **Ошибки**:
   * При возникновении ошибок (например, отсутствие файлов) приложение выведет сообщение. Следуйте инструкциям на экране.
4. **Резервное копирование**:
   * Регулярно сохраняйте файлы kesh\_\*.txt и БД.