Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
Системное программирование

Андреев Сергей Иванович

Разработка и реализация протокола с нулевым разглашением для решения задачи анонимной передачи данных в Ethereum-подобных блокчейн сетях.

Выпускная квалификационная работа бакалавра

Научный руководитель:
доц. каф. системного программирования,
к.ф.м.н. Романовский К. Ю.

Рецензент:
директор по развитию ООО «Манзони»,
к.т.н. Тихомиров В. А.

Санкт-Петербург
2021

SAINT-PETERSBURG STATE UNIVERSITY

Mathematical Software and Information Systems Administration
System Programming

Sergey Andreev

Development and implementation of a zero knowledge protocol for solving the problem of anonymous data transfer in Ethereum-like blockchain networks.

Bachelor's Thesis

Scientific supervisor:
Assoc. Prof. of the dep. of System Programming
PhD Romanovsky K. Y.

Reviewer:
Chief Business Developer Officer Manzoni LLC,
PhD Tikhomirov V. A.

Saint-Petersburg
2021

Оглавление

Введение

В последнее время наблюдается активный рост количества денежных операций с использованием криптовалют. Такие транзакции проходят блокчейн-сетях, представляющих из себя распределенные децентрализованные базы данных. Например, за 2021 год среднее месячное количество транзакций в блокчейн-сети Ethereum выросло с 17 миллионов до 37 миллионов [6]. Такой бурный рост обусловлен надежностью и прозрачностью проводимых операций.

Однако, прозрачность и децентрализованность системы имеют и недостатки. Для обеспечения консенсуса о едином состоянии сети необходима полная открытость вносимых изменений для всех ее участников. Это значит, что любой участник сети может отследить все транзакции, когда либо отправленные или полученные на какой-либо адрес [1]. Если участник сети скомпрометирует свою личность, то есть допустит установления связи между своим адресом в сети и своей личностью, то все его прошлые и будущие операции будут публично доступны.

Свою личность можно легко скомпрометировать, например, при покупке или выводе криптовалюты на биржах, так как зачастую для таких операций требуется банковская карта. Также свою личность можно скомпрометировать просто при обменах внутри сети со своими знакомыми или использовании различных "горячих кошельков".

На данный момент существуют различные блокчейны обеспечивающие анонимность транзакций [15], однако, они не пользуются такой популярностью, как Ethereum. Это обусловлено тем, Ethereum — это не просто криптовалюта, это целая платформа, которая позволяет легко создавать собственные криптовалюты-токены внутри одного блокчейна. Такие токены представляют из себя программы, использующие блокчейн для сохранения данных и выполнения своего кода. Эти токены пользуются очень большой популярностью: на момент 2021 года существует более 370 тысяч различных токенов, а объем торгов токена Tether USDT в феврале 2021 оказался самым высоким среди всех криптовалют, в 4 раза превзойдя объем торгов криптовалюты Ethereum [5].

Децентрализованные биржи (DEX) позволяют обменивать токены в основанных на технологии Ethereum блокчейнах. Такие биржи управляются открытыми смарт-контрактами, поэтому логика осуществления торгов строго задана, неизменна и прозрачна. Это значит, что любой желающий может убедится в честности проводимых операций, а также торговать цифровыми активами без централизованного посредника. В связи с этим децентрализованные биржи набирают сильную популярность и занимают важное место в области обмена цифровых активов: только за 2020 год объем торгов на таких биржах вырос с 3 миллиардов долларов до 121 миллиарда долларов [7]. Однако, в силу прозрачности своей работы, такие биржи подвержены проблеме отсутствия анонимности участников, как и обычные транзакции.

Анонимность действий в сети позволяет преодолевать санкционные режимы, свободно финансировать благотворительные, политические и прочие организации. Бизнесу необходима анонимность транзакций для сокрытия своей финансовой отчетности от конкурентов и злоумышленников [4]. Частные пользователи же могут просто желать оставаться анонимными для сохранения неприкосновенности частной жизни.

Таким образом, возникает проблема обеспечения анонимности при торговле токенами на децентрализованных биржах в основанных на технологии Ethereum блокчейнах.

Протокол анонимной передачи активов был разработан в 2020 году компанией Tornado Cash [1]. Но данный протокол не позволяет осуществлять торговлю, а позволяет лишь единожды произвести депозит, размер которого строго задан протоколом, и единожды перевести его на другой адрес. Но даже после этого будет невозможно сохранить конфиденциальность своей дальнейшей финансовой стратегии: обмены крупных сумм токенов, хоть уже и не будут привязаны к скомпрометированным публичному адресу, но будут привязаны к одному и тому же анонимному адресу. Крупные компании и владельцев токенов зачастую такое может не устроить, так как раскрытие даже такой информации, как частая продажа фиксированного токена с одного и того же аноним-

[1] https://tornado.cash/

ного адреса, может очень сильно повлиять на состояние рынка. Разрешению описанной проблемы и было решено посвятить данную работу.

Постановка задачи

Целью данной работы является разработка и реализация протокола с нулевым разглашением для решения задачи анонимной передачи данных с целью осуществления обмена активов в Ethereum-подобных блокчейн сетях.

Для достижения цели были поставлены следующие задачи:

- Провести обзор предметной области и анализ существующих решений
- Разработать протокол передачи и валидации данных с нулевым разглашением
  - Для случаев создания кошелька и внесения депозита
  - Для случая обмена активов
  - Для случая частичного или полного снятия средств
- Разработать архитектуру системы анонимного обмена активов и реализовать взаимодействующие по протоколу части разработанной системы
- Провести тестирование реализованных частей системы

1. Обзор

Обзор включает в себя введение в предметную область блокчейн-платформы Ethereum, обзор доказательств с нулевым разглашением, а также средств для их создания. В последней секции обзора представлено сравнение похожих решений.

1.1. Ethereum-подобные блокчйены

Blockchain представляет из себя децентрализованную распределенную базу данных [20]. База данных состоит из выстроенной специальным образом последовательности блоков, которую хранит каждый узел сети. Блок содержит набор транзакций, хеш предыдущего блока, некоторые дополнительные данные и хеш от всех данных, содержащихся в этом же блоке. Попытка изменения истории операций участником сети изменит хеш блока, содержащего информацию об этих операциях, и сделает всю последующую цепочку недействительной.

Отличительной особенностью блокчейна Ethereum является наличие двух различных типов аккаунтов, разделяющих все адресное пространство: внешних аккаунтов, контролируемых приватным ключом, и смарт-контрактов, управляемых своим кодом.

Смарт-контракты представляют из себя программы, хранящие свой код и данные в блокчейне, способные взаимодействовать с блокчейном: отправлять средства, взаимодействовать с другими контрактами и т.д. Внешние аккаунты взаимодействуют с контрактами посредством транзакций, вызывая функции контракта. Выполняется код контрактов на виртуальных машинах майнеров, а результат выполнения программы попадает в блок. После публикации смарт-контракта в сети, его код становится общедоступным, и изменить его уже невозможно. Таким образом, смарт-контракты позволяют создавать децентрализованные приложения или программы, доверие к которым основывается на общедоступности и неизменности их кода, а также на децентрализованной природе среды их исполнения.

1.1.1. Токены и DEX

Яркими примерами таких программ служат токены и DEX (Decentralized exchange) — децентрализованные обменники этих токенов.

Токены представляют из себя контракты, реализующие определенный интерфейс. Такой интерфейс обычно включает в себя структуру для хранения балансов данного токена у пользователей, методы для передачи токенов, получения информации о балансе и т.д. Одними из самых распространенных стандартов являются ERC-20 и ERC-721 [13] [9] [16]: они позволяют рассматривать токены как электронный актив, который так или иначе можно переводить с аккаунта на аккаунт.

Децентрализованные биржи — это контракты, позволяющие обменивать токены некоторых стандартов ERC и основную валюту сети [2]. Но в отличие от обычных бирж, в DEX отсутствует центральный управляющий аппарат.

DEX обычно состоит из контракта-маршрутизатора и множества контрактов-обменников для различных пар токенов [17]. Участники сети добровольно вносят свои средства на контракты-обменники в качестве вкладов, получая за это комиссию, а вложенные средства используются при обменах. Контракт-маршрутизатор оценивает состояние контрактов-обменников, определяет курс обмена и предоставляет интерфейс для транзакций-обменов.

Децентрализованные биржи обладают большими преимуществами перед обычными биржами, и становятся все популярнее в последнее время [3]. Это обосновано прозрачностью обменов на таких биржах, отсутствием необходимости внесения своих данных, таких как номер банковской карты, данные паспорта.

Однако, прозрачность взаимодействия с DEX не позволяет проводить анонимные торги: адрес обменивающего средства пользователя, обмениваемые суммы всегда открыты. Эту проблему можно решить, если проксировать обмены через посредника, взаимодействующего с бир-

[2]Далее токеном будем называть контракт стандарта ERC-20 или основную валюту сети (Ether в Ethereum mainnet)

жей от своего имени, и позволяющего после торгов вывести средства на необходимый анонимный адрес. Для сохранения децентрализованности, такой посредник должен представлять из себя не off-chain сервер, а on-chain контракт-анонимайзер.

1.2. Доказательства с нулевым разглашением

Так как код смарт-контракта, его данные в памяти, параметры передаваемые в его методы при вызовах в транзакциях полностью открыты, то невозможно обеспечить приватность взаимодействия с контрактом-посредником, а значит и анонимность, классическими алгоритмами симметричного или асимметричного шифрования.

Однако, даже в такой ситуации, протоколы доказательства с нулевым разглашением могут обеспечить требуемую анонимность.

Такие протоколы позволяют доказать факт обладания некоторой информацией без раскрытия ее содержимого [11]. Например, возможно доказать факт знания прообраза одного из хэшей данного списка, не раскрывая ни прообраз, ни образ в списке.

Протокол	Интерактивный	Сложность создания доказательства	Сложность проверки доказательства	Размер доказательства	Необходимость trusted ceremony
zk-SNARK	нет	$O(N \cdot log(N))$	$O(1)$	$O(1)$	да
zk-STARK	да	$O(N \cdot poly - log(N))$	$O(N \cdot poly - log(N))$	$O(N \cdot poly - log(N))$	нет
Bulletproofs	нет	$O(N \cdot log(N))$	$O(N)$	$O(log(N))$	нет

Таблица 1: Протоколы доказательства с нулевым разглашением

В таблице 1 представлено сравнение протоколов доказательства с нулевым разглашением. Сравнение проводилось с целью проанализировать протоколы на необходимость предварительного взаимодействия с проверяющей стороной для генерации доказательства, сложность создания и проверки доказательства, его размер, и необходимость предварительной подготовки с использованием мультиподписи для создания доказательства и валидаторов доказательства.

Интерактивные протоколы требуют взаимодействия с проверяющей стороной для генерации доказательства, что не подходит, когда проверяющая сторона — смарт-контракт. Сложность вычислений и объем

требуемой памяти проверяющего необходимо минимизировать, так как от этого зависит комиссия при переводе. Trusted setup ceremony позволяет заранее подготовить открытую мультиподпись, что обеспечивает неинтерактивность протокола [10]. Таким образом, протокол zk-SNARK (Zero-knowledge Succinct Non Interactive ARgument of Knowledge) лучше всего подходит для решения поставленной задачи.

1.2.1. Circom и snarkjs

Для создания собственного протокола доказательства с нулевым разглашением на основе zk-SNARK использовался язык Circom [3] и библиотека snarkjs [4].

Описание сигналов и ограничений протокола

Рис. 1: Создание ключа для получения доказательства и валидатора программистом

На рисунке 1 представлена последовательность действий при создании схемы-ключа, необходимой для генерации различных доказательств по заданной схеме, а также необходимой для генерации валидатора доказательств на языке Solidity, используемого для написания

[3]https://github.com/iden3/circom
[4]https://github.com/iden3/snarkjs

смарт-контрактов на платформе Ethereum.

Входные и выходные сигналы, а также накладываемые на них ограничения описываются на языке Circom на начальном этапе разработки протокола. Сигналы могут быть публичными и приватными: публичные сигналы будут видны валидатору доказательства, а приватные — нет. Примерами публичных сигналов могут служить пары токенов для обмена, адрес получателя средств при выводе и прочие параметры, которые необходимо знать контракту-анонимайзеру для выполнения бизнес-логики. Примерами приватных сигналов могут служить ключи, используемые для создания подписи в доказательстве, баланс кошелька. Ограничения позволяют заложить в доказательстве выводимую из набора сигналов информацию, не разглашая приватных сигналов. Например, ограничения позволяют доказать, что при обмене будет использоваться не больше токенов, чем имеется на счету, без разглашения информации о количестве имеющихся на счету токенов.

Компилятор Circom позволяет преобразовать исходный код в арифметическую систему ограничений. Trusted setup процедура позволяет каждому желающему вложить свой вклад в мультиподпись полученной системы ограничений. На основе подписанной системы ограничений формируется ключ с нулевым разглашением, используя который пользователь сможет создавать доказательства по созданному протоколу [8]. Программа snarkjs позволяет осуществлять проверку доказательств и поставляется как независимая CLI-утилита, либо как JavaScript библиотека. Также с помощью подписанной системы ограничений и snarkjs можно сгенерировать блокчейн-валидатор доказательств для этой системы ограничений.

Создание доказательства

На рисунке 2 представлена последовательность действий при генерации выходных параметров по заданной схеме — ключу с нулевым разглашением. С помощью программы snarkjs на основе входных сигналов получаются выходные сигналы и доказательство с нулевым разглашением. Доказательство представляет из себя набор гомоморфных

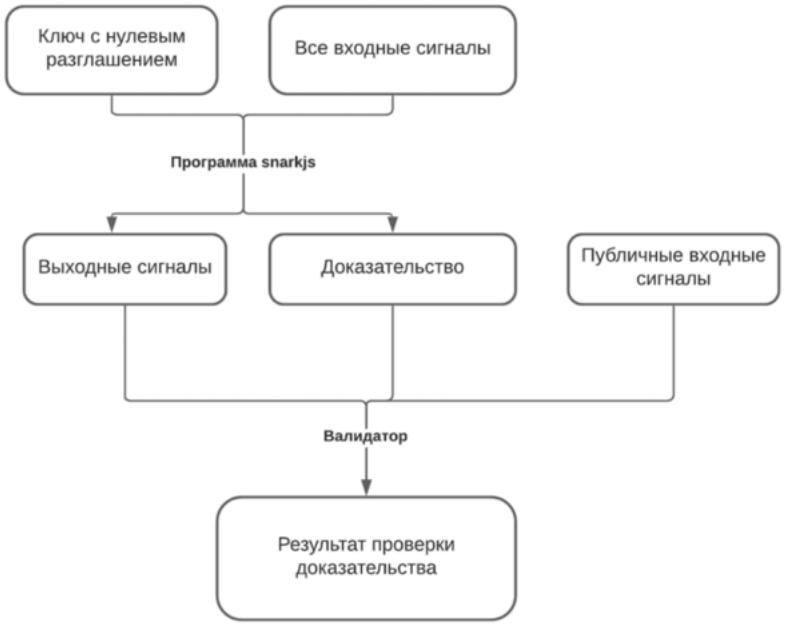

Рис. 2: Генерация выходных параметров и доказательства. Валидация доказательства

образов значений специальных полиномов, соответствующих входным сигналам, в заданной с помощью trusted setup процедуры точке, используемых при проверке выполнимости системы арифметических ограничений валидатором. Так как программа snarkjs написана на JavaScript, то создавать доказательства по протоколу можно в клиентском браузерном веб-приложении. Полученные при генерации доказательства значения вместе с публичными входными сигналами и выходными сигналами позволяют валидатору (Solidity контракту, или snarkjs утилите) проверить корректность доказательства.

1.3. Сравнение аналогов

На данный момент существует несколько протоколов анонимной передачи данных для платформы Ethereum. Некоторые из таких протоколов приведены в таблице 2.

Сравнение проводилось с целью проанализировать наличие и при-

[5]https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf
[6]https://crypto.stanford.edu/ buenz/papers/zether.pdf
[7]https://docs.starkware.co/starkex-docs-v2/overview

Протокол	Поддерживается токенов	Интеграция с DEX	Наличие поддерживаемой реализации	Полностью децентрализован
Tornado cash [5]	6	-	+	+
Zether [6]	1	-	+	+
StarkEx [7]	>100	+	+	-

Таблица 2: Протоколы анонимной передачи данных

меняемость текущих решений с точки зрения децентрализованности и возможности торговли с использования популярных токенов.

Параметры для сравнения позволили проанализировать сервисы на количество поддерживаемых токенов, возможность торговли на DEX, наличие поддерживаемой реализации и децентрализованность. Чаще подобные протоколы не позволяют проводить анонимные торги на DEX, а только проксируют простой перевод средств, что может не устроить крупные компании или владельцев больших сумм криптотокенов, так как раскрытие даже такой информации, как частая продажа фиксированного токена с одного и того же анонимного адреса, может очень сильно повлиять на состояние рынка.

StarkEx позволяет торговать на DEX более чем стами токенами, однако, он не является полностью децентрализованным, что делает его неподходящим для пользователей, желающих получить абсолютную приватность, надежность и прозрачность.

2. Разработка протокола передачи данных

В данной главе описан протокол анонимной передачи данных для платформы Ethereum, позволяющий проводить торги на DEX. В начале описываются участвующие в протоколе стороны, их возможности, и используемые понятия. Далее приводится способ хранения данных участников с обеспечением анонимности. После этого подробно описываются поддерживаемые протоколом функции.

2.1. Описание участников протокола и их возможностей

Участниками протокола являются создатель доказательства (клиент) и его валидатор (контракт). Доказательство создается клиентом по заранее заданной и подписанной арифметической системе ограничений, после чего передается вместе с публичными входными и выходными сигналами валидатору.

Протокол позволяет клиентам создавать кошельки — то есть логические цепочки состояний списка сумм токенов. Валидатор не имеет информации о списке сумм токенов на кошельке, то есть о состоянии цепочки, однако, имеет возможность проверить, что при создании доказательства состояние кошелька было изменено в соответствии с правилами: например, при снятии токенов баланс на кошельке должен уменьшиться на снимаемую сумму.

После создания кошелька, то есть инициализации цепочки состояний, клиент может внести депозит, совершить анонимный обмен, снять часть средств, либо снять средства и удалить кошелек. Внесение депозита — это изменение состояния кошелька в сторону увеличения сумм токенов без разглашения старого и нового состояния кошелька. Обмен подразумевает под собой уменьшение суммы одного из токенов в состоянии кошелька и увеличение другой в соответствии с предложениями DEX. Частичное снятие — это уменьшение суммы токенов на кошельке и фактический вывод средств на указанный клиентом адрес. Удале-

ние кошелька подразумевает фактический вывод средств на указанный клиентом адрес без возможности продолжения цепочки состояний кошелька.

2.2. Способ хранения данных участников

Так как валидатор не получает информации о состоянии кошелька, то список сумм токенов хранит клиент, обновляя его после каждой операции. Однако, валидатор должен иметь возможность убедиться, что клиент использует действительно существующую цепочку состояний некого кошелька, для которого было доказано внесение необходимой суммы. При этом не должна быть раскрыта информация, по которой можно было бы идентифицировать клиента, или сопоставить новое состояние цепочки с одним из предыдущих.

Для обеспечения описанных условий валидатор сохраняет лишь хэши от входных данных каждой операции. Хэши сохраняются листах специально структуры — дерева Меркла.

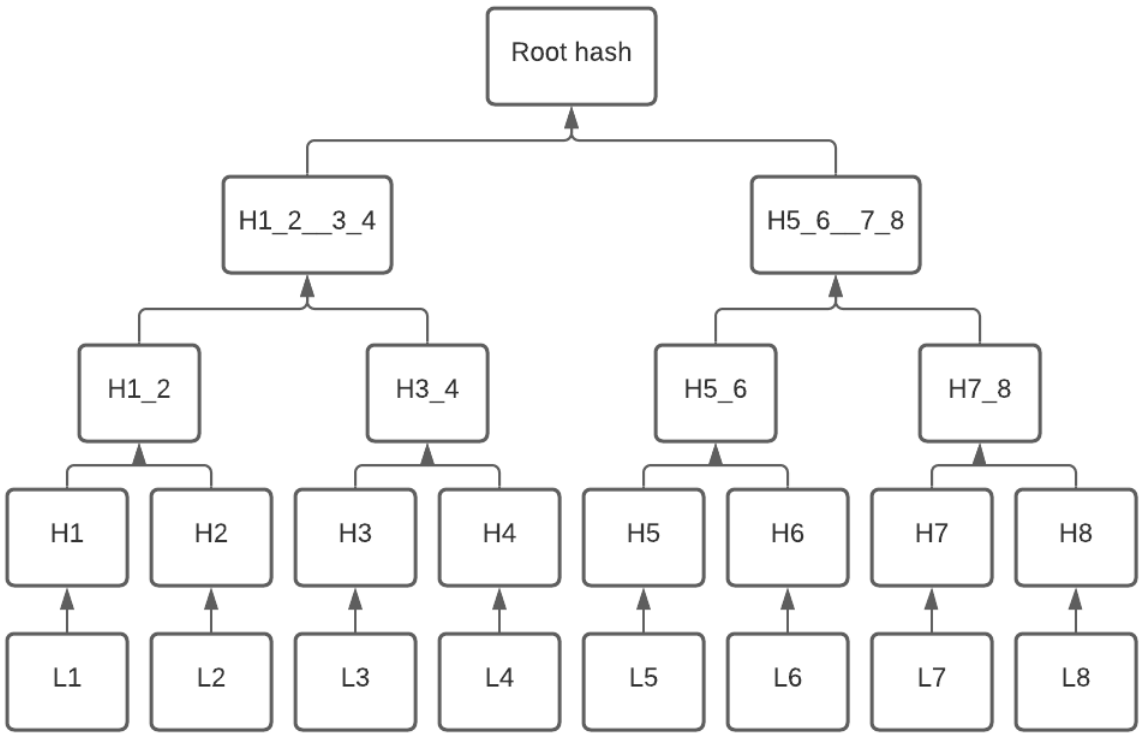

Рис. 3: Структура дерева Меркла

На рисунке 3 изображено строение дерева хэшей высоты 3. Хэши $(H1, ..., H8)$ от полезных данных $(L1, ..., L8)$ данные размещаются в листах дерева. Далее для каждой пары листов вычисляется хэш от их конкатенации. Полученные хэши размещаются в узлах над листами. Аналогичные вычисления продолжаются до тех пор, пока не будет получен корневой хэш дерева [2]. Особенностью такого дерева является невозможность изменения одного из листов без изменения корня дерева, то есть корень является "отпечатком" всех данных в листах дерева. Такая структура данных позволяет проверить, что данный хэш действительно содержится в листе: для этого необходимо получить путь в дереве от данного листа до корня. Путем называется последовательность хэшей в некоторых узлах дерева, необходимая для вычисления корневого хэша на основе данного листа. Например, путь для листа $H2$ будет представлять из себя последовательность $(H1, H3\_4, H5\_6\_\_7\_8)$.

При создании или изменении кошелька хэш от всех данных операции помещается в лист дерева Меркла, а сами данные состояния не сохраняются в валидаторе. Этот хэш в протоколе будем называть **commitment**. При совершении операции, изменяющей состояние кошелька, клиент на основе информации событий о добавлении commitment-листов от валидатора локально строит дерево Меркла, после чего вычисляет путь от своего commitment прошлой операции до корня дерева. После этого в доказательство включается сравнение полученного вычислением значения корня дерева и реальное значение, хранящееся у валидатора. При проверке доказательства валидатор сможет убедиться в том, что корень дерева Меркла был вычислен клиентом в соответствии с правилами, а также в том, что вычисленный и реальный корни совпадают. Таким образом, если клиент при совершении операции будет использовать неправильное состояние кошелька, то есть попробует воспользоваться средствами, которые он не вносил, то вычисляемый и реальный хэши корней дерева Меркла будут различными, и доказательство не будет принято.

2.3. Описание операций протокола

Описание операции включает в себя описание входных публичных и приватных сигналов, выходных сигналов доказательства, а также накладываемых на них ограничений. Сигналы как данные могут быть двух типов: числа в поле $\mathbb{Z}/p\mathbb{Z}$,
где $p = 21888242871839275222246405745257275088548364400416034343$
$69820418657580849561 7$ — простое, или массивы заданной длины из таких чисел. При этом числа в промежутке $[0, p/2]$ будем считать положительными, а числа в промежутке $[p/2 + 1, p)$ — отрицательными. Отрицательное число $-X$ будет задаваться числом $p - X$. Сигналы, представляющие из себя массивы, будут иметь обозначение в формате <название сигнала>[размер массива]. Сигналы, представляющие из себя числа не будут иметь дополнительных пометок.

Введем обозначения: TN — TokensNumber, количество поддерживаемых кошельком токенов, TRH — TreeHeight, высота (длина доказывающего пути) дерева Меркла. Высота определяет количество листов дерева (2^{TRH}), то есть количество возможных операции без перезапуска валидатора. Численные значения данных параметров зависят от бизнес-требований к конкретной реализации протокола.

2.3.1. Создание кошелька

Операция создания кошелька означает внесение средств и фиксирование этого события путем добавления соответствующего **commitment** в лист дерева Меркла — это будет первый **commitment** в цепочке **commitment-ов**, соответствующей цепочке состояний кошелька, для клиента.

Публичные входные сигналы

tokens[TN] — список сумм токенов для внесения. Соответствие между суммами и токенами устанавливается посредством индексов в массиве: первый элемент массива представляет из себя сумму для внесения заранее заданных токенов первого типа.

Приватные входные сигналы

secret — неразгалшаемый приватный ключ операции, обеспечивает анонимность.

nullifier — приватный идентификатор операции. Разглашается при последующей операции и добавляется в список израсходованных ключей.

Выходные сигналы

commitment — публичный идентификатор операции, результат хеширования входных сигналов. Будет добавлен в лист дерева Меркла.

При проведении каждой операции генерируются два приватных ключа: **secret** и **nullifier**. Они используются при формировании доказательства для следующей операции, доказывая факт проведения предыдущей операции, и осуществляя защиту кошелька от использования сторонними пользователями. При этом ключ **secret** не разглашается никогда, что обеспечивает невозможность создания злоумышленником поддельного доказательства с измененными параметрами операции. Ключ **nullifier** разглашается при следующей операции, и сохраняется валидатором, что обеспечивает защиту от двойного проведения одной и той же операции.

2.3.2. Внесение депозита

Операция внесения депозита означает добавление средств на уже существующий кошелек. Отличие этой операции от создания кошелька заключается в том, что при добавлении средств новый **commitment** добавляется в уже существующую цепочку, а не начинает ее. Это значит, что клиенту необходимо доказать знание предыдущего **commitment** своей цепочки, а также всех формирующих его параметров. При этом **nullifier** предыдущей операции будет раскрыт и сохранен валидатором как использованный, что препятствует ветвлению цепочки состояний кошелька.

Публичные входные сигналы

root — текущий корень дерева Меркла. Используется для доказательства его совпадения с вычисляемым корнем.

nullifier_old — nullifier предыдущей операции. Будет раскрыт и сохранен валидатором как использованный.

tokens_deltas[TN] — список сумм токенов для внесения в рамках депозита.

Приватные входные сигналы

tokens_old[TN] — список сумм токенов в текущем состоянии кошелька.

tokens_new_success[TN] — список сумм токенов в новом состоянии кошелька при успешном выполнении операции. При неудачном выполнении будет оставлен предыдущий список токенов.

old_secret — **secret** предыдущий операции.

new_nullifier_success, **new_nullifier_fail**, **new_secret_success**, **new_secret_fail** — новые **nullifier** и **secret** для случаев успешного и неуспешного завершения операции соответственно.

path_elements[TRH] — путь в дереве Меркла до **commitment** предыдущей операции.

path_indices[TRH] — индексы в паре для каждого из элементов пути. Индекс 0 обозначает, что элемент пути является левым в паре при хешировании, а 1 — правым.

Выходные сигналы

commitment_success — публичный идентификатор операции при успешном исходе, результат хеширования входных сигналов, соответствующих успешному исходу операции, а также общих сигналов (**root**, **nullifier_old**, **tokens_deltas[TN]**, **tokens_old[TN]**, **old_secret**, **path_elements[TRH]**, **path_indices[TRH]**). Будет добавлен в лист дерева Меркла при успешном проведении операции.

commitment_fail — публичный идентификатор операции при неудачном исходе, результат хеширования входных сигналов, соответствую-

щих неудачном исходу операции, а также общих сигналов. Будет добавлен в лист дерева Меркла при неудачном проведении операции.

Ограничения Доказательство с нулевым разглашением для данной операции содержит обоснования того, что каждый элемент **token_new_success[TN]** равен сумме соответствующих элементов **token_old[TN]** и **tokens_deltas[TN]**. А также обоснование того, что при формировании **commitment_success** использовался **tokens_new_success[TN]** и другие сигналы, соответствующие успешной операции; а при формировании **commitment_fail** — **tokens_old[TN]** и другие сигналы, соответствующие неудачной операции. Также доказывается корректность вычисления корня дерева меркла и сравнения его с реальным корнем.

2.3.3. Обмен токенов

Обмен токенов — операция, проводимая над уже существующим кошельком, цель которой — увеличить баланс одного из токенов на кошельке путем анонимной продажи некоторого количества другого токена на DEX. Аналогично операции по внесению депозита, обмен токенов добавляет новый **commitment** к уже существующей цепочке и сохраняет **nullifier** предыдущей операции в список использованных **nullifier**-ов.

Входные и выходные сигналы Все приватные и выходные сигналы данной операции совпадают с выходными и приватными сигналами операции внесения депозита, и имеют такой же смысл, за исключением дополнительного сигнала **tokens_new_fail[TN]**, который дублирует старый список средств, за вычетом комиссии за совершение операции. Данного сигнала не было в предыдущих операциях, так как в них пользователь самостоятельно посылал транзакцию в блокчейне. Публичные сигналы тоже дублируют публичные сигналы операции внесения депозита, с некоторыми отличиями. Во-первых, добавляется публичный сигнал **fee**. Данный сигнал указывает, какая комиссия в токене с индексом

0 будет снята с кошелька пользователя для оплаты услуг relayer-а — посредника, пересылающего транзакцию клиента валидатору, для сокрытия адреса отправителя. Во-вторых, добавляется сигнал **relayer**, указывающий адрес relayer-а, который будет проксировать транзакцию. На этот адрес валидатор отправит комиссию. В-третьих, сигнал **tokens_deltas[TN]** несет другой смысл: один из элементов этого массива является положительным числом и указывает размер увеличения баланса в целевом токене на кошельке при успешном обмене. Также массив содержит отрицательное [8] число, показывающее размер уменьшения баланса кошелька в обмениваемом токене. Остальные элементы массива являются нулями, что проверяется валидатором.

Ограничения Помимо ограничений из операции по внесению депозита дополнительно накладываются ограничения на проверку достаточности средств обмениваемого токена: соответствующий обмениваемому токену элемент **tokens_deltas[TN]** должен в сумме с соответственным **tokens_old[TN]** быть не меньше 0. Также баланса нулевого токена должно хватить на оплату **fee**.

2.3.4. Частичное снятие средств

Операция частичного снятия средств позволяет клиенту вывести часть средств с кошелька на заданный адрес. При этом, как и в двух предыдущих операциях, в цепочку **commitment**-ов добавляется отпечаток нового состояния кошелька, а старый **nullifier** раскрывается и сохраняется у валидатора.

Входные и выходные сигналы Приватные входные и выходные сигналы данной операции дублируют сигналы операции по обмену токенов, а среди публичных сигналов есть один дополнительный: сигнал **receiver**, указывающий на адрес, на который необходимо вывести средства. Также в отличие от операции обмена сигнал **tokens_deltas[TN]** содержит список сумм токенов для снятия.

[8] В определенном в разделе 2.3 смысле

Ограничения Ограничения дублируют ограничения операции обмена токенов, но на **tokens_deltas[TN]** накладываются ограничение, доказывающее достаточность средств для снятия заданных сумм токенов и уплаты **fee**.

2.3.5. Удаление кошелька

В отличие от частичного снятия средств, удаление кошелька не продолжает цепочку **commitment**-ов, но при этом, как и в других операциях, изменяющих состояние существующего кошелька, необходимо доказать знание информации о предыдущем звене цепочки. При этом **nullifier** предыдущей операции сохраняется в списке использованных ключей.

Входные и выходные сигналы Публичные входные сигналы включают в себя **root**, **nullifier_old**, **recepient**, **fee**, **relayer**, имеющие тот же смысл, что и в предыдущих операциях. Последним публичным сигналом является **tokens_withdraw_list[TN]**, содержащий суммы токенов для снятия. Этот сигнал обеспечивает дополнительную анонимность, так как клиент получает возможность оставить на удаляемом кошельке некоторую небольшую сумму токенов, чтобы внешний наблюдатель не мог определить операции клиента по соответствию входящих и выходящих сумм.

Приватные входные сигналы **tokens_old[TN]**, **secret_old**, **path_elements[TRH]**, **path_indices[TRH]** выполняют те же функции, что и в случае других операций.

Выходных сигналов данная операция не имеет, так как является заключающей в последовательности изменяющих кошелек операций.

Ограничения Помимо ограничений на соответствие корней дерева Меркла, вводится ограничение на не превышение каждого из элементов **tokens_withdraw_list[TN]** соответствующего ему элемента **tokens_old[TN]**.

2.4. Взаимодействие по протоколу

При взаимодействии по протоколу клиент может совершать множество операций, образующих цепочку состояний его кошелька. Каждая новая операция использует состояние, полученное в результате предыдущей операции, как исходные данные, поэтому клиент хранит актуальное состояние кошелька локально. Контракт же сохраняет только **commitment**-ы. На диаграмме 4 изображена последовательность состояний кошелька клиента и изменяющих его операций.

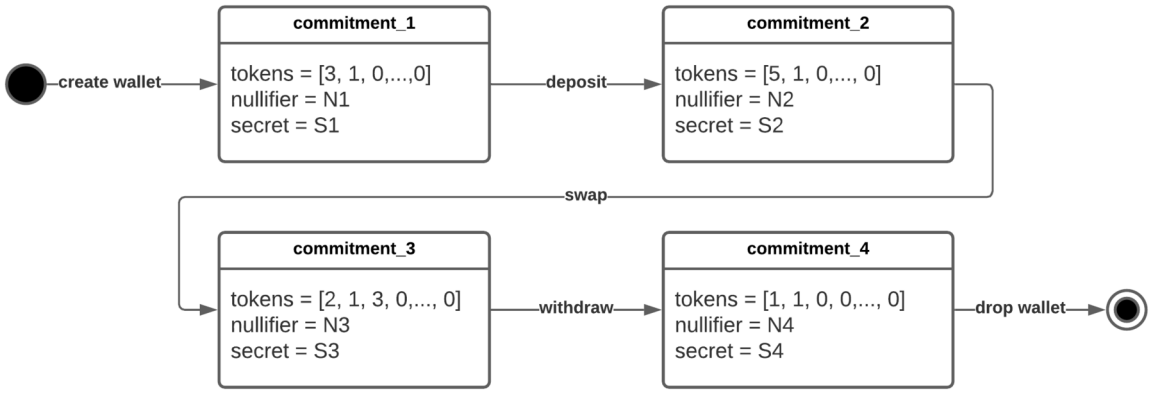

Рис. 4: Цепочка состояний кошелька клиента

Публичные данные, сохраняемые контрактом, изображены в заголовках блоков цепочки. Тела блоков состоят из приватных данных, используемых при создании доказательства, но не раскрываемых клиентом, за исключением **nullifier**, который раскрывается при следующей операции.

Диаграмма последовательностей 5 иллюстрирует взаимодействие клиента и контракта при операциях создания кошелька и обмена. После получения сообщения с доказательством и публичными данными от клиента контакт проверяет корректность доказательства. Если доказательство корректно, то выполняется особенная для каждой операции бизнес-логика. В случае создания кошелька это вставка **commitment**-а в дерево Меркла. В случае обмена **nullifier** проверяется на оригинальность и добавляется в список израсходованных **nullifier**-ов. После этого производится обмен заданных токенов на DEX. В зависимости от

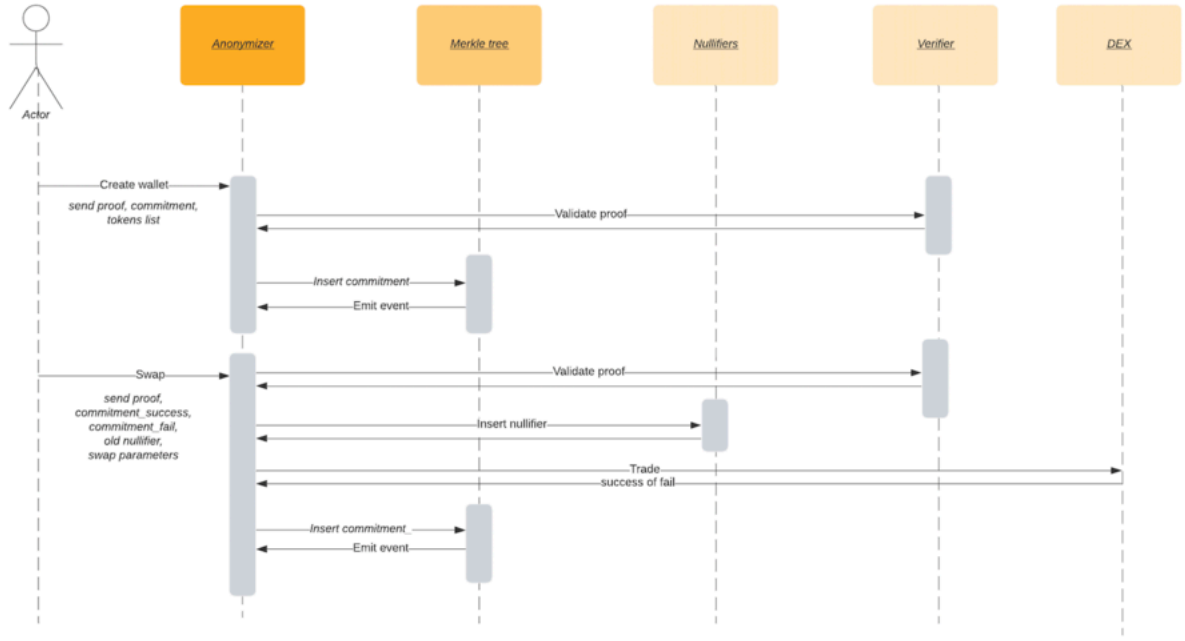

Рис. 5: Диаграмма последовательностей взаимодействия по протоколу при операциях создания кошелька и обмена

успешности обмена в дерево Меркла помещается либо **commitment_success**, либо **commitment_fail**. Операция может быть неуспешной, если с момента генерации доказательства курс обмена на DEX сильно поменяется.

3. Реализация

Раздел включает в себя высокоуровневое описание архитектуры системы, описание архитектуры и детали реализации on-chain модуля — Ethereum смарт-контрактов, а также описание архитектуры и детали реализации клиентского приложения.

3.1. Архитектура системы анонимного обмена активов

Система состоит из нескольких модулей. On-chain модуль, представляющий из себя несколько смарт-контрактов, необходим для валидации доказательств и выполнения фактических операций по обмену токенов на DEX, внесению, или снятию средств, а также хранению информации о дереве Меркла. Второй модуль представляет из себя клиентское приложение. Его цель формировать доказательства, хранить и изменять состояние кошелька, отправлять запросы на on-chain модуль и реагировать на его события.

Также система взаимодействует со вспомогательным модулем Relayer. Его задача — проксировать запросы от клиента на контакт для сокрытия адреса отправителя транзакции. Так как этот модуль не принимает прямого участия в протоколе, а является лишь дополнительным проксирующим звеном, его реализация не входит в данную работу. Схема взаимодействия частей системы изображена на рисунке 6.

При создании кошелька или внесении депозита клиент напрямую взаимодействует с on-chain модулем, самостоятельно посылая транзакции, так как другого способа передать средства на платформе Ethereum нет. При выполнении остальных операций взаимодействие происходит через Relayer.

Перед выполнением операции клиент запрашивает данные о состоянии контракта: актуальный корень дерева Меркла, список его листов. После этого клиент самостоятельно вычисляет путь до необходимого **commitment** в дереве Меркла и с помощью библиотеки snarkjs и клю-

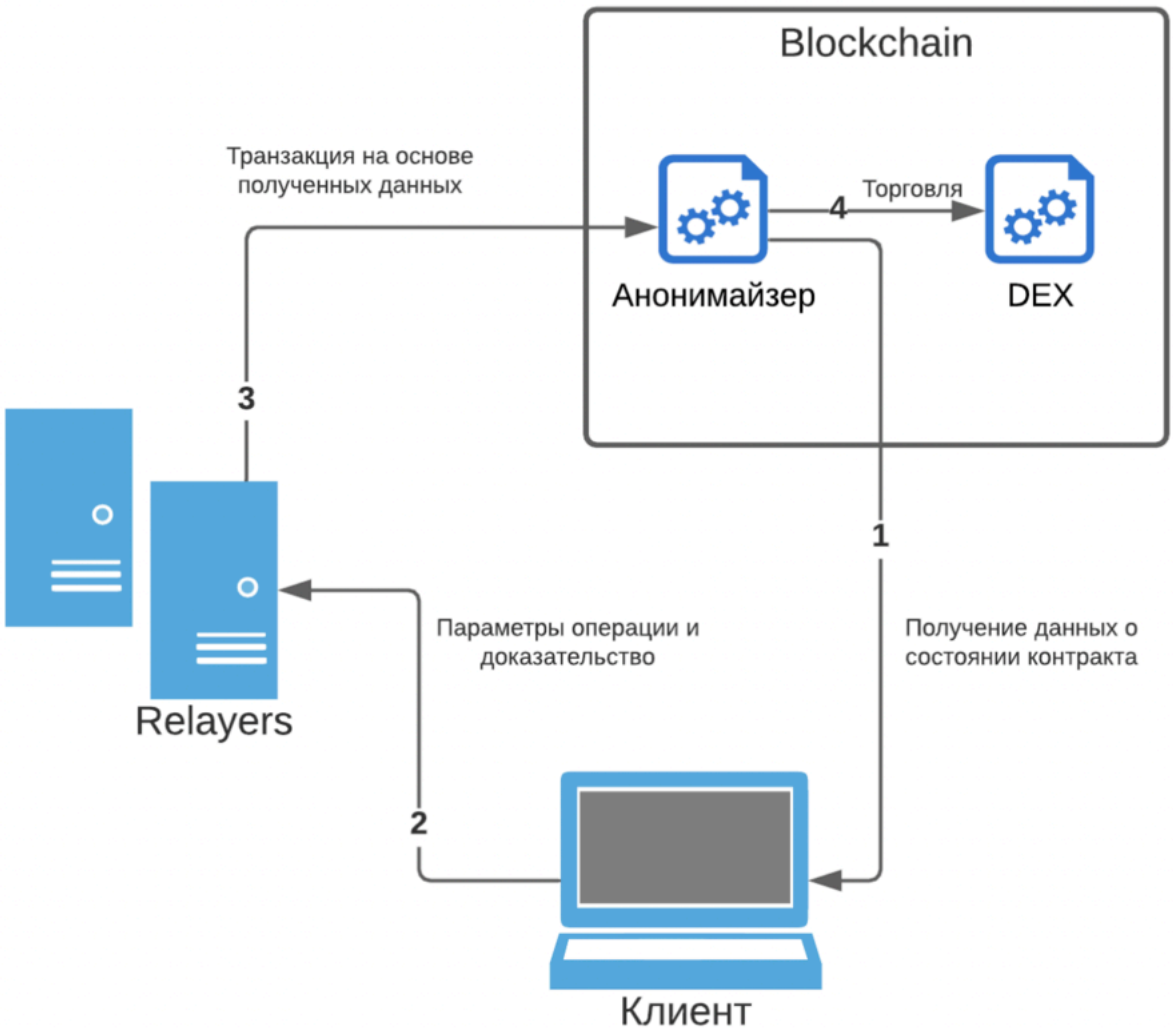

Рис. 6: Взаимодействие частей системы. Изображен процесс проведения операций по торговле, частичному или полному снятию

ча с нулевым разглашением создает доказательство на основе сохраненных данных о состоянии кошелька.

Полученное доказательство посылается на relayer, который формирует транзакцию и отправляет ее на контракт-анонимайзер. Relayer не может поменять параметры операции, так как тогда доказательство будет невалидным. Также relayer не может создать свое доказательство от имени клиента, так как ему не известен соответствующий **secret**. После завершения транзакции контракт-анонимайзер посылает relayer-у комиссию с кошелька клиента в качестве компенсации внутрисетевой комиссии Ethereum.

Контракт-анонимайзер выполняет заданную операцию, изменяя де-

рево Меркла в соответствии с успешностью выполнения. Также генерируется Ethereum-событие, которое помогает клиенту узнать о результате выполнения операции.

3.2. Разработка и реализация on-chain модуля

On-chain модуль представляет из себя несколько смарт-контрактов: основной контракт, предоставляющий внешний интерфейс для клиента, контракт-дерево Меркла, необходимый для имитации данной структуры, контракт с бизнес-логикой операций, и пять контрактов-валидаторов доказательств. Архитектура модуля изображена на рисунке 7. Язык реализации контрактов — Solidity [9]. Реализованные контракты представляют 800 строк кода без учета валидаторов, полученных из Circom описания протокола.

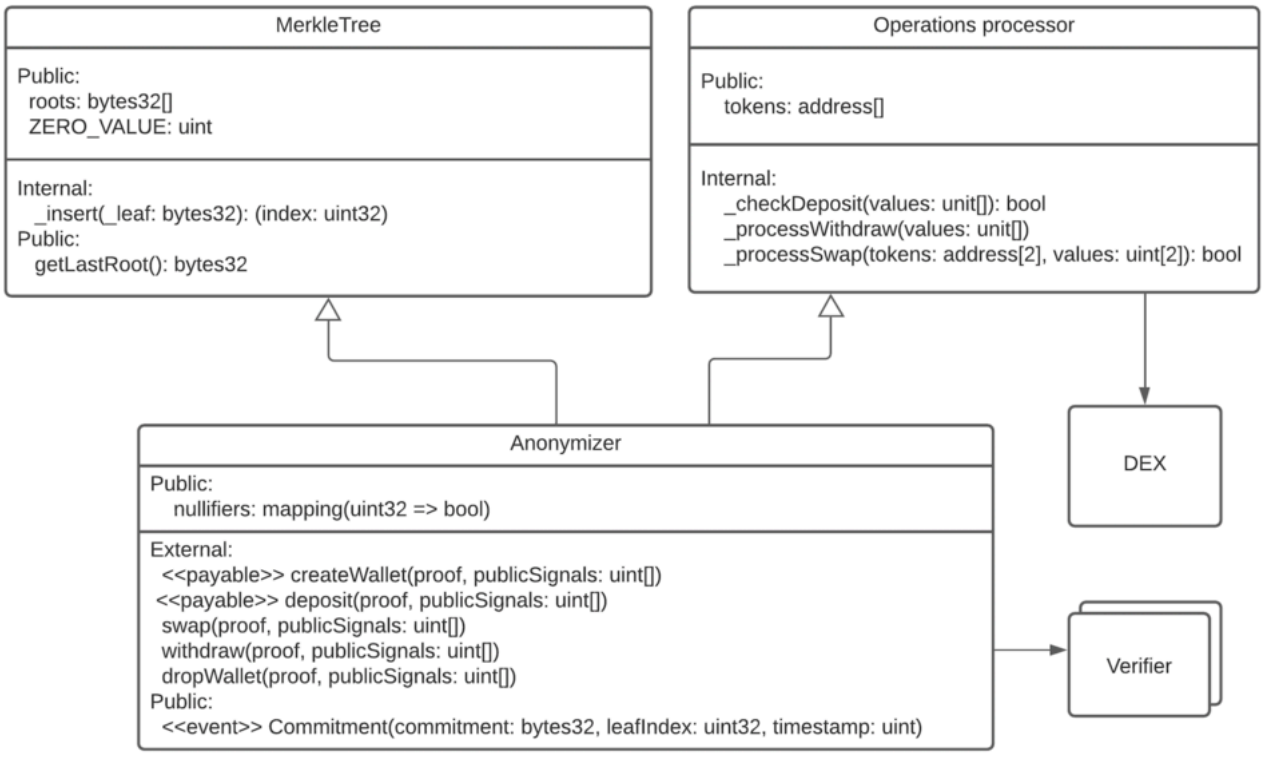

Рис. 7: Архитектура on-chain модуля

Контракт Anonymizer предоставляет внешний интерфейс модуля и управляет остальными частями модуля. Он содержит отображение

[9]https://docs.soliditylang.org/en/v0.8.4/

nullifiers для сохранения израсходованных идентификаторов операции и препятствует ветвлению цепочки. Событие Commitment позволяет не хранить листы дерева на контракте вообще, а хранить только необходимы для вставки следующего узла элементы, так как все произошедшие события сохраняются в блокчейне, и для восстановления дерева локально клиент может легко их получить. Также данное событие позволяет клиенту понять, завершилась операция удачно, или нет.

Контракт MerkleTree реализует функциональность по взаимодействию с деревом Меркла, позволяя добавлять листы в виртуальное дерево, изменяя тем самым его корень. При этом сохраняется несколько последних корней, так как при в процессе генерации и пересылки доказательства клиентом корень дерева может измениться вследствие взаимодействия с другим клиентом.

OperationsProcessor содержит низкоуровневую логику выполнения операций: проверяет факт внесения депозита, производит работу с переводами токенов с адреса пользователя, так как их нельзя прикрепить к транзакции, как стандартную валюту сети. Также этот контракт взаимодействует с DEX, осуществляя обмен токенов, и выполняет работу по снятию средств.

Verifier — это сгенерированные библиотекой snarkjs контракты, единственная задача которых — проверять корректность доказательства.

3.3. Разработка и реализация клиентского приложения

Для взаимодействия по протоколу требуется клиентское приложение. Было решено реализовать его в виде command line программы. При проектировании учитывалась дальнейшая необходимость создания веб-интерфейса, поэтому языком реализации был выбран TypeScript. Этот язык позволяет писать надежный объектно-ориентированный код, кроме того, используется в современных frontend-фреймворках [12], например, в Angular. Программа была разбита на модули для удобства замены интерфейса командной строки на веб-интерфейс. При реализации

клиентского приложения было написано 4500 строк кода.

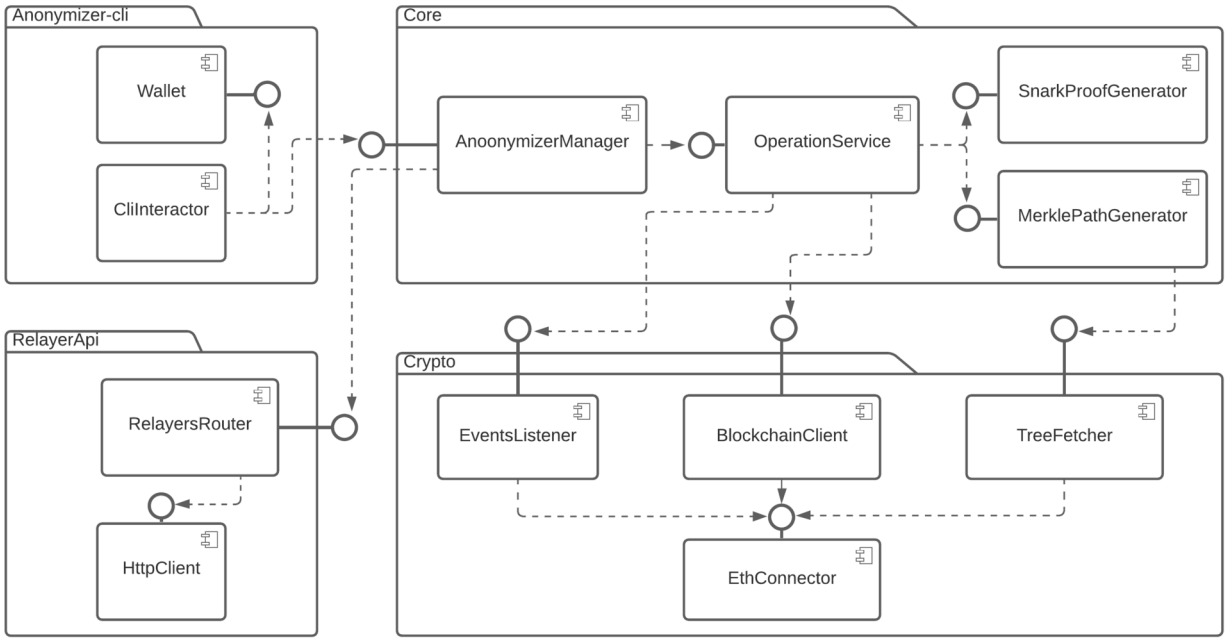

Рис. 8: Архитектура клиентского приложения

На рисунке 8 изображена диаграмма компонентов клиентского приложения.

Модуль Anonymizer-cli обеспечивает доступ к функциональности приложения и хранит кошелек пользователя.

Модуль Core содержит бизнес-логику приложения, формируя доказательства, взаимодействуя с relayer-ами через модуль Realyer-api и блокчейном с помощью модуля Crypto. Компонент AnonymizerManager управляет выполнением программы. OperationSevice представляет из себя набор классов с едиными интерфейсом, выполняющих различные операции протокола. SnarkProofGenerator создает доказательство с помощью библиотеки snarkjs, а MerklePathGenerator воссоздает необходимый для доказательства путь в дереве Меркла.

Модуль Crypto предоставляет функциональность по получению информации из on-chain части анонимайзера, позволяет отправлять транзакции на контракт-анонимайзер при создании кошелька или внесении депозита, а также прослушивает события контракта-анонимайзера, сообщая результат выполнения операции. TreeFetcher проходит по блока-

ми блокчейна, собирая события о добавлении листа, и воссоздает дерево. BlockchainClient предоставляет интерфейс для отправки транзакций. EventsListener прослушивает события добавления листа на контракте-анонимайзере, сообщая об успешности или неудачности операции.

Модуль RelayerApi содержит логику по проксированию операций через relayer-ы для случаев обмена токенов, снятия средств, или удаления кошелька. Перед формированием доказательства данный модуль позволяет выбрать наиболее релевантный relayer и получить его данные: размер взимаемой комиссии **fee** и его блокчейн-адрес **relayer**, используемые при генерации доказательства. Так как данные параметры являются публичными сигналами, хост-relayer сможет убедиться, что в результате выполнения операции будет получена необходимая комиссия на верный адрес. После генерации собранное доказательство отправляется по сети на выбранный до этого relayer, который формирует транзакцию, и посылает ее на контракт-анонимайзер.

4. Тестирование

Для проверки корректности работы отдельных частей системы, а также правильности совместной работы были написаны модульные и интеграционные тесты [18]. В модульных тестах выполняется проверка отдельных конечных сущностей. Интеграционные тесты позволяют проверить взаимодействие совокупности частей системы.

4.1. Тестирование on-chain части

Для апробации и тестирования on-chain модуль был развернут в тестовой сети kovan [10] — это практически полная копия сети Ethereum, но с абсолютно бесплатной валютой.

Модульные тесты контракта OperationsProcessor позволили проверить корректность работы бизнес-логики on-chain модуля для успешных и заведомо неудачных операций.

Модульные тесты контакта Anonymizer позволили убедиться в корректности взаимодействия частей on-chain модуля.

Тестирование и развертывание on-chain части выполнялось с помощью Ethereum фреймворка truffle [14].

4.2. Тестирование клиентского приложения

Тестирование клиентского приложения проводилось с помощью фреймворка Jest [19]. В рамках модульного тестирования было написано 57 тестов, покрывающих 93% бизнес-логики клиентского приложения.

Интеграционные тесты проверили взаимодействие клиентского приложения и on-chain модуля: для поддерживаемых операций с помощью клиентского приложения генерировалось доказательство и параметры операции, и посылалось на контакт-анонимайзер в тестовой сети. После этого проверялось, что нужное количество токенов действительно было переведено, обменено, или снято, а также, что корректный commitment

[10] https://kovan-testnet.github.io/website/proposal/

был сохранен в дереве Меркла. Интеграционные тесты позволили смоделировать последовательное выполнение различных операций. Для тестирования операций обмена токенов, снятия средств и удаления кошелька был создан имитирующий работу relayer-а сервис.

Заключение

В рамках данной работы были достигнуты следующие результаты:

- Проведен обзор предметной области и анализ существующих решений
- Разработан протокол передачи и валидации данных с нулевым разглашением
  - Для случаев создания кошелька и внесения депозита
  - Для случая обмена активов
  - Для случая частичного или полного снятия средств
- Разработана архитектура системы анонимного обмена активов и реализованы взаимодействующие по протоколу части разработанной системы
- Проведено тестирование реализованных частей системы

Результаты данной работы будут использованы в разрабатываемом компанией ООО «Манзони» децентрализованном финансовом сервисе «Humpty Dumpty» [Приложение А].

Список литературы

[1] Amir Pasha Motamed Behnam Bahrak. Quantitative analysis of cryptocurrencies transaction grap. — SpringerOpen, 2019. — URL: https://appliednetsci.springeropen.com/articles/10.1007/s41109-019-0249-6 (online; accessed: 25.04.2021).

[2] Becker Georg. Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis. — Seminararbeit Ruhr-Universit¨at Bochum, 2008. — P. 8 – 11. — URL: https://www.emsec.ruhr-uni-bochum.de/media/crypto/attachments/files/2011/04/becker_1.pdf (online; accessed: 25.04.2021).

[3] Behrens Alexander. Ethereum DEX trading hits nearly a half billion in 24-hour volume. — 2020. — URL: https://decrypt.co/38503/ethereum-dex-trading-half-billion-24-hour-volume (online; accessed: 25.04.2021).

[4] Benefits of anonymous cryptocurrency. — 2019. — URL: https://blog.incognito.org/5-benefits-of-anonymous-cryptocurrency-019/ (online; accessed: 25.04.2021).

[5] Biggest cryptocurrencies in the world based on 24h volume. — 2021. — URL: https://www.statista.com/statistics/655511/leading-virtual-currencies-globally-by-purchase-volume/ (online; accessed: 25.04.2021).

[6] Blockchair: Ethereum transaction count. — 2021. — URL: https://blockchair.com/ru/ethereum/charts/transaction-count?granularity=month (online; accessed: 25.04.2021).

[7] DEX trade volume. — 2020. — URL: https://www.theblockcrypto.com/data/decentralized-finance/dex-non-custodial (online; accessed: 25.04.2021).

[8] David Schwartz Maria Bell. The circom language. — 2018. — URL: https://docs.circom.io/ (online; accessed: 25.04.2021).

[9] Fabian Vogelsteller Vitalik Buterin. EIP-20: ERC-20 Token Standard. — 2015. — URL: https://eips.ethereum.org/EIPS/eip-20 (online; accessed: 25.04.2021).

[10] Gabizon Ariel. Explaining SNARKs. — 2017. — URL: https://electriccoin.co/blog/snark-explain/ (online; accessed: 25.04.2021).

[11] JUHA PARTALA TRI HONG NGUYEN2 SUSANNA PIRTTIKANGAS. Non-Interactive Zero-Knowledge for Blockchain: A Survey. — IEEE Access, 2020. — P. 227945 – 227947. — URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9300214 (online; accessed: 25.04.2021).

[12] Marin Kaluža Krešimir Troskot Bernard Vukelić. COMPARISON OF FRONT-END FRAMEWORKS FOR WEB APPLICATIONS DEVELOPMENT. — Zbornik Veleučilišta u Rijeci, 2018. — P. 268 – 270. — URL: https://hrcak.srce.hr/199922 (online; accessed: 25.04.2021).

[13] Monika di Angelo Gernot Salzer. Tokens, Types, and Standards: Identification and Utilization in Ethereum. — 2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS), 2020. — P. 6. — URL: https://ieeexplore.ieee.org/abstract/document/9126009 (online; accessed: 25.04.2021).

[14] Pieter Hartel Mark van Staalduinen. Truffle tests for free - Replaying Ethereum smart contracts for transparency. — arXiv, 2018. — P. 2 – 5. — URL: https://arxiv.org/pdf/1907.09208.pdf (online; accessed: 25.04.2021).

[15] SETH SHOBHIT. Private Cryptocurrencies. — 2021. — URL: https://www.investopedia.com/tech/

five-most-private-cryptocurrencies/ (online; accessed: 25.04.2021).

[16] William Entriken Dieter Shirley Jacob Evans Nastassia Sachs. EIP-721: ERC-721 Token Standard. — 2018. — URL: https://eips.ethereum.org/EIPS/eip-721 (online; accessed: 25.04.2021).

[17] Yuen C Lo Francesca Medda. Uniswap and the rise of the decentralized exchange. — MPRA, 2020. — P. 1 – 4. — URL: https://mpra.ub.uni-muenchen.de/103925/ (online; accessed: 25.04.2021).

[18] А.А. Пивень Ю.И. Скорин. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. — Харьковский национальный экономический университет, 2012. — С. 56 – 58. — URL: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/soi_2012_1_4_14.pdf (дата обращения: 25.04.2021).

[19] Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. 2-е издание. — Издательский дом Питер, 2021. — С. 70 – 83. — URL: https://arxiv.org/pdf/1907.09208.pdf (дата обращения: 25.04.2021).

[20] Кириллов И. Как технология блокчейн меняет современный мир. — СиБ, 2019. — С. 62 – 63. — URL: http://sib.com.ua/sib-02-99-2018/bitkoin.html (дата обращения: 25.04.2021).

Приложение А. Справка о внедрении

Генеральный директор
ООО « Манзони »
Ершов Дмитрий Николаевич

СПРАВКА

о внедрении результатов выпускной квалификационной работы бакалавра Андреева Сергея Ивановича на тему «Разработка и реализация протокола с нулевым разглашением для решения задачи анонимной передачи данных в Ethereum-подобных блокчейн сетях».

Настоящей справкой подтверждается, что разработки Андреева Сергея Ивановича будут использованы в разрабатываемом компанией ООО « Манзони » продукте — децентрализованном финансовом сервисе «Humpty Dumpty».

Разработанный Андреевым Сергеем Ивановичем в рамках выпускной квалификационной работы бакалавра протокол анонимной передачи данных позволит осуществлять анонимные торги на децентрализованном финансовом сервисе «Humpty Dumpty».

Генеральный директор ООО « Манзони » Ершов Д.Н.

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
LIMITED LIABILITY COMPANY
«МАНЗОНИ»
«MANZONI»
SAINT-PETERSBURG
САНКТ-ПЕТЕРБУРГ

10 мая 2021 г.