ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ
И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ИНФОРМАЦИОННЫХ СИСТЕМ

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ГОЛОСОВАНИЯ НА ОСНОВЕ ТЕХНОЛОГИИ БЛОКЧЕЙН

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.04.01 Математика и компьютерные науки, группы 07001631
Гребенник Олега Геннадьевича

Научный руководитель
к.т.н., доцент
Муромцев В.В.

Рецензент
Заведующий кафедрой ПИиИТ,
к.т.н., профессор Ломакин В.В

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В ближайшее десятилетие цифровые денежные средства и технология блокчейн потенциально могут изменить мир так, как когда-то его изменило изобретение интернета, а в свое время появление компьютера или электричества.

Блокчейн – это технология, которая позволит сократить практически до нуля половину имеющихся сегодня транзакционных издержек в масштабе компании, холдинга, рынка, страны и всего глобального сообщества.

В современном мире, взаимосвязанном и объединенном глобальной сетью, экономическая деятельность осуществляется посредством коммерческих сетей, которые стирают национальные, географические границы и границы юрисдикций. Как правило, такие сети переплетаются на торговых площадках, где производители, потребители, поставщики, партнеры, активные участники рынка или посредники, а также прочие заинтересованные лица владеют, управляют ценностями, известными под названием активы, а также реализуют свои права и привилегии на них [1].

Активы могут быть материальными и физическими, как, например, машины и дома, либо нематериальными и виртуальными, — как сертификаты на акции и патенты. Получение права собственности на активы и их передачу, известную как транзакция, создает ценность коммерческих сетей.

Как правило, участниками транзакций являются различные покупатели, продавцы и посредники (например, банки, аудиторы или нотариусы), коммерческие соглашения и контракты между которыми вносятся в разнообразные реестры. В коммерческой деятельности, как правило, используется несколько реестров для ведения учета активов, находящихся в собственности, и активов, передаваемых участниками друг

другу в различных видах деятельности. Реестры являются системами учета экономической деятельности и интересов предприятий [2].

Реестры, использующиеся сегодня в предпринимательской деятельности, во многом несовершенны. Они неэффективны, дорогостоящи, а их функционирование непрозрачно и подвержено мошенническим манипуляциям и неправомерным действиям. Эти проблемы являются следствием использования сторонних централизованных систем, основанных на доверии, таких как: финансовые, расчетно-клиринговые организации и другие посредники существующих организационных структур [3].

Такие централизованные системы реестров создают своего рода помехи и препятствия, растягивающие время выполнения транзакций. Недостаточная прозрачность их работы, а также подверженность коррупции и мошенничеству приводят к возникновению споров. При этом их урегулирование, совершение обратных сделок и страхование транзакций довольно затратно и в средствах, и во времени — все эти риски и неопределенности приводят к упущенным возможностям для бизнеса.

Неупорядоченные копии реестров, используемые в собственных системах каждого участника, становятся причиной принятия ошибочных коммерческих решений на основе временных недостоверных данных. В лучшем случае принятие решения на основе актуальной информации откладывается на время приведения в соответствие отличающихся копий реестров [26].

Блокчейн был изобретен для осуществления экономической деятельности людей независимо от банков и государств. Но сегодня государства, корпорации и банки являются одними из крупнейших локомотивов внедрения этой технологии в повседневную жизнь.

Пусть эти структуры не заинтересованы в полной имплементации парадигм распределенного реестра, но они видят, как блокчейн и умные контракты могут увеличить эффективность работы даже централизованных структур.

Пример применения смарт-контрактов в повседневной жизни общества — приближение эры цифрового государства.

Выборы в каждой стране — это одна из самых важных и трудоемких задач, когда необходимо в жестко ограниченное время получить и обработать информацию от миллионов граждан. Существующие системы даже в наилучшем воплощении могут давать сбои. Но смарт-контракты убирают из выборного процесса практически любой риск [19].

Многие страны мира, включая ЕС, Австралию, Россию и Украину, всерьез рассматривают переход на блокчейн-голосование. Не исключено, что в ближайшем будущем мы будем выбирать государственных лидеров, не опуская бумажки в урны, а запуская смарт-контракт [34].

К тому же блокчейн-голосование может помочь и с проблемой явки избирателей там, где инерция вызвана длинными очередями, заполнением множества бумаг и прочей волокитой.

Исходя из всего вышесказанного, целью выпускной квалификационной работы будет разработка автоматизированной системы голосования на основе технологии блокчейн. Данная система позволит проводить различного вида голосования без возможности фальсификации результатов.

Для достижения цели были поставлены следующие задачи:

- анализ существующих проблем при проведении всевозможных голосований;
- анализ возможностей технологии блокчейн;
- формирование требований к системе, постановка задачи на разработку;
- разработка архитектуры системы;
- проектирование и реализация «токена»;
- проектирование и реализация умного контракта для проведения голосования;
- конфигурация и подготовка системы к деплойменту;
- конфигурация и развертывание приватной блокчейн сети;

- тестирование разработанной системы.

В первой главе проводится описание технологии блокчейн, её преимуществах и недостатках при использовании в качестве основы для разработки автоматизированной системы голосования. Также с данной рассматривается платформа Ethereum на базе, которой будет разрабатываться система голосования. Кроме того, в данной главе описывается концепция «умных контрактов» являющихся основным инструментом для разработки системы.

Во второй главе проводится проектирование архитектуры системы, разработка «умного контракты», а также «токена» позволяющего проводить голосование. Также в данной главе проводится детально описание технологии блокчейн и механизмов, позволяющих использовать её в качестве основы для системы голосования.

В третьей главе описаны настройка и тестирование системы, а также приведено описание процесса создания приватной сети и настройки провайдера Metamask.

В заключении сделан вывод о степени достижения поставленных целей и задач.

Данная выпускная квалификационная работа содержит 68 страниц, 35 рисунков, 4 листинга кода и 1 приложение.

ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Блокчейн

Первая попытка решить проблему безопасности платежей в интернете была предпринята в 1981 году. Главные вызовы заключались в том, что пользователь вынужден сообщать о себе слишком много сведений. Плюс всегда была третья сторона. Это значит, существовал контрагентский риск и необходимость платить комиссию [23].

Только в 1993 году была создана первая цифровая платежная система, которая была относительно безопасна и позволяла анонимно проводить платежи онлайн—eCash. Однако по прошествии 5 лет компания обанкротилась—тогда пользователей мало волновала безопасность и приватность платежей онлайн [35].

В 2008г. произошел шторм в глобальной финансовой индустрии. Некто по имени Сатоши Накамото (Satoshi Nakamoto) публикует статью, где описывает новый протокол электронной системы платежей. Он создан для пиринговой сети [18].

Пиринговая сеть основана на равноправии участников. В ней отсутствуют выделенные серверы, а каждый узел (peer) выступает и как клиент, и как сервер [7].

Вскоре Накамото выпустил и первое приложение для обмена биткоинами. Сеть стала стремительно набирать последователей. Ведь такого раньше не было, чтобы люди могли обмениваться деньгами безопасно, приватно и без третьих лиц.

Таким образом, биткоин заложил мощный фундамент, на котором стремительно развивается технология блокчейн. Код приложения находится в открытом доступе. Любой может скачать его бесплатно, запустить или использовать для создания новых приложений. Их число ничем не ограничено.

Это открывает огромные перспективы для внедрения технологии в различных сферах жизни.

1.1.1 Описание технологии

Блокчейн — это защищенный от несанкционированного доступа цифровой реестр общего пользования, который ведет учет транзакций в публичной или закрытой одноранговой сети. Распределенный между всеми узлами сети реестр непрерывно записывает историю операций с активами между одноранговыми (одного порядка) узлами сети в виде блоков информации [2].

Структура системы построенной на базе технологии блокчейн представлена на рис. 1.1.

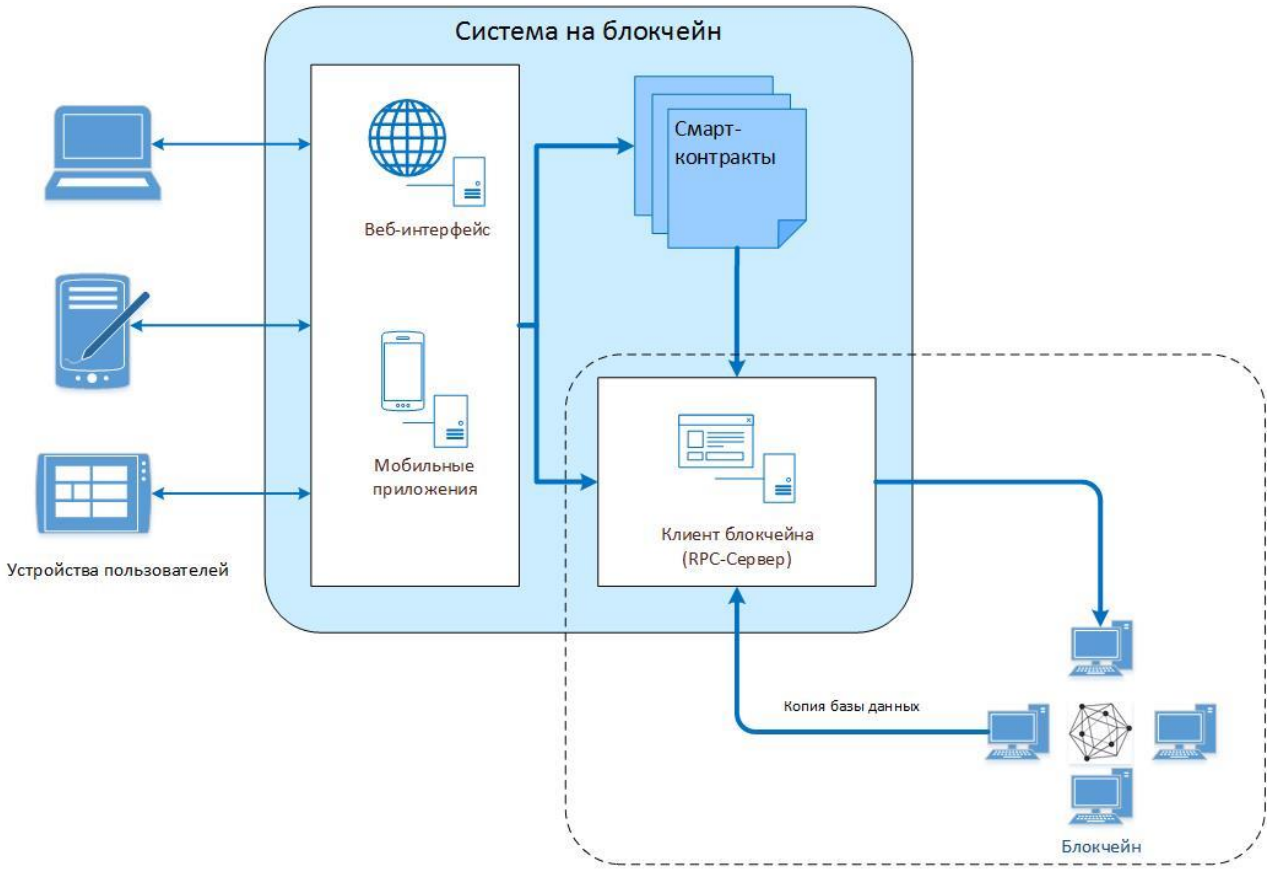

Рис. 1.1. Структура блокчейн системы

Все утвержденные блоки транзакций соединяются в цепочку — с начального блока до последнего добавленного, отсюда и название технологии — блокчейн (англ. block chain — цепочка блоков). Таким образом, блокчейн выступает в качестве единого источника достоверных данных, а участники блокчейн-цепи видят только те транзакции, которые относятся именно к ним [19].

Вместо того чтобы обращаться к третьим лицам, например, финансово-кредитным организациям, в качестве посредников при проведении транзакций, узлы блокчейн-сети используют специальный протокол консенсуса для согласования содержимого реестра, а также криптографические алгоритмы хеширования и электронно-цифровые подписи для обеспечения целостности транзакции и передачи ее параметров [20].

Механизм консенсуса гарантирует, что распределенные реестры являются точными копиями, что снижает риск появления мошеннических транзакций, поскольку постороннее вмешательство может возникнуть во многих местах одновременно. Криптографические алгоритмы хеширования, такие как алгоритм вычислений SHA256, гарантируют, что любое изменение входных данных транзакции, даже самое незначительное, приведет к появлению другого значения хеша в результатах расчетов, что указывает на вероятность компрометации входных данных транзакции. Электронно-цифровые подписи гарантируют, что транзакции осуществляются легитимными отправителями (подписаны закрытыми ключами), а не злоумышленниками [34].

Децентрализованная одноранговая блокчейн-сеть лишает отдельных участников или групп участников возможности контролировать базовую инфраструктуру или дестабилизировать всю систему [17]. Все участники сети равны и подключаются к ней по одним и тем же протоколам. Участниками могут быть физические лица, государственные структуры, организации или объединения всех перечисленных типов участников.

По сути система записывает хронологический порядок проведения транзакций со всеми узлами сети, признавшими действительность транзакций посредством выбранной модели консенсуса. Результатом являются не подлежащие отмене транзакции, согласованные всеми участниками сети децентрализованно.

Blockchain—это технология, позволяющая участникам системы передавать активы друг другу надёжным способом, не требующим участия посредников [11].

Блокчейн является потенциально революционной технологией в том плане, что участники обмена любыми активами освобождаются от необходимости наличия доверия между ними и от необходимости наличия централизованного регулирующего органа. Распределенный журнал данных на основе блокчейн способен гарантировать неизменность, целостность и надежность любых дискретных единиц в системе, где стороны не обязаны доверять друг другу [19]. В традиционном протоколе блокчейна все транзакции записываются в публичный реестр и доступны для просмотра уполномоченным участникам. Как только транзакция является совершенной, она подтверждается всеми участниками сети и блок с параметрами этой транзакции моментально добавляется в блокчейн [12]. Последний добавленный блок содержит информацию не только о последней транзакции, но и обо всех предыдущих, что делает практически невозможным удаление или изменение ранее добавленного блока, тогда как пришлось бы изменять все последующие. Схема работы блокчейна представлена на рис. 1.2.

Например, в blockchain можно хранить записи о денежных переводах. И в криптовалютах blockchain как раз используется для фиксирования информации о том, кто, кому и сколько виртуальных денег перевел. Однако в blockchain можно хранить и другие активы. В общем, все, что можно записать на бумаге, можно записать и в blockchain с одним лишь отличием—в blockchain просто невозможно подменить или подделать записи [16].

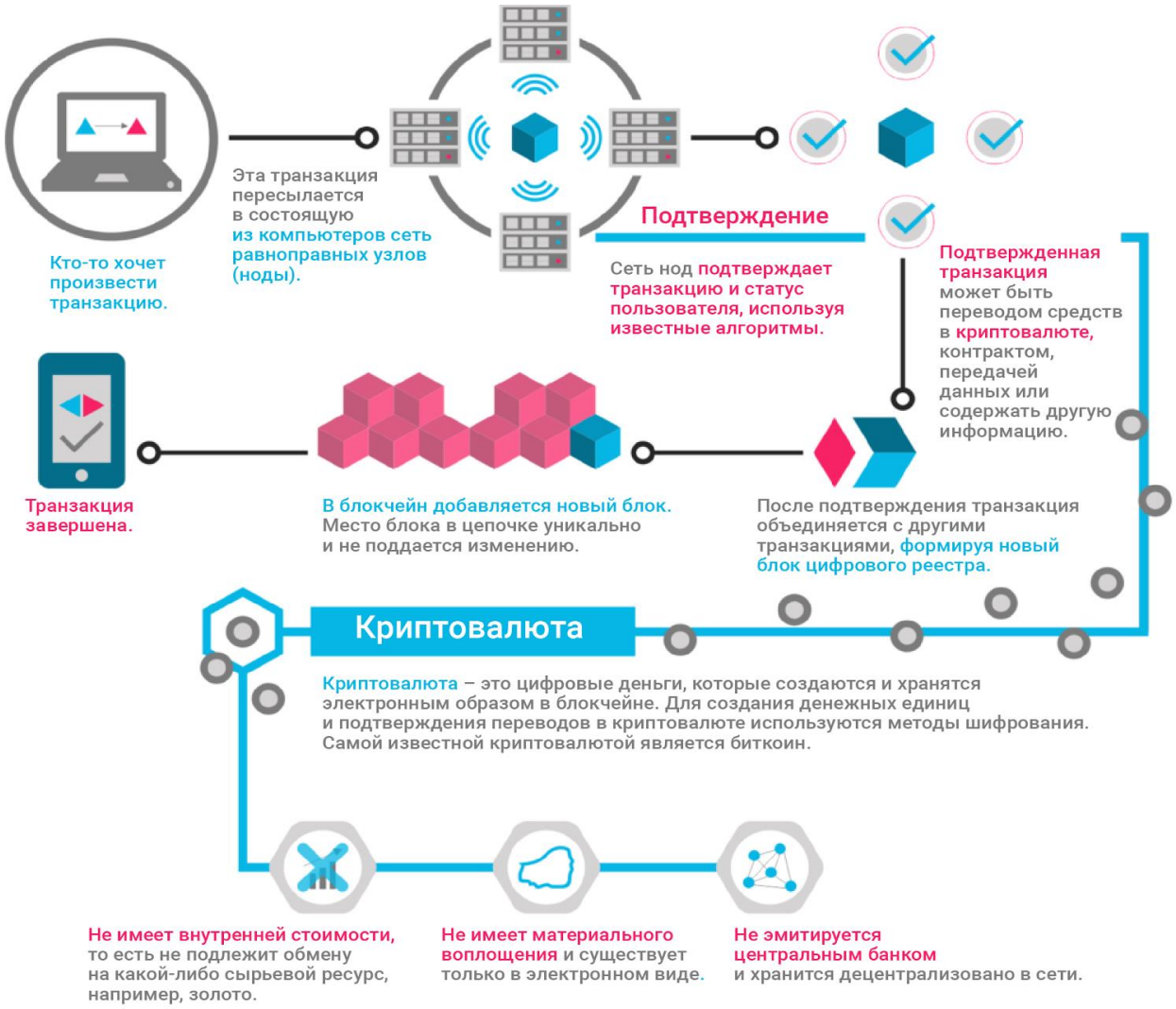

Рис. 1.2. Схема работы блокчейна

В основе blockchain лежат семь принципов:

1. Сетевая целостность. Доверие возникает изнутри системы, оно не продиктовано извне. Участники честны на словах и на деле, уважают интересы других, готовы отвечать за последствия своих действий, их решения прозрачны. Целостность закодирована в каждом шаге процесса и распределена между всеми участниками, а не принадлежит одному [25].
2. Распределение нагрузки. Энергозатраты распределены по всей пиринговой сети. Нет одного выключателя. Ни один из участников не может выключить систему. Если центральный орган заблокирует человека или группу, система продолжит работать [34]. Если примерно половина сети

попытается взять контроль над всей сетью, остальные увидят, что происходит.

3. Ценность как стимул. Система выравнивает стимулы всех заинтересованных сторон. Сатоши Никамото запрограммировал приложение таким образом, чтобы вознаграждать тех, кто развивает программу. Активный пользователь сети получает 50 биткоинов за каждый завершенный блок по прошествии 4 лет. Через следующие 4 года—25 биткоинов, затем 12.5 и т.д. [38].
4. Безопасность. Каждый участник сети должен использовать шифрование. Меры безопасности встроены в сеть. Они предоставляют конфиденциальность и подлинность. У пользователя два ключа: один для шифрования, другой для дешифрования. Метод получил название «инфраструктура открытых ключей» (PKI).
5. Приватность. Люди должны контролировать свои данные. У людей должно быть право решать, какие сведения, когда, как и в каком объеме сообщать о себе [2].
6. Защищенность прав. Права собственников прозрачны и закреплены. Фиксирование времени транзакции и PKI не только исключают двойное расходование, но и фиксируют право собственности на каждый биткоин в сети. Мы можем торговать только тем, что принадлежит нам [17]. Кроме биткоин, это может быть любая ценность, в том числе и интеллектуальное право.
7. Вовлеченность. Экономика работает лучше, когда она работает для каждого. Это означает снижение входного барьера. Любой, у кого есть сотовый телефон, может участвовать в рынке как производитель или потребитель.

Преимущества использования блокчейна:

- В традиционных коммерческих сетях все участники обеспечивают поддержку собственных дублируемых реестров, расхождения между которыми приводят к возникновению споров, увеличивают время

выполнения расчетов, а также требуют привлечения посредников со всеми сопутствующими расходами [27]. В то же время использование распределенных реестров на основе технологии блокчейн, в которых транзакции не могут быть изменены после принятия консенсуса и внесения в реестр, может сэкономить предпринимателям время и деньги, а также снизить возможные риски.

- Блокчейн-технологии сулят более высокую прозрачность взаимодействия между заинтересованными участниками, улучшенную автоматизацию, адаптацию реестров под индивидуальные требования, а также более высокий уровень доверия к ведению учета [14]. Механизмы консенсуса в блокчейне имеют преимущества консолидированного и упорядоченного массива данных, имеющего меньший процент погрешностей и квазиреальные справочные данные, и позволяющего участникам вносить изменения в описания принадлежащих им активов.
- Поскольку ни один участник не владеет центральным источником происхождения информации, содержащейся в распределенном реестре, блокчейн-технологии повышают уровень доверия и обеспечивают целостность информационного потока между участниками [29].
- Неизменность механизмов блокчейна приводит к снижению затрат на аудит и повышению прозрачности соблюдения нормативных требований. А поскольку контракты, заключаемые в коммерческих сетях на базе блокчейн-технологий, являются интеллектуальными, автоматизированными и окончательными, бизнес только выигрывает от высокой скорости выполнения, снижения затрат и рисков, а также своевременных расчетов по контрактам.
- Для того чтобы блокчейн стал эффективным решением, требуется сеть. Однако сеть бывает разных видов. Это может быть сеть между организациями в виде производственно-сбытовой цепочки, либо сеть

внутри одной организации. Внутри организации блокчейн-сеть можно использовать для распространения данных между отделами или, как вариант, для создания сети аудита или корпоративного контроля [31]. Помимо этого, сеть может существовать и между отдельными людьми, которым, к примеру, необходимо хранить данные, цифровые активы или контракты в блокчейне.

1.1.2 Голосование и блокчейн

Голосование с использованием технологии блокчейн позволяет решить множество проблем, которые возникают при использовании любого другого вида «опросов». Использование данной технологии предоставляет:

- Возможность для пользователей сети участвовать в общественной жизни прямо со своего ноутбука (или смартфона, или игровой платформы). При таком подходе в этот процесс могут включиться все, кто в силу своей трудовой деятельности или необходимости заботиться о ребенке и выполнять семейные обязательства, оказывается слишком вымотан или просто не успевает посетить собрание, где принимаются решения [22].
- Предотвращение перехвата инициативы одним из участников, пытающимся воспользоваться собранием в личных интересах.
- Отсутствие необходимости собирать всех заинтересованных лиц в определенном месте и в определенное время для принятия решений по повседневным вопросам. Таким образом, люди смогли бы получить возможность принимать участие в общественной жизни, когда им это удобно и так, как им это удобно [6].
- Возможность сравнить решения, принятые разными органами или юрисдикциями, или сопоставить их с результатами прошлых местных голосований. Все детали общественного процесса самоопределения

могут быть также доступны всем желающим. Каждый участник может проверить как собственный голос, так и удостовериться в точности учета голосов других участников [23].

Использование технологии блокчейн в последнее время набирает популярность среди всевозможного вида голосований:

- Национальный расчетный депозитарий РФ успешно протестировал прототип системы электронного голосования для собраний владельцев облигаций e-proxy voting на основе технологии блокчейн. Прототип системы e-proxy voting разработан на базе сетевой распределенной криптографической платформы NXT и использует международный стандарт ISO 20022 для обмена сообщениями. Разработка велась совместно с компанией DSX Technologies [34].
- В Эстонии: Акционеры компаний, размещённых на Таллиннской Фондовой Бирже Nasdaq, в скором времени получат возможность голосовать во время собраний акционеров при помощи новой системы голосования e-voting, основанной на технологии блокчейн. Данная система разработана в Nasdaq, став вторым по счёту блокчейн-проектом биржи, и её реализация стала возможной в результате партнёрства с платформой электронного гражданства e-Residency, работающей при поддержке правительства Эстонии [22].
- В Сьерра-Леоне (Западная Африка) прошли первые в истории выборы с применением технологии блокчейн. Техническое решение предоставил швейцарский фонд Agora, занимающийся цифровыми технологиями в голосовании. Информация с каждого поданного бумажного бюллетеня записывалась в блокчейн. Agora давала гарантию прозрачности выборов. Швейцарский стартап работает с частным (проприетарным) блокчейном. Это означает, что правом вносить записи в реестр обладают лишь специальные уполномоченные органы. В случае выборов в Сьерра-Леоне ими выступили Красный Крест, Высшая техническая школа Лозанны и Универститет Фрайбурга. Остальные люди,

желающие выступить наблюдателями, могли следить за процессом в режиме «только для чтения» [30].

1.2 Платформа Ethereum

Создание в 2009 году Биткойна привело к возникновению социального и технологического движения, оказавшего серьёзное влияние на всю финансовую отрасль [17].

Многообещающая технология, известная под названиями «блокчейн» и «распределённая книга учёта». Это была первая технология, позволившая безопасно отправлять деньги через Интернет, не боясь обмана или цензуры. К счастью, несколько пионеров отрасли смогли разглядеть скрытый потенциал технологии блокчейна, которая лежала в основе этой безопасной платёжной системы [13].

То, что изначально позволило многим людям отправлять и получать деньги, также было способно децентрализовать и изменить Интернет.

Но сперва необходимо было решить одну проблему: Биткойн не был создан для передачи больше нескольких килобайт данных за одну транзакцию, не способен он был и выполнять вычисления, не предусмотренные его ограниченным программным языком. Сатоши Накамото, создатель Биткойна, верил, что ограниченная функциональность системы положительно скажется на её безопасности [17].

Яркий пример реализации идеологии блокчейн—компания Ethereum. Ее соосновал канадский программист Виталик Бутерин, родившийся в подмосковной Коломне. Виталик представлял Ethereum как «мировой компьютер»; он должен был связать воедино виртуальную машину (EVM), язык программирования (Solidity), токен (ETH) и топливо («gas») для обеспечения транзакций внутри сети. Эта комбинация позволяет создавать

сложные программно-вычислительные инструкции — широко известные как смарт-контракты и децентрализованные приложения (DApp) [25].

1.3 Умные контракты

Сегодня существует отдельный тип юристов, который занимается составлением и сопровождением контрактов. Такие контракты написаны юридическим языком, содержат большое количество страниц и не всегда до конца понятны подписантам.

Традиционные контракты не только сложны в составлении, но и требуют привлечения третьих лиц для обеспечения их соблюдения. В случае разночтений, стороны вынуждены обращаться в суды, что отнимает еще больше времени и денег [16].

С приближением цифровой эры, дигитализация затронула и эту важную часть общественных взаимоотношений. В 1994 году юрист и криптограф Ник Сабо описал концепцию умных контрактов (smart contracts), определив такой контракт как “электронный протокол передачи информации, обеспечивающий исполнение сторонами условий контракта”.

По мнению автора концепции, смарт-контракты позволили бы обеспечивать автоматическое выполнение условий сделок (производство выплат, конфиденциальность и даже принудительное исполнение обязательств сторон) с минимальными затратами на их сопровождение и без необходимости привлечения третьих лиц для обеспечения доверия [26].

Хотя технология, способная поддерживать смарт-контракты, с тех пор заметно развилась, предложенное Сабо определение и сейчас точно выражает суть понятия.

Возникновение технологии блокчейн открыло перспективу для создания систем, позволяющих заключать и автоматически исполнять сделки по

достижении заранее заданных условий, минуя централизованных посредников.

В отличие от юридического языка бумажных договоров, код не подвержен лингвистическим нюансам и двойным толкованиям. Поскольку смарт-контракты являются программами и создаются на основе компьютерной логики, стороны сделки могут быть уверены, что условия, прописанные в коде контракта, будут соблюдены неукоснительно и не могут быть изменены задним числом. В обиходе это правило формулируется кратко: «код — это закон» [27]. На рис. 1.3. представлены некоторый преимущества умных контрактов над традиционными контрактами.

Рис. 1.3. Преимущества умных контрактов

Тем не менее отказ от услуг централизованных посредников и автономное исполнение смарт-контрактов позволяют существенно экономить на обеспечении честности их соблюдения. Так как любой отдельно взятый посредник может оказаться заинтересованным в том или ином исходе сделки,

а суммы на кону могут быть немаленькими, стоимость услуг доверенных лиц зачастую может оказаться достаточно высокой.

Поэтому важной особенностью умных контрактов на блокчейне является децентрализованное исполнение [35]. Условия, необходимые для соблюдения умных контрактов, распространяются по распределенной сети блокчейна с помощью тех же механизмов, которые передают информацию об обычных транзакциях. Когда компьютеры в сети получают информацию о контракте, каждый из них приходит к независимому решению относительно выполнения условий контракта, после чего сверяется с остальными узлами сети. Таким образом, ни одна сторона не может самостоятельно повлиять на решение, поскольку исполнение сделки находится в руках всей системы целиком. Схема работы смарт-контракта представлена на рис. 1.4.

Рис. 1.4. Схема работы смарт-контракта

Объектами смарт-контракта выступают:

- подписанты (от 2 сторон) — участники договора, которые подтверждают свое участие электронной подписью;

- предмет договора — объект, который находится внутри системы умного контракта, например, криптовалюта, или к которому программа имеет беспрепятственный доступ без участия человека. В будущем все большее количество предметов и вещей будут подключаться к интернету [40];
- условия — алгоритм в виде четкого математического описания, которое имеет ясную логику и последовательность.

Для того, чтобы умные контракты могли существовать, требуются определённые условия:

1. Использование широко распространенных методов электронной подписи на основе публичных и приватных ключей (асимметричное шифрование).
2. Существование открытых, децентрализованных и доверительных сторонам контракта баз данных для исполняемых транзакций, работа которых полностью исключает человеческий фактор. Как пример: блокчейн в Bitcoin.
3. Децентрализация среды исполнения умного контракта. Как пример: Ethereum, Codius, Counterparty [30].
4. Достоверность источника цифровых данных. Как пример: корневые центры сертификации SSL в базах современных интернет-браузеров [8].

Если рассматривать смарт-контракты при использовании в разных сферах жизнедеятельности, то можно выделить ряд важных преимуществ:

- независимость — больше не нужно прибегать к услугам посредников для заключения сделок;
- безопасность — смарт-контракт находится в распределенном реестре, его условия нельзя изменить;
- экономия — избавляясь от посредников, стороны умного контракта могут сотрудничать на более выгодных условиях;
- отсутствие издержек — в случае выполнения условий контракты, стороны сразу обмениваются активами [3].

Умные контракты нельзя назвать идеальным инструментом для построения взаимоотношений между людьми. У них тоже есть несколько недостатков. Среди минусов:

- правовой статус — для работы умных контрактов, используется криптовалюта, а ее пока не принимают в качестве официального финансового инструмента;
- ошибки — для составления умного контракта нужно прописывать всевозможные условия и варианты развития сделок, чем сложнее процесс, тем труднее создать смарт-контракт;
- отсутствие понимания — большинство пользователей пока слабо понимают, что собой представляют смарт-контракты.

1.4 Постановка задачи

Требуется спроектировать и реализовать систему для проведения голосований на основе технологии блокчейн.

Реализуемая система должна выполнять следующие задачи:

- проведение голосования;
- хранение данных о проведенном голосовании;
- предоставление данных уполномоченному пользователю.

Кроме того, данная система должна обладать следующими свойствами:

- децентрализованностью;
- защитой от фальсификации данных;
- масштабируемостью
- защитой данных от несанкционированного доступа;
- высокой гибкость;
- доступностью.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ

2.1 Структура блокчейна

Блокчейн — это один из видов распределенного хранения данных, использует 3 ранее известных технологии: одноранговые сети, шифрование и базы данных. База данных представляет из себя цепочку блоков, которая специальным образом шифруется и хранится на всех узлах сети в одном и том же виде (репликация — точная копия). Весь секрет заключается в связях между блоками за счет криптографии, как следствие практически невозможно подделать информацию в блоках [3][8].

Блокчейн позволяет безопасно распространять и/или обрабатывать данные между несколькими лицами через недоверенную сеть. Данными может быть что угодно, но наиболее интересным вариантом данных является возможность передачи информации, которая требует наличия третьей доверенной стороны. Примерами такой информации являются деньги (требуют участия банка), права на собственность (требуют участия нотариуса), договор на заем и т.д. В сущности, блокчейн устраняет необходимость в участии третьего доверенного лица [13].

2.1.1 Структура блока

Каждый блок состоит из адреса, даты и времени создания, хэша и списка транзакций.

Структура блока представлена на рис. 2.1:

- Адрес – публичный ключ, генерируемый ассиметричным алгоритмом шифрования (например, RSA), на основе придуманного пользователем приватного ключа.

- Дата и время – тот момент, когда был создан блок (у транзакции тоже есть дата и время создания).
- Хэш (связующий) – вычисляется с помощью SHA512 от адреса предыдущего блока и суммы хэшей всех транзакций текущего блока, почему связующий? Потому что при его вычислении требуется адрес предыдущего блока [23].
- Информация — сообщение, сумма денег (криптовалюты), документы, история болезней, программный код (умные контракты) и т.д.

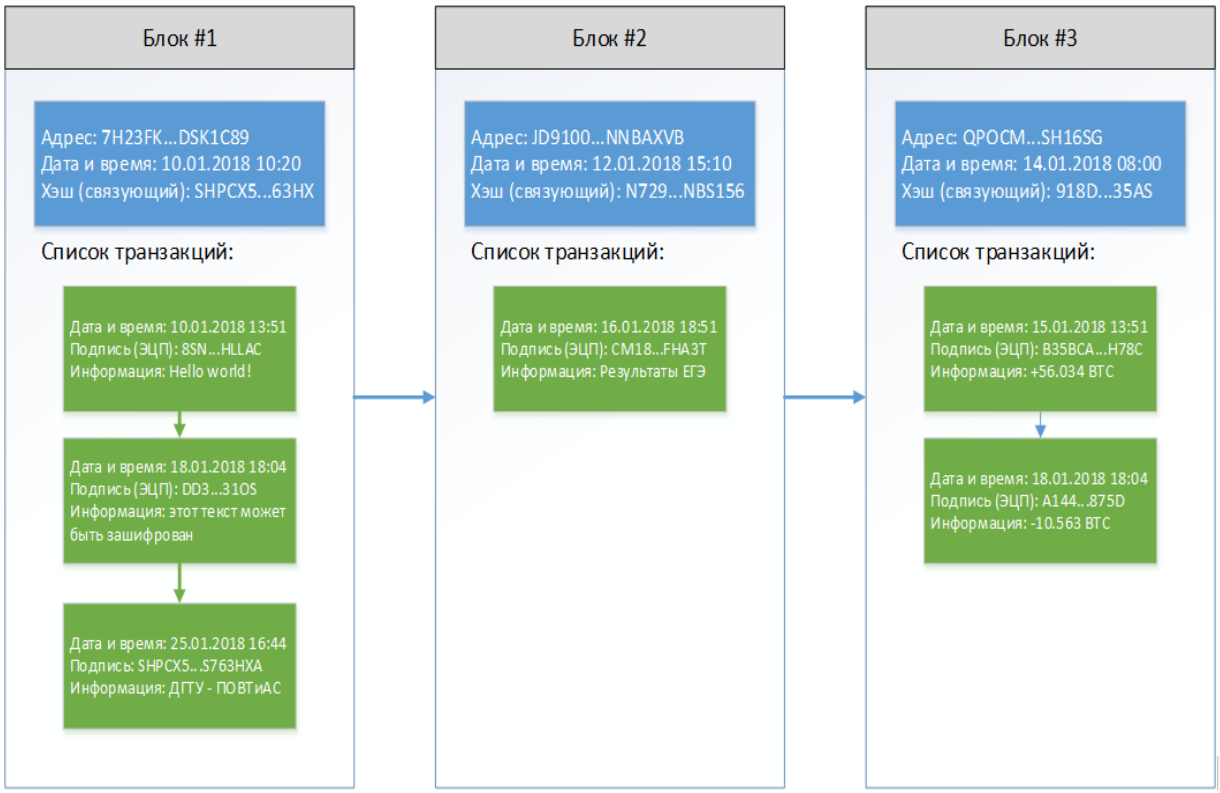

Рис. 2.1. Структура блокчейна из 3-х блоков

2.1.2 Алгоритм ECDSA

Алгоритм ECDSA (Elliptic Curve Digital Signature Algorithm) использует эллиптические кривые (elliptic curve) и конечные поля (finite field) для подписи

данных, чтобы третья сторона могла подтвердить аутентичность подписи, исключив возможность её подделки. В ECDSA для подписи и верификации используются разные процедуры, состоящие из нескольких арифметических операций [17].

Эллиптические кривые

Эллиптическая кривая над полем K — это кубическая кривая над алгебраическим замыканием поля K, задаваемая уравнением третьей степени с коэффициентами из поля K и «точкой на бесконечности». Одной из форм эллиптических кривых являются кривые Вейерштрасса [37].

$$y^2 = x^3 + ax + b \qquad (2.1)$$

Для коэффициентов a = 0 и b = 7 (используемых в биткойне), график функции представлен на рис. 2.2.

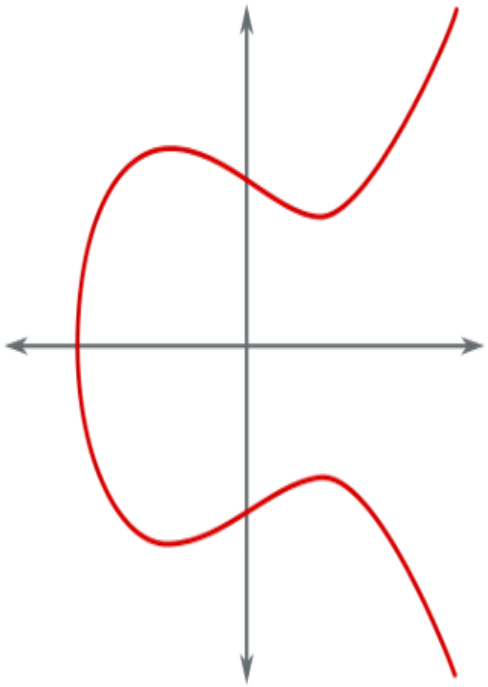

Рис. 2.2. Эллиптическая кривая

Эллиптические кривые имеют несколько интересных свойств, например, невертикальная линия, пересекающая две некасательные точки на

кривой, пересечет третью точку на кривой [24]. Суммой двух точек на кривой P + Q называется точка R, которая является отражением точки -R (построенной путем продолжения прямой (P; Q) до пересечения с кривой) относительно оси X [19]. График суммы двух точек представлен на рис. 2.3.

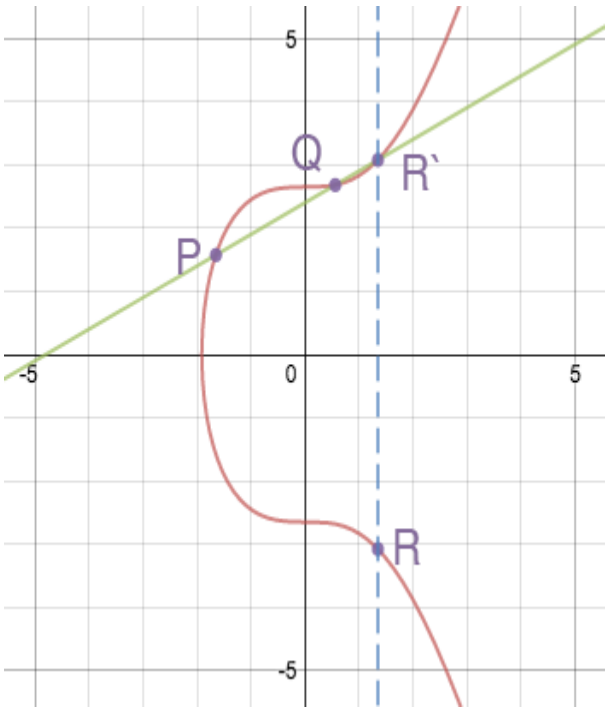

Рис. 2.3. Сумма двух точек на кривой

Если же провести прямую через две точки, имеющие координаты вида P (a, b) и Q (a, -b), то она будет параллельна оси ординат. В этом случае не будет третьей точки пересечения [32]. Чтобы решить эту проблему, вводится так называемая точка на бесконечности (point of infinity), обозначаемая как O. Поэтому, если пересечение отсутствует, уравнение принимает следующий вид

$$P + Q = O \tag{2.2}$$

Если необходимо сложить точку саму с собой (удвоить её), то в этом случае проводится касательная к точке Q [13]. Полученная точка пересечения отражается симметрично относительно оси X. График удвоения точки представлен на рис. 2.4.

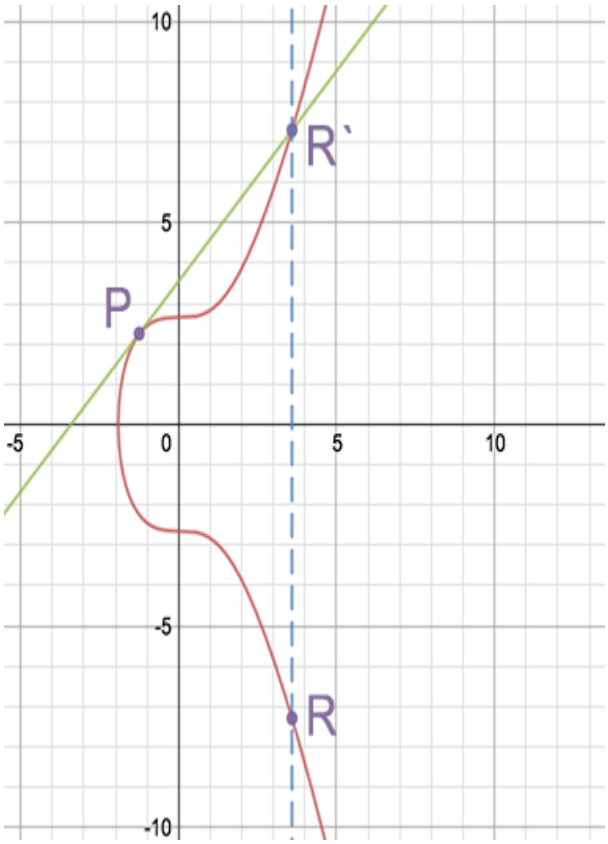

Рис. 2.4. Удвоение точки

Эти операции позволяют провести скалярное умножение точки R = k*P, складывая точку P саму с собой k раз [36]. Однако стоит отметить, что для работы с большими числами используются более быстрые методы.

Эллиптическая кривая над конечным полем

В эллиптической криптографии (ECC) используется такая же кривая, только рассматриваемая над некоторым конечным полем. Конечное поле в контексте ECC можно представить как предопределенный набор положительных чисел, в котором должен оказываться результат каждого вычисления.

$$y^2 = x^3 + ax + b \pmod p \qquad (2.3)$$

Например, 9 mod 7 = 2. В результате получится конечное поле от 0 до 6, и все операции по модулю 7, над каким бы числом они ни осуществлялись,

дадут результат, попадающий в этот диапазон.

Все названные выше свойства (сложение, умножение, точка в бесконечности) для такой функции остаются в силе, хотя график этой кривой не будет походить на эллиптическую кривую [39]. Эллиптическая кривая биткойна, $y^2 = x^3 + 7$, определенная на конечном поле по модулю 67 представлена на рис. 2.5.

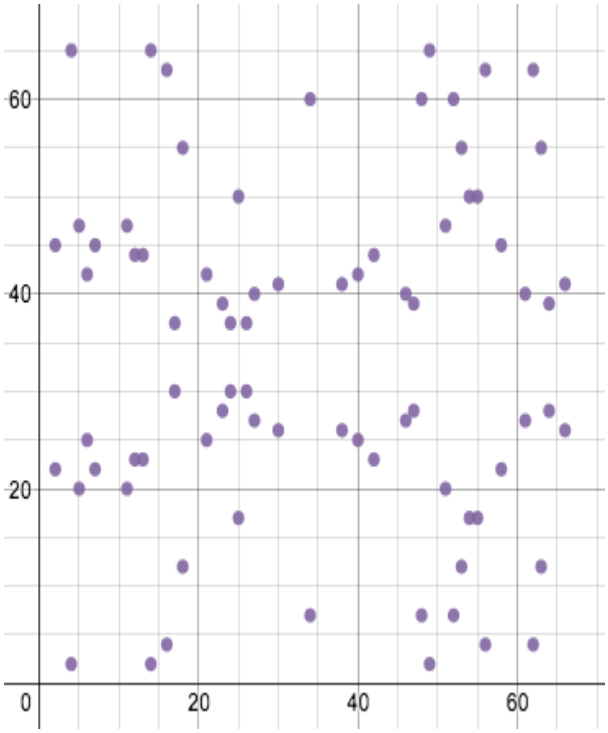

Рис. 2.5. Эллиптическая кривая биткойна

Множество точек (см. рис. 2.5), в которых все значения x и y представляют собой целые числа между 0 и 66. Прямые линии, нарисованные на графике (см. рис. 2.5), теперь будут как бы «оборачиваться» вокруг поля, как только достигнут барьера 67, и продолжатся с другого его конца, сохраняя прежний наклон, но со сдвигом [22]. Например, сложение точек (2, 22) и (6, 25) в этом конкретном случае показано на рис. 2.6.

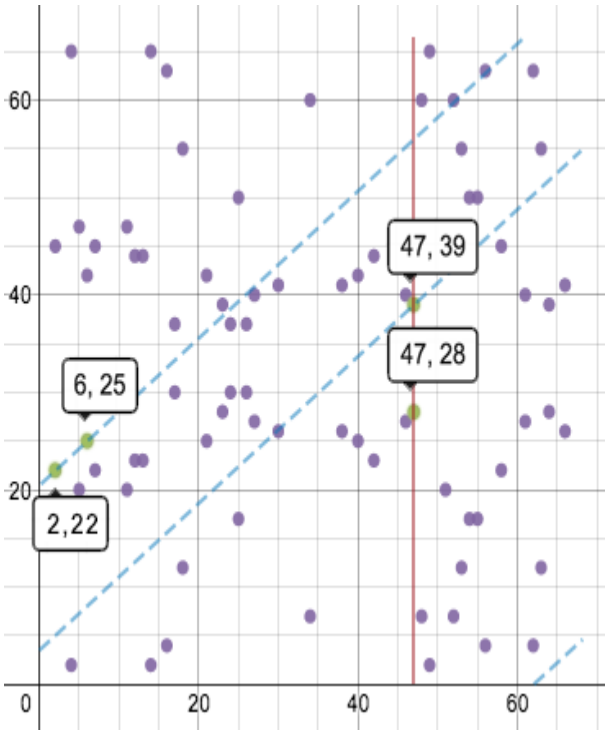

Рис. 2.6. Сложение точек (2, 22) и (6, 25)

ECDSA в биткойне

В протоколе биткойна зафиксирован набор параметров для эллиптической кривой и её конечного поля, чтобы каждый пользователь использовал строго определенный набор уравнений [17]. Среди зафиксированных параметров выделяют уравнение кривой (equation), значение модуля поля (prime modulo), базовую точку на кривой (base point) и порядок базовой точки (order) [37]. Этот параметр подбирается специально и является очень большим простым числом.

В случае биткойна используются следующие значения:

1. Уравнение эллиптической кривой:

$$y^2 = x^3 + 7 \qquad (2.4)$$

2. Простой модуль: $2^{256} — 2^{32} — 2^9 — 2^8 — 2^7 — 2^6 — 2^4 — 1$ = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F.

3. Базовая точка:

   04 **79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798** 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

   Жирным шрифтом выделена координата X в шестнадцатеричной записи. За ней сразу следует координата Y.

4. Порядок: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141.

Этот набор параметров для эллиптической кривой известен как secp256k1 и является частью семейства стандартов SEC (Standards for Efficient Cryptography), предлагаемых для использования в криптографии [21]. В биткойне кривая secp256k1 используется совместно с алгоритмом цифровой подписи ECDSA (elliptic curve digital signature algorithm). В ECDSA секретный ключ — это случайное число между единицей и значением порядка [25]. Открытый ключ формируется на основании секретного: последний умножается на значение базовой точки. Уравнение имеет следующий вид:

$$\text{Открытый ключ} = \text{секретный ключ} * G \qquad (2.4)$$

Это показывает, что максимальное количество секретных ключей (следовательно, биткойн-адресов) — конечно, и равняется порядку. Однако порядок является невероятно большим числом, так что случайно или намеренно подобрать секретный ключ другого пользователя практически не возможно.

Вычисление открытого ключа выполняется с помощью тех же операций удвоения и сложения точек. Это тривиальная задача, которую обычный персональный компьютер или смартфон решает за миллисекунды [31]. А вот обратная задача (получение секретного ключа по публичному) — является проблемой дискретного логарифмирования, которая считается вычислительно сложной. Лучшие известные алгоритмы ее решения, вроде ро Полларда,

имеют экспоненциальную сложность. Для secp256k1, чтобы решить задачу, нужно порядка 2^{128} операций, что потребует времени вычисления на обычном компьютере, сопоставимого со временем существования Вселенной [22].

Когда пара секретный/публичный ключ получена, её можно использовать для подписи данных [34]. Эти данные могут быть любой длины. Обычно первым шагом выполняется хеширование данных с целью получения уникального значения с числом битов, равным битности порядка кривой (256).

После хеширования, алгоритм подписи данных z выглядит следующим образом (G — базовая точка, n — порядок, а d — секретный ключ):

1. Выбирается некоторое целое k в пределах от 1 до n-1.
2. Рассчитывается точка (x, y) = k * G с использованием скалярного умножения.
3. Находится r = x mod n. Если r = 0, то возврат к шагу 1.
4. Находится s = (z + r * d) / k mod n. Если s = 0, то возврат к шагу 1.
5. Полученная пара (r, s) является нашей подписью.

После получения данных и подписи к ним, третья сторона, зная публичный ключ, может их верифицировать. Шаги для проверки подписи такие (Q — открытый ключ):

1. Проверка, что и r, и s находятся в диапазоне от 1 до n-1.
2. Рассчитывается $w = s^{-1} \bmod n$.
3. Рассчитывается u = z * w mod n.
4. Рассчитывается v = r * w mod n.
5. Рассчитывается точка (x, y) = uG + vQ.
6. Если r = x mod n, то подпись верна, иначе — недействительна.

Безопасность ECDSA связана со сложностью задачи поиска секретного ключа, описанной выше [33]. Помимо этого, безопасность исходной схемы зависит от «случайности» выбора k при создании подписи. Если одно и то же значение k использовать более одного раза, то из подписей можно извлечь секретный ключ. Поэтому современные реализации ECDSA, в том числе используемые в большинстве биткойн-кошельков, генерируют

k детерминировано на основе секретного ключа и подписываемого сообщения.

2.1.3 Электронная цифровая подпись

Чтобы информацию внутри транзакций нельзя было подделать, каждая транзакция внутри блока подписывается электронной цифровой подписью (ЭЦП) [17].

Электронно-цифровая подпись – это последовательность байтов, формируемая путем преобразования подписываемой информации по криптографическому алгоритму и предназначенная для проверки авторства электронного документа [7].

ЭЦП основывается на использовании асимметричного шифрования и хэш-функциях. Кратко о методах шифрования:

- симметричное шифрование использует один и тот же ключ и для зашифровывания, и для расшифровывания [28];
- асимметричное шифрование использует два разных ключа: один для зашифровывания (который также называется открытым), другой для расшифровывания (называется закрытым).

В асимметричных алгоритмах шифрования, шифрование производится с помощью открытого ключа, а расшифровка с помощью закрытого [32]. Но в асимметричных схемах цифровой подписи подписание производится с применением закрытого ключа, а проверка подписи — с применением открытого, то есть шифруем закрытым, а проверяем открытым. Одним из таких алгоритмов может быть RSA. Выбор ассиметричного шифрования обосновывается тем, что другие участники сети должны убедиться в том, что именно владелец блока внес изменения и подписал блок своей подписью [33].

Закрытый и открытый ключи

Закрытый (приватный) ключ генерируется самим пользователем, используется для подписи транзакций. Хранится в тайне, тот кто владеет приватным ключем имеет доступ к ячейке блокчейна, которая может быть представлена кошельком, контейнером с какими-либо данными (например, личная переписка, важные документы и т.д.) [34].

Открытый (публичный) ключ должен быть сгенерирован на основе приватного ключа, то есть между ними есть математическая связь. Он может быть опубликован, более того, в блокчейне его используют как адрес блока, а также в качестве проверки подлинности подписи информации в других блоках, сторонними участниками сети. Знание открытого ключа не дает возможности определить закрытый ключ [24].

Алгоритм подписания информации (документа)

Для создания подписи потребуется:

- Асимметричный алгоритм шифрования (например, RSA).
- Хэш-функция (например, SHA512).
- Информация, которую необходимо подписать.

Поскольку асимметричные алгоритмы достаточно медленные по сравнению с симметричными, то объем подписываемых данных играет большую роль и если он велик, то обычно берут хэш от подписываемых данных, а не сами данные. Хэш получают с помощью хэш-функций, например, SHA512, которая принимает на вход некую информацию и возвращает хэш определенной длины [33]. Таким образом, ЭЦП ставится не на сам документ, а на его хэш [11]. Хэш-функции не являются частью алгоритма ЭЦП, поэтому в схеме может быть использована любая надёжная хэш-функция.

Этапы:

1. С помощью RSA, генерируется пара публичный и приватный ключи.
2. Подписываемые данные подставляются в функцию SHA512 и получается хэш.

3. Полученный хэш и закрытый ключ подставляются в функцию асимметричного шифрования RSA, то есть RSAEncode(хэш от информации, закрытый ключ), на выходе получается строка – ЭЦП.

Алгоритм подписи данных продемонстрирован на рис. 2.7.

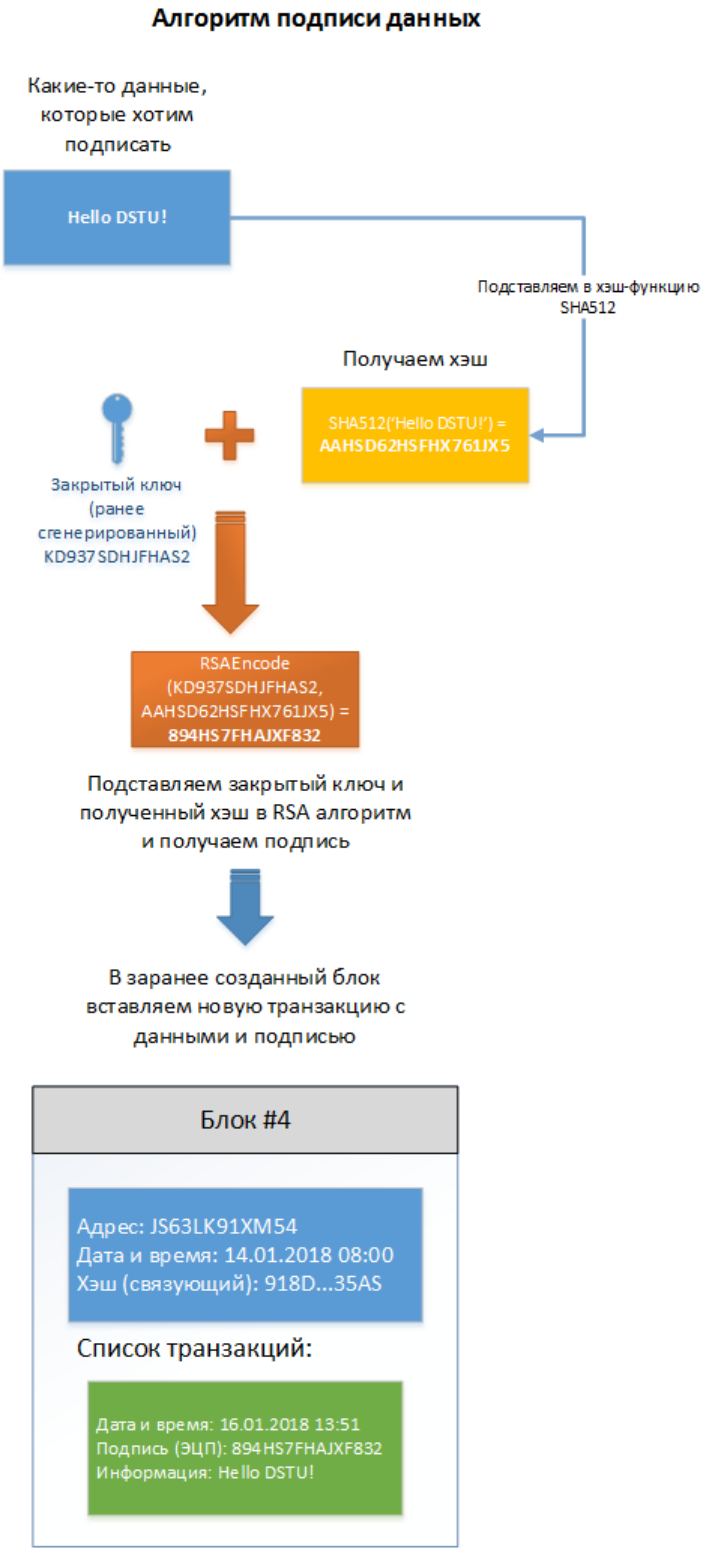

Рис. 2.7. Алгоритм подписи данных

Связующий хэш блока

Связующий хэш пересчитывается при каждом добавлении новой транзакции [12]. Он считается путем суммирования всех хэшей транзакций текущего блока и адреса предыдущего блока:

Хэш (связующий) = SHA512(block_prev_adress_hash + transaction_hash1 + transaction_hash2 + … + transaction_hashN)

Например, из рисунка (см. рис. 2.1) видно, что хэш 3-го блока вычислялся как адрес второго блока и два хэша двух внутренних транзакций:

Хэш 3 блока = SHA512(JD9100...NNBAXVB + B35BCA...H78C + A144...875D)

Именно связующий хэш объединяет блоки в единую цепь и самое главное защищает блокчейн от подделки злоумышленниками. Допустим, если кто-то захочет “выкинуть” или вставить свой блок в середину цепочки, то блоки, следующие за ним, уже не пройдут проверку, т.к. их хэш основывался на адресе, который хотят подменить или убрать [23].

Одноранговая сеть (P2P)

Одноранговая (равноправная) сеть – это сеть, основанная на равноправии участников. Часто в такой сети отсутствуют выделенные серверы, а каждый узел (peer) является как клиентом, так и выполняет функции сервера [35]. В отличие от архитектуры клиент-сервера, такая организация позволяет сохранять работоспособность сети при любом количестве и любом сочетании доступных узлов [29]. Участники сети называются пиры.

Организация сети

Каждый клиент, участвующий в работе сетевого приложения P2P, должен быть способен выполнять следующие операции:

- обнаруживать других клиентов;
- подключаться к другим клиентам;
- взаимодействовать с другими клиентами.

При подключении новый клиент связывается с трекером, который содержит списки подключенных узлов [3]. Таких трекеров может быть много,

они нужны для оптимальной связи между клиентами, под оптимальностью понимается, что скачивать цепочку блоков лучше с узла, который географически расположен рядом, чем тот, который дальше [25]. То есть связующий центр (трекер) позволяет узнать новому клиенту, кто уже есть в сети и предоставляет список наиболее “подходящих” узлов. Пример сети продемонстрирован на рис. 2.8.

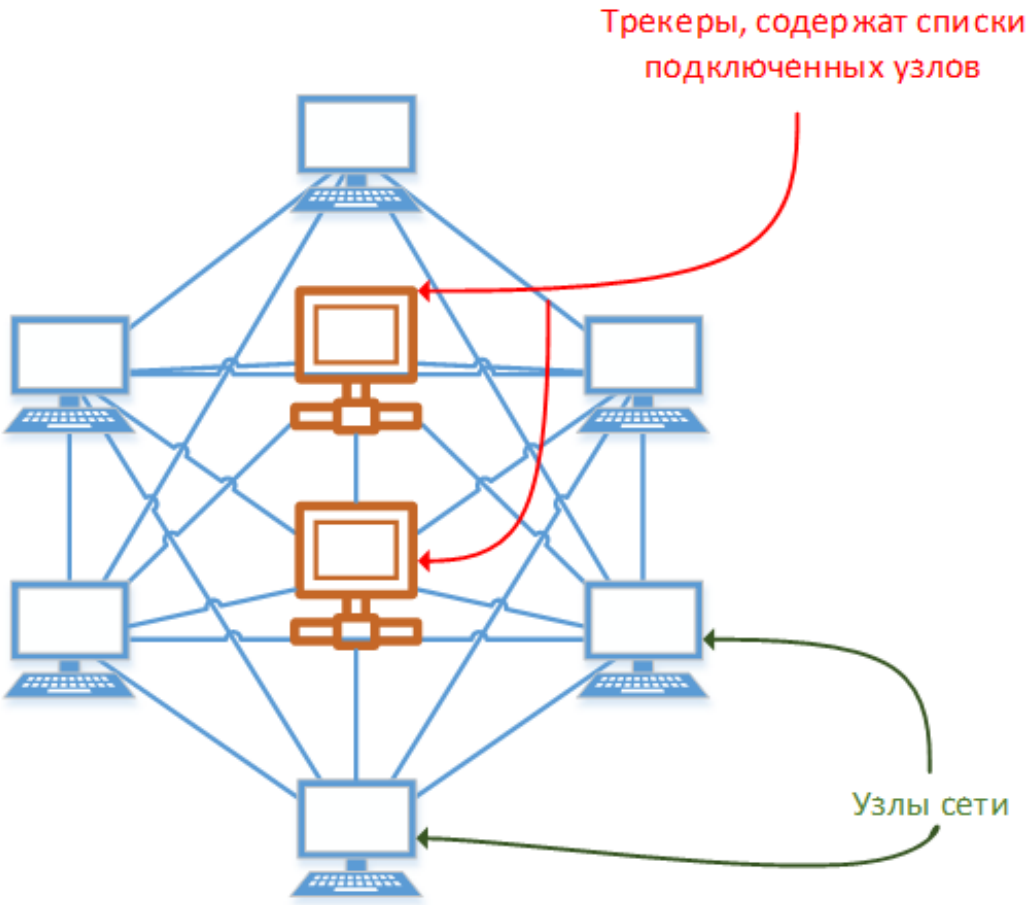

Рис. 2.8. Децентрализованная сеть с трекерами

Майнинг

В зависимости от типа блокчейна (децентрализованный вариант с использованием доказательства работы или централизованный с доверенным центром) блоки транзакций (пустые) могут создаваться майнерами или главным узлом (центром) [31]. Майнеры – узлы сети, которые вычисляют новый блок (имеется ввиду блок транзакции). Дело в том, что в некоторых типах блокчейна создать новый блок не так просто, необходимо решить

сложную задачу путем перебора чисел (если быть точным получить хэш с 10 нулями в начале ~ 0000000000HD83HA653JA…83JS), это сделано с целью безопасности, чтобы другие участники не смогли быстро подменить цепочку блоков, т.к. на расчет такого хэша могут уйти часы, дни, недели (подменив один, придется пересчитать другие блоки) [35].

В других типах блокчейна эти хэши уже могут быть вычислены заранее и соответственно отпадает необходимость в майнерах, как добытчиках блоков, здесь цель майнера уже не добывать блок, а предоставлять свой жесткий диск для хранения цепочки [8].

Проверка данных блокчейна

После добавления нового блока или вставки новых данных в список транзакций уже созданного блока, необходимо, чтобы другие участники сети проверили данную информацию.

Алгоритм проверки транзакции

Алгоритм проверки транзакции показан на рис. 2.9. Проверку транзакций другими участниками сети можно разделить на этапы:

1. Получение данных и подписи из новой транзакции.
2. Получение хэша SHA512 от данных.
3. Подпись и открытый ключ взятый из адреса этого же блока расшифрованный с помощью RSADecode(подпись, публичный ключ).
4. Сравнение полученного на этапе 2 хэша, с хэшем полученным на этапе 3 из расшифрованной подписи [17]. Если совпали, значит данные корректны и подписаны ключем владельца, транзакция добавляется в блок. Если не совпали, данные фальшивы и транзакция отклоняется и не добавляется в блок.

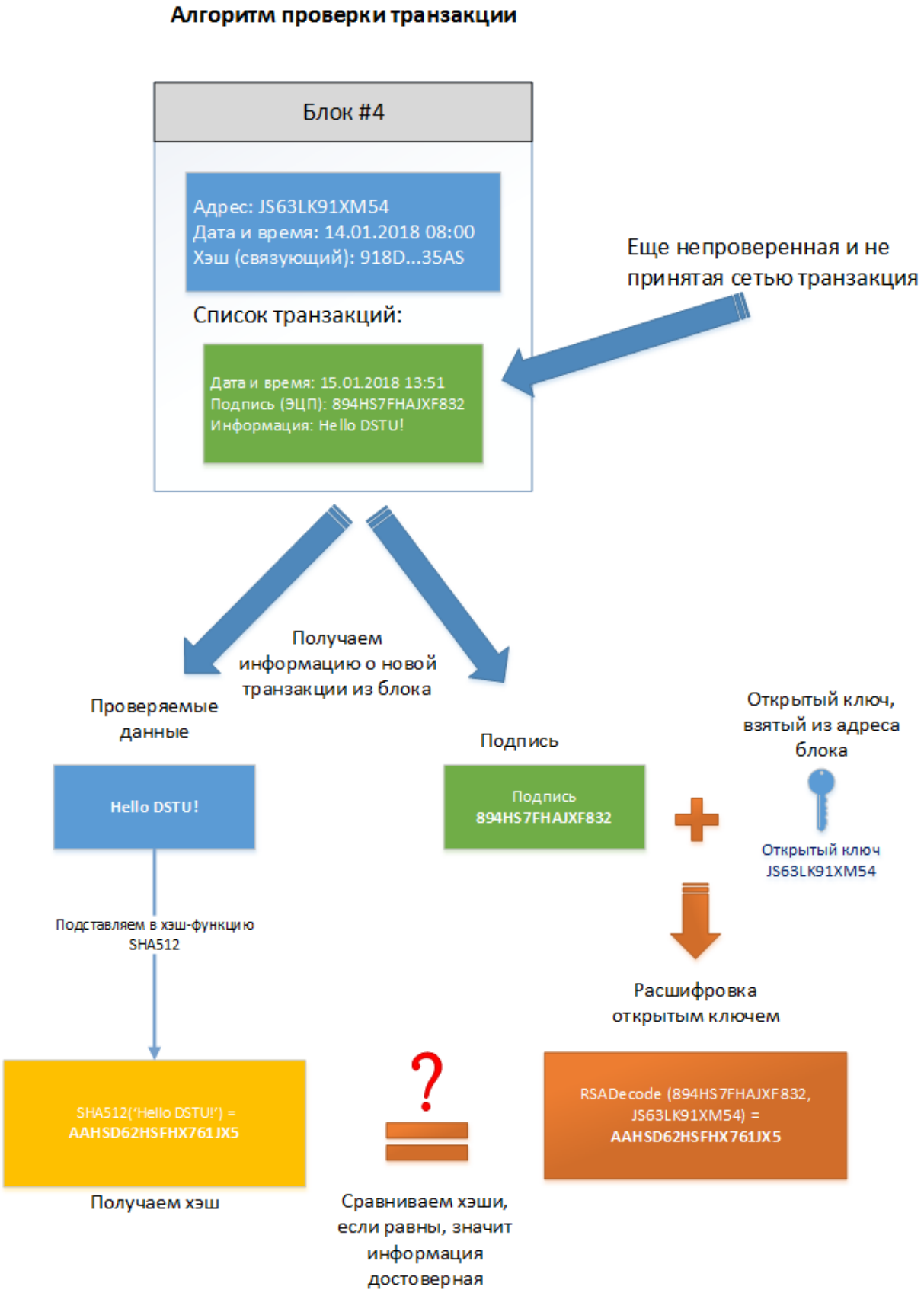

Рис. 2.9. Алгоритм проверки транзакции

Алгоритм проверки блока

Проверка нового блока делится на этапы:

1. Берется адрес последнего принятого блока (текущий блок еще не принят и не является последним) и список транзакций текущего блока. Затем вычисляется хэш [11].

   Хэш = SHA512(block_prev_adress_hash + transaction_hash1 + transaction_hash2 + … + transaction_hashN)

2. Полученный хэш сравнивается с хэшем (связующим) еще непринятого блока. Если совпали, тогда блок корректный и добавляется в цепочку. Иначе, данные некорректные и блок не принимается [21].

2.2 Разработка «токена»

В мире Ethereum существует стандарт токена или монеты. Если контракт отвечает стандарту, то его можно назвать монетой (валютой или токеном) [29]. Стандарт монеты на базе эфира называется — ERC20 [20].

Чтобы смарт-контракт считался валютой он должен содержать:

1. Методы
   - function totalSupply() constant returns (uint256 totalSupply);
   - function balanceOf(address _owner) constant returns (uint256 balance);
   - function transfer(address _to, uint256 _value) returns (bool success);
   - function transferFrom(address _from, address _to, uint256 _value) returns (bool success);
   - function approve(address _spender, uint256 _value) returns (bool success);
   - function allowance(address _owner, address _spender) constant returns (uint256 remaining).
2. События

- event Transfer(address indexed _from, address indexed _to, uint256 _value);
- event Approval(address indexed _owner, address indexed _spender, uint256 _value).

Контракт который выполняет обязательные требования можно назвать частично-совместимыми с ERC20.

Когда ERC20 еще не был стандартом существовали негласные правила («дополнительная информация») — наличие трех полей в внутри контракта:

- string public constant name = "Token Name";
- string public constant symbol = "SYM";
- uint8 public constant decimals = 18.

На текущий момент эти правила также входят в стандарт ERC223.

В идеале монета должна выполнять все соглашения.

Детальное описание требований:

1. Метод function totalSupply() constant returns (uint256 totalSupply). Возвращает суммарное количество выпущенных монет. Эту функцию может вызвать любой.
2. Метод function balanceOf(address _owner) constant returns (uint256 balance). Возвращает количество монет, принадлежащих _owner. Может вызвать любой. Кошельки для отображения монеты вызывают именно эту функцию.
3. Метод function transfer(address _to, uint256 _value) returns (bool success). Передает _value монет на адрес _to. Когда пользователь будет перемещать свои монеты на другой адрес вызываться будет именно эта функция. Соответственно монеты должны браться с баланса пользователя, который вызвал эту функцию. Метод должен создавать событие Transfer (описан будет далее) в случае успешного перемещения монет [32].
4. Метод function transferFrom(address _from, address _to, uint256 _value) returns (bool success). Передает _ value монет от _from к _to.

Пользователь должен иметь разрешение на перемещение монеток между адресами, дабы любой желающий не смог управлять чужими кошельками. Фактически эта функция позволяет доверенному лицу распоряжаться определенным объемом монеток на счету. Дать разрешение на управление средствами можно следующей функцией. Метод должен создавать событие Transfer (описан будет далее) в случае успешного перемещения монет [15].

5. Метод function approve(address _spender, uint256 _value) returns (bool success). Разрешает пользователю _spender снимать со счета (точнее со счета вызвавшего функцию пользователя) средства не более чем _value. На основе этого разрешения должна работать функция transferFrom. Метод должен создавать событие Approval (описан будет далее) [24].
6. Метод function allowance(address _owner, address _spender) constant returns (uint256 remaining). Возвращает сколько монет со своего счета разрешил снимать пользователь _owner пользователю _spender.
7. Событие event Transfer(address indexed _from, address indexed _to, uint256 _value). Событие, которое должно возникать при любом перемещении монет. Т.е. его нужно создавать внутри функций transfer и transferFrom в случае успешного перемещения монет [20].
8. Событие event Approval(address indexed _owner, address indexed _spender, uint256 _value). Событие должно возникать при получении разрешения на снятие монет. Фактически должно создаваться внутри функции approve.
9. Поле string public constant name = "Token Name". Хранит полное название монеты.
10. Поле string public constant symbol = "SYM". Хранит короткое название монеты. Иначе говоря — символ. С этим символом валюта будет отображаться на биржах и в кошельках.
11. Поле uint8 public constant decimals = 18. Количество знаков после запятой. 18 — это наиболее распространенное значение.

На рис. 2.10. представлена схема наследования стандартов для «токенов».

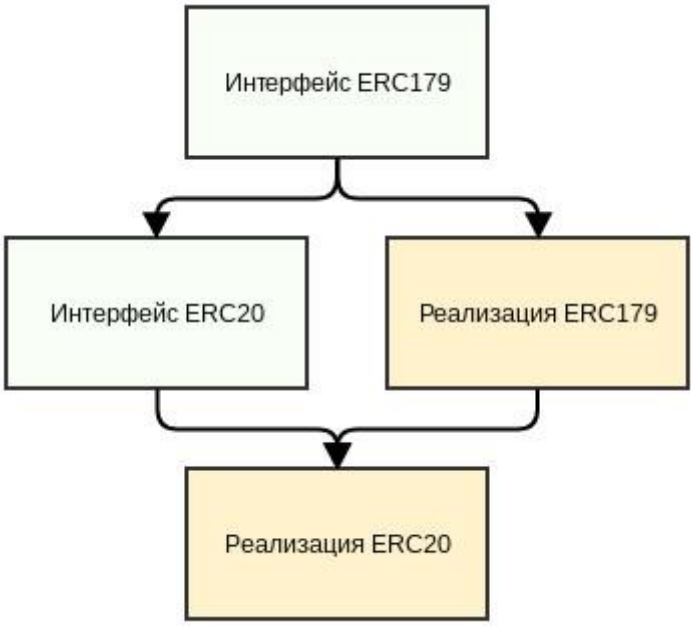

Рис. 2.10. Схема наследования «токенов»

Далее рассмотрен код модификатора onlyOwner. Код данного модификатора представлен на листинге 2.1.

Листинг 2.1. Код модификатора

```
    modifier onlyOwner() public {
        require(msg.sender == owner);
        _;
    }
```

Конец листинга

Он состоит всего из двух строк. В первой проверяется является ли исполнитель владельцем. Если не является, то происходит выход из функции. А если является, то будет исполнена вторая строчка, в которой вместо специального символа «_» будет исполнен код функции, которая сейчас выполняется с модификатором.

Теперь, если потребуется написать функцию, которая может быть исполнена только владельцем контракта, то достаточно добавить onlyOwner к ее заголовку.

Помимо задач ERC20 в токене есть еще функция отвечающая за эмиссию новых монет [34]. Это тоже широко-используемый шаблон. Контракты, которые могут выпускать новые монеты, обычно, называют Mintable. А поскольку выпускать новые монеты может только владелец контракта, то Mintable наследуется от Ownable.

С учетом всего вышесказанного структура контрактов токена теперь выглядит как на рис. 2.11.

Важно понимать, что от библиотек не наследуются. Их «используют», т.е. подключют к контракту. Например, для SafeMath это выглядит так:

using SafeMath for uint256;

Возьмем готовый код стандартных шаблонов из репозитория ZeppelinSolidity и перепишем токен:

1. ERC20Basic — интерфейс ERC179 , т.е. ERC20 без механизма предоставления разрешений.
2. ERC20 — интерфейс ERC20.
3. BasicToken — абстрактная реализация ERC179.
4. StandartToken — абстрактная реализация ERC20.
5. Ownable — предоставляет возможность ограничивать доступ к функциям всем кроме владельца контракта.
6. SafeMath — библиотека безопасных математических операций. В случае проблем с выполнением операции работа функции прерывается.
7. MintableToken — предоставляет возможность выпуска новых монет. Если выпуск новых монет больше не нужен, то вызывается finishMinting (в ICO вызывается как правило после окончания распродажи монет). Возобновить выпуск монет больше будет нельзя.

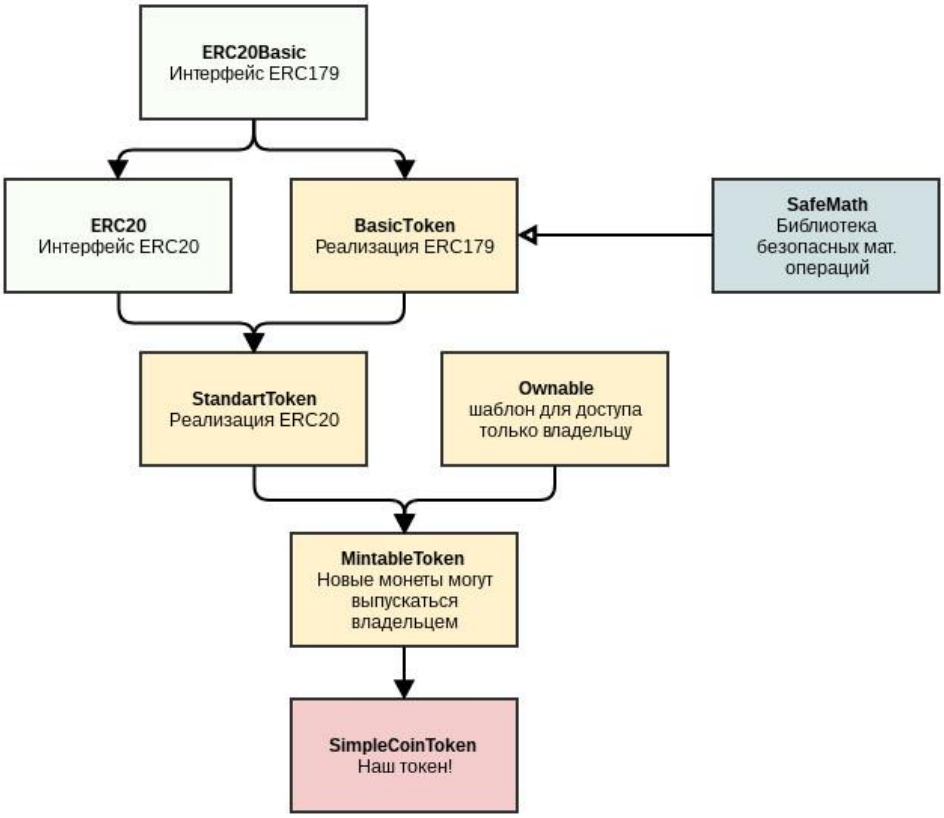

Рис. 2.11. Обновленная схеме наследования «токена»

На листинге 2.2 представлен шаблон токена для голосования.

Листинг 2.2. Шаблон токена для голосования

```
pragma solidity ^0.4.13;
contract VoteTokenCoin is MintableToken {
    string public constant name = "Vote Coin Token";
    string public constant symbol = "VCT";
    uint32 public constant decimals = 18;
}
```

Конец листинга

2.3 Язык Solidity

Язык был предложен в августе 2014 года Гэйвином Вудом (Gavin Wood). В дальнейшем разработка языка была выполнена под руководством Кристиана

Райтвизнера (Christian Reitwiessner) командой Solidity в рамках проекта Ethereum. Это один из четырех языков (среди Serpent, LLL и Mutan), спроектированных для трансляции в байт код виртуальной машины Ethereum [35].

Получил широкое распространение с появлением технологий блокчейна, в частности стека технологий на основе Ethereum, для создания программного обеспечения умных контрактов.

Solidity статически типизированный JavaScript-подобный язык программирования, созданный для разработки самовыполняющихся контрактов, исполняющихся на виртуальной машине Ethereum(EVM) [36]. Программы на языке Solidity транслируются в байткод EVM. Solidity позволяет разработчикам создавать самодостаточные приложения, содержащие бизнес-логику, результирующую в неотменяемые транзакционные записи блокчейна [23].

Не очевидные особенности solidity:

1. Смарт-контракт после деплоя имеет свой адрес. Любой юзер может что-то запросить у контракта по этому адресу или переслать средства на этот адрес [27].
2. Заданные переменные в "корне" вашего смарт-контракта будут храниться всегда там. При вызове функции вы можете к этим данным обратиться - прочитать или изменить.
3. Внутри смарт-контракта монет обычно хранится структура в виде адрес кошелька в сети = количество монет. Это и есть все данные, необходимые монете. В ERC20 уже описана эта и некоторые другие функциональности, например, отправка токенов от одного адреса другому [12].
4. Для хранения таблиц данных существуют структуры.

2.4 Разработка умного контракта

Смарт-контракт является программой написанной на языке программирования. Для платформы Ethereum используется язык Solidity [35].

Существуют важные нюансы, когда дело доходит до разработки умного контракта:

- Код является открытым исходным кодом, но есть также «открытая среда выполнения». Это означает, что все общение между клиентами и контрактом видимо для участников сети, которые участвуют в выполнении этих вызовов [36]. Такой основополагающий принцип требует разного подхода к проектированию, когда все конфиденциальные данные или меры аутентификации должны быть надлежащим образом обработаны с использованием шифрования и встроенных методов, таких как учетные записи Ethereum и закрытые ключи, например, некоторые операции могут быть ограничены владельцем контракта [32].
- Хранение очень дорого. Это сильно контрастирует с тем, к чему привыкли разработчики. В типичном современном подходе лозунг «хранение дешево» вездесущ. Мы часто сохраняем и запрашиваем сотни гигабайт данных без излишней суеты. Хранение на блокчейне имеет значительную цену на газ, поэтому для разработки контракта требуется много низкоуровневых соображений о том, что хранится и какие типы следует использовать [37].

В ходе разработки будет использован framework truffle, который поможет с управлением миграциями, компиляцией, управлением зависимостями, а также тестированием.

Помимо этого, будет использован фреймворк OpenZeppelin, который содержит в себе набор контрактов и библиотек, написанных на языке Solidity и уже доказавших свою полезность и безопасность временем [38].

Для начала необходимо установить framework truffle. Для этого нужно

скачать установочный файл для операционной системы и устанавить его, после этого создать папку для проекта. Далее на терминале перейти в созданную папку и инициализировать проект при помощи команды truffle init.

Открыв проект в среде для разработки (Intellij IDEA) можно увидеть, что после инициализации было создано 3 папки:

- contracts, предназначенная для хранения исходного кода умных контрактов;
- migrations, содержащая файлы миграций, которые служат для публикации контрактов в блокчейн;
- test, содержащая тесты контрактов. Важно отметить, что тесты можно писать как на Solidity, так и на JavaScript.

На рис. 2.12. представлено «дерево» проекта.

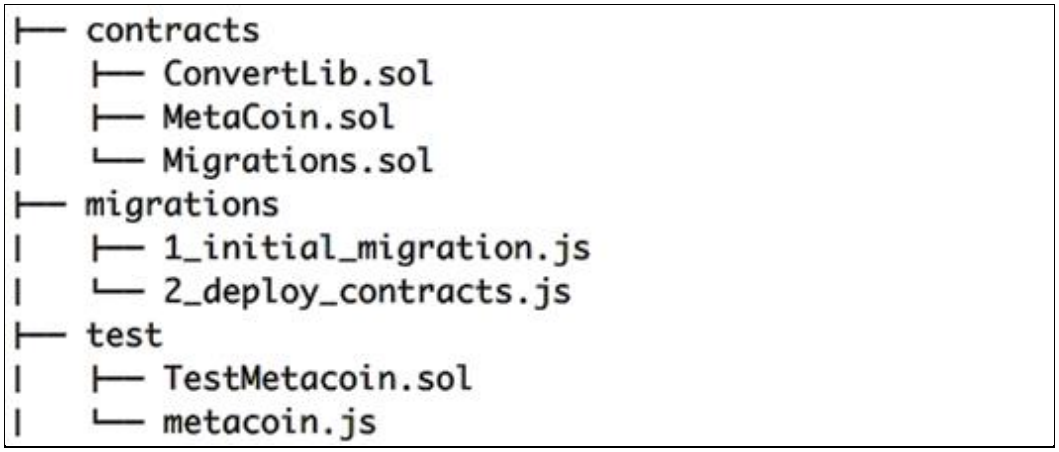

Рис. 2.12. «дерево» проекта

Как можно заметить, truffle уже создал пару простых умных контрактов. Они будут использованы для тестирования установки.

2.5 Архитектура приложения

Приложение состоит из двух основных частей: контракта, работающего в сети Ethereum и клиента. Архитектура приложения представлена на

рисунке 2.13.

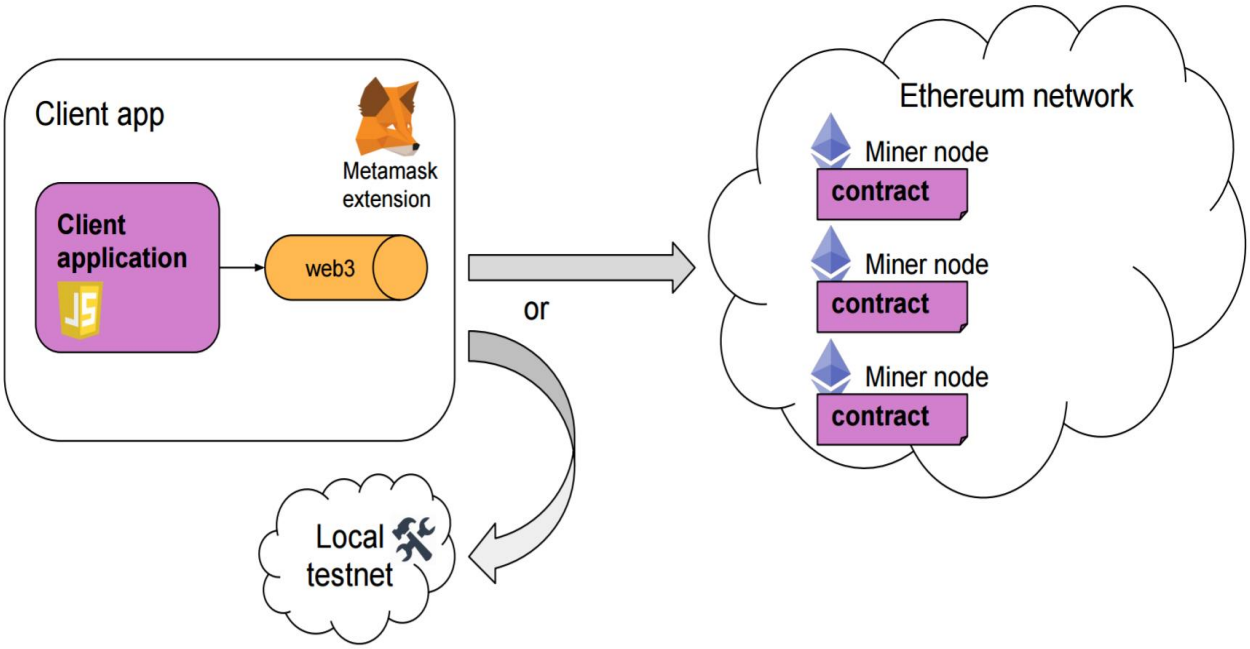

Рис. 2.13. Архитектура приложения

2.6 Клиентское приложение

В данном примере клиент - это веб-приложение, работающее в браузере. Для вызова сетевого API Ethereum нужна web3.js библиотека и подключение к сетевому узлу Ethereum. Узлы можно запускать самостоятельно или подключаться к существующим через мост / прокси, например, используя Metamask [39].

Metamask это плагин для браузера, который служит мостом для подключения к сети и ввода web3 экземпляра в код. Metamask также позволяет выбирать различные сети (например, локальные для тестирования) и учетные записи Ethereum [38].

Клиентское приложение - это всего лишь набор статических ресурсов, которые могут выполняться локально или быть выставлены на веб-сервере, таком как Apache. На рис. 2.14. представлена архитектура системы в связке с клиентским приложением.

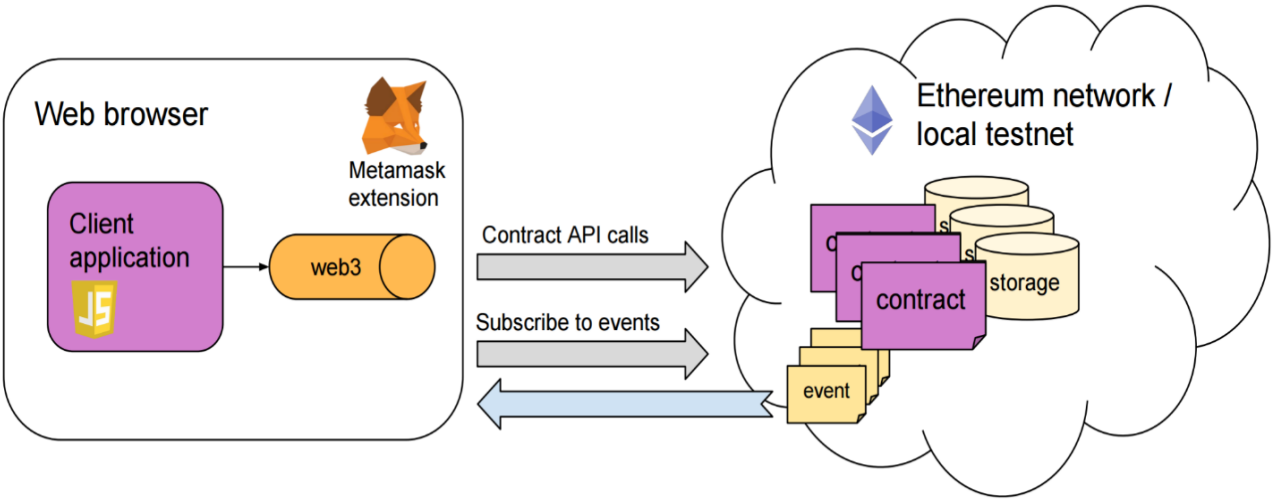

Рис. 2.14. Архитектура системы с клиентским приложением

2.7 Алгоритм работы системы

Алгоритм работы системы разбит на шаги, которые представлены на рис. 2.15.

1. Создатель голосования инициирует новое голосование. Они создают новую учетную запись Ethereum, которая создает частную пару с открытым ключом. Открытый ключ - это только адрес учетной записи. Эта запись теперь называется учетной записью.
2. Создатель голосования называет контракт для хранения нового голосования по блочной цепочке, связанного с адресом подписывающего лица.
3. Создатель голосования распределяет «токен» голосования избирателю.
4. Избиратель готовит свой токен. Ключ подписывающего лица используется для цифровой подписи хэша их адреса (избирателя).
5. Избиратель посылает свой голос на контракт, вместе с «токеном» и подписью этого «токена» (сделанную с использованием закрытого ключа подписывающего лица).

6. Контракт проверяет возможность голосования избирателем (наличие «токена голосования» на счету избирателя.
7. В случае успешной проверки, «токен голосования» зачисляется на счет «кандидата».

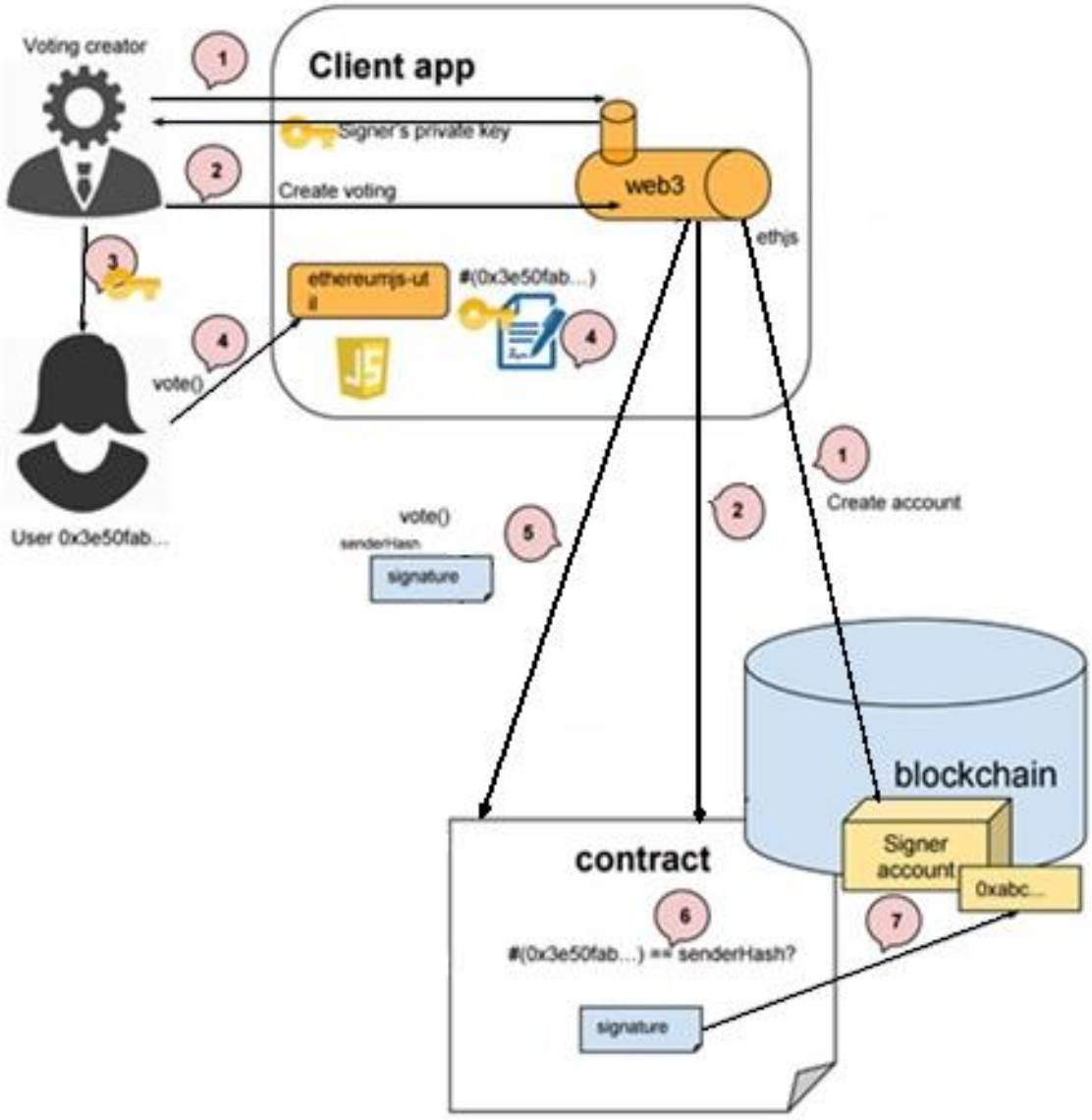

Рис. 2.15. Алгоритм работы системы

Эти проверки гарантируют, что:

- Только пользователи, которые знают секретный ключ подписчика, могут отправлять голоса.
- Никто не может олицетворять аутентифицированного пользователя и менять свой голос.

ГЛАВА 3. НАСТРОЙКА И ТЕСТИРОВАНИЕ СИСТЕМЫ

3.1 Создание приватной сети

Для того чтобы развернуть приватную сеть требуется установить необходимое ПО.

Для установки софта необходимо выполнить следующие команды:

- sudo apt-get install software-properties-common;
- sudo add-apt-repository -y ppa:ethereum/ethereum;
- sudo apt-get update;
- sudo apt-get install ethereum.

В пакете ethereum также содержится консольный интерфейс Geth, необходимый для работы с сетью.

Создание Генезиса

После установки всех необходимых пакетов, необходимо перейти к формированию Генезиса. Генезис – Самый первый блок в цепочке блокчейн, всегда заполняется вручную, даже в реальной сети.

Создадим файл genesis.json выполнив следующую команду: «`sudo nano genesis.json`».

В полученный файл необходимо записать параметры, пример параметров представлен в листинге 3.1.

ParentHash - хэш ссылка на предыдущий блок блокчейна. В данном случае первым блоком является генезис, поэтому он равен нулю

mixhash, nonce – параметры, используемые в технологии the Proof-of-Work (PoW) [40]. Они подтверждают проведение требуемого количества вычислений для блока.

Листинг 3.1. Параметры genesis.json

```
{
    "nonce": "0x0000000000000042",
    "timestamp": "0x0",
    "parentHash":
    "0x0000000000000000000000000000000000000000000000000000000000000000",
    "extraData": "0x0",
    "gasLimit": "0x8000000",
    "difficulty": "0x400",
    "mixhash":
    "0x0000000000000000000000000000000000000000000000000000000000000000",
    "coinbase": "0x3333333333333333333333333333333333333333",
    "alloc": { }
}
```

Конец листинга.

Coinbase – бенефициарий. Требуется для корректной работы системы расчета вознаграждения при генерации новых блоков. (Для реального проекта параметры необходимо изменить, т.к. приведенные настройки подвергают сеть взлому.)

Запуск node и инициализация genesis

Перед запуском нескольких узлов эфира необходимо убедиться, что:

- каждый экземпляр имеет отдельный каталог данных (--datadir);
- каждый экземпляр работает на разных портах(both eth and rpc)(--portand —rpcport);
- в случае кластера экземпляры должны знать друг о друге ipc endpointдолжны быть уникальными или ipc интерфейс должен быть выключен (--ipcpath or —ipcdisable).

Для инициализации блока genesis первой ноды (node1) необходимо выполнить команду geth со следующими параметрами:

- --identity "Billing1";
- --rpc;
- --rpcport "8082";
- --rpccorsdomain "*";
- --datadir "/home/project/network/node2";
- -- ipcdisable --port "30302";
- --rpcapi "db,eth,net,web3";
- --networkid 1999;
- init /home/project/network/genesis.json.

В результате выполнения команды был получен результат, представленный на рис. 3.1.

```
I1212 14:31:04.487121 cmd/geth/chaincmd.go:131] successfully wrote genesis block and/or chain rule set: 6650a0ac6c5e805475e7ca48eae5df0e32a2147a154bb2222731c770ddb5c158
```

Рис. 3.1. Результат инициализации первой ноды

Для инициализации блока genesis для второй ноды (node2) необходимо на другом терминале выполнить команду geth со следующими параметрами:

- --identity "Billing2";
- --rpc;
- --rpcport "8083";
- --rpccorsdomain "*";
- --datadir "/home/project/network/node3";
- -- ipcdisable;
- --port "30303";
- --rpcapi "db,eth,net,web3";
- --networkid 1999;
- init /home/project/network/genesis.json.

В результате выполнения команды был получен результат представленный на рис. 3.2.

```
I1212 14:31:04.487121 cmd/geth/chaincmd.go:131] successfully wrote genesis block and/or chain rule set: 6650a0ac6c5e805475e7ca48eae5df0e32a2147a154bb2222731c770ddb5c158
```

Рис. 3.2. Результат инициализации первой ноды

Запуск nodes и клиента

Для запуска первой ноды необходимо вызвать команду geth со следующими параметрами:

- --identity "Billing1";
- --rpc;
- --rpcport "8082";
- --rpccorsdomain "*";
- --datadir "/home/project/network/node2";
- --ipcdisable;
- --port "30302";
- --rpcapi "db,eth,net,web3";
- --networkid 1999;
- console 2>> /home/project/network/node2/filelog1.log.

Для создания аккаунта с паролем “password” необходимо выполнить команду: «`personal.newAccount(“password“)`»

Адрес аккаунта отображается в консоле следующим образом: «`0x04d4eddae29f46b0578e23cc9b01ccdb1a8ba86c`»

Для присвоения аккаунту статуса майнера необходимо выполнить команду: «`miner.setEtherbase(eth.accounts[0])`»

Для запуска генерации блоков необходимо выполнить команду: «`miner.start()`»

После формирования DAG в консоли будут отображены записи, сообщающие о корректности формирования блоков. Записи о формировании блоков представлены на рис. 3.3.

```
I1212 14:31:04.822979 eth/backend.go:484] checking DAG (ethash
dir: /home/.ethash)
I1212                                          14:31:05.955535
vendor/github.com/ethereum/ethash/ethash.go:276]          Done
generating DAG for epoch 0, it took    1.135322105s    I1212
214:31:07.589805 miner/unconfirmed.go:83]
mined potential block #1 [050f878e…], waiting for 5 blocks to
confirm    I1212 14:31:08.590261 miner/worker.go:514] commit
new work on block 2 with 0 txs & 0 uncles. Took 248.713µs
I1212 14:31:08.596575 miner/unconfirmed.go:83]
mined potential block #1 [e88f1c28…], waiting for 5 blocks to
confirm
I1212 14:31:08.959788 miner/unconfirmed.go:83] mined potential
block #2 [8ace02fc…], waiting for 5 blocks to   confirm
```

Рис. 3.3. Результат формирования блоков

Для того чтобы остановить майнинг необходимо выполнить команду: «miner.stop()»

Далее необходимо получить данные о первой ноде, для этого необходимо выполнить команду: «admin.nodeInfo.enode»

Данные о первой ноде представлены на рис. 3.4.

```
enode://2e21fe4357c72368a61737d90ff7f645df121f99601942c879acb
fb3fb53a6b90f66ebe7bc87086c0b9584056b357
dc1f7424b93c5348af7429c839cc9bf151b@[::]:30302
```

Рис. 3.4. Информация о первой ноде

Для запуска второй ноды необходимо вызвать команду geth со следующими параметрами:

- --identity "Billing2";

- --rpc;
- --rpcport "8083";
- --rpccorsdomain "*";
- --datadir "/home/project/network/node3";
- -- ipcdisable;
- --port "30303";
- --rpcapi "db,eth,net,web3";
- --networkid 1999;
- console 2>> /home/project/network/node3/filelog2.log.

После этого необходимо связать узлы.

Для этого в каждый узел необходимо добавить уникальный адрес другого узла, выполнив в консоли следующую команду: «admin.nodeInfo.enode»

Далее необходимо передать полученный идентификатор в функцию: «admin.addPeer("идентификатор узла")»

Для вывода в консоль данных о второй ноде необходимо выполнить команду: «admin.nodeInfo.enode»

Данные о второй ноде представлены на рис. 3.5.

```
enode://23409a6196304d1f49d0e9ea832b2fa005c4ee424c44dae5c4fd9
633187fa15411201eb6737dfc2de001ff1b4aeb6
53d06e558ffae9b87254eb047cc77aa7025@[::]:30303
```

Рис. 3.5. Информация о второй ноде

Для синхронизации между нодами необходимо выполнить команду которая представлена в листинге 3.2.

Листинг 3.2. Команда для синхронизации между нодами

```
admin.addPeer("enode://23409a6196304d1f49d0e9ea832b2fa005c4ee424
c44dae5c4fd9633187fa15411201eb6737dfc2
de001ff1b4aeb653d06e558ffae9b87254eb047cc77aa7025@[127.0.0.1]:30
303")
```

Конец листинга.

В консоли второй ноды можно увидеть, что блоки, сгенерированные в первой ноде импортировались во вторую ноду.

Таким образом в приватной сети 2 клиента будут синхронизировать блоки друг друга по мере их появления.

3.2 Настройка Metamask

Создание учетной записи

Для того, чтобы начать пользоваться Metamask, необходимо создать учетную запись. Для этого необходимо ввести пароль, подтвердить его, и нажать кнопку Create. На рис. 3.6. представлен интерфейс создания кошелька [37].

В следующем окне можно увидеть так называемые SEED WORDS. Их необходимо сохранить в надежном месте. Только с помощью них будет возможно восстановить аккаунты Metamask. На рис. 3.7. представлен интерфейс заполнения seed words.

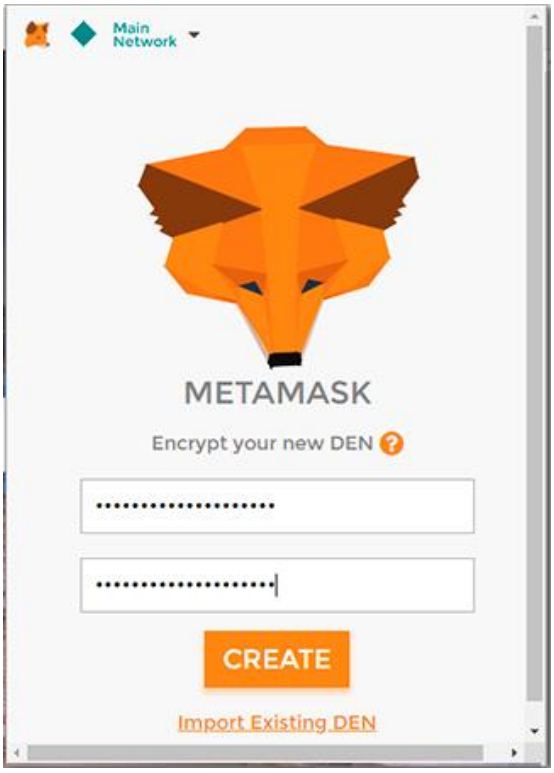

Рис. 3.6. Окно создания кошелька

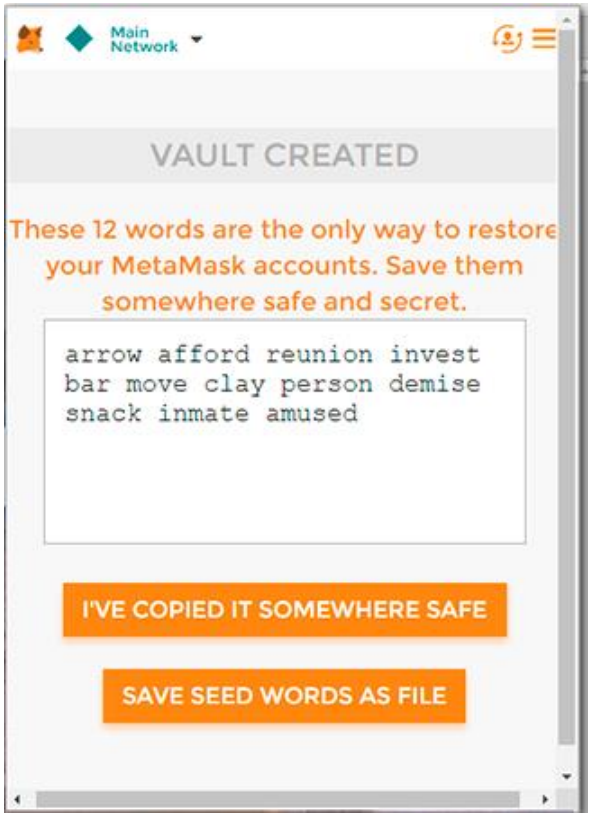

Рис. 3.7. Seed words метамаска

На данном этапе учетная запись создана.

Создание кошелька в Metamask

Если кошельки ранее не были сгенерированы, можно воспользоваться кошельком, созданным по умолчанию в момент создания учетной записи. На рис. 3.8. представлен кошелек, созданный по умолчанию.

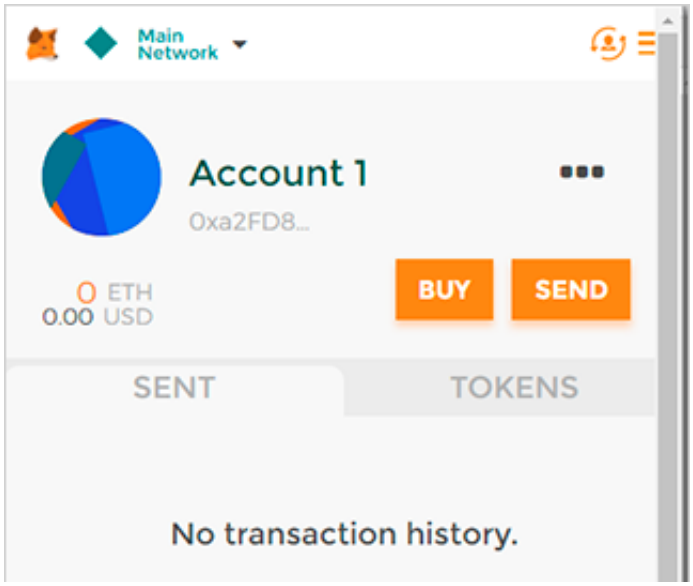

Рис. 3.8. Кошелек сгенерированный по умолчанию

Выбрать другой аккаунт, создать новый или импортировать существующий можно в меню Manage Accounts. Интерфейс выбора активного аккаунта представлен на рис. 3.9.

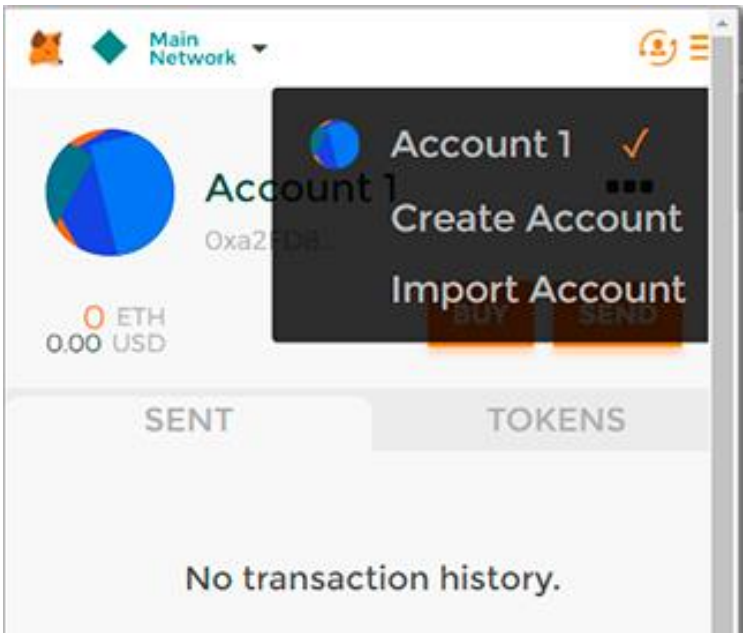

Рис. 3.9. Интерфейс выбора аккаунта

Импортирование аккаунта

Для того чтобы добавить существующий кошелек в Метамаск, в меню manage accounts необходимо выбрать пункт Import Account.

Импорт аккаунта доступен двумя способами:

- Ввод Private Key.
- Загрузка JSON File.

На рис. 3.10. представлен интерфейс импортирования существующего кошелька.

Private Key – секретный ключ, который создается одновременно с кошельком. На рис. 3.11 показано как выглядит секретный ключ.

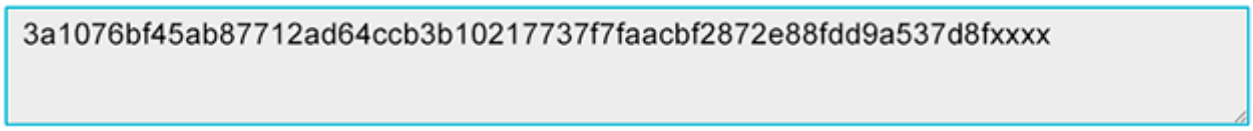

Рис. 3.11. Секретный ключ

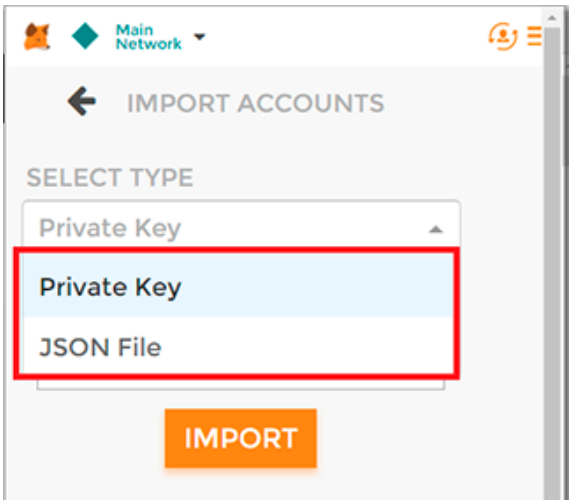

Рис. 3.10. Интерфейс импорта кошелька

JSON File используется для импорта кошелька созданного ранее с помощью других систем.

Операции по пополнению кошелька Metamask

Для пополнения кошелька эфиром или любыми другими ERC20 токенами – достаточно предоставить адрес кошелька отправителю. Его можно скопировать в буфер, нажав на Copy Address to clipboard в меню аккаунта. На рис. 3.12 представлен интерфейс программы для пополнения кошелька эфиром или ERC20 токенами.

После нажатия кнопки адрес будет скопирован в буфер обмена операционной системы. Адрес кошелька представлен в виде: «0xa2FD83811f82d3E7e3DdE9Ab330F926296768aB8».

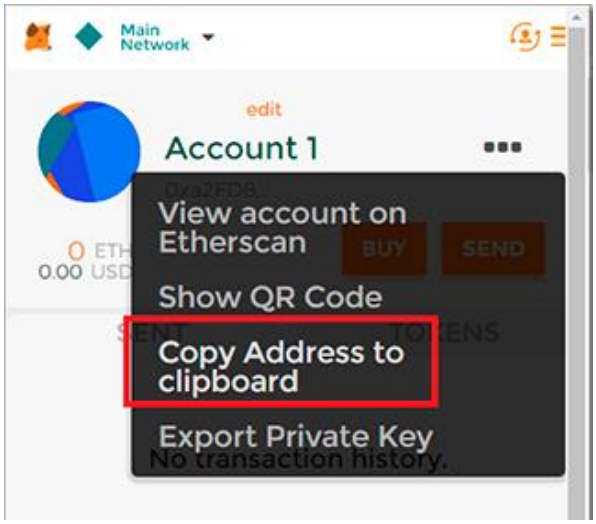

Рис. 3.12. Интерфейс пополнения кошелька

Отследить поступления и состояние кошелька можно в Etherscan. Переход по ссылке осуществится при нажатии на View account in Etherscan в меню аккаунта. На рис. 3.13 представлен интерфейс сервиса Etherscan позволяющего отслеживать все операции, перемещения эфира и других ERC20 токенов.

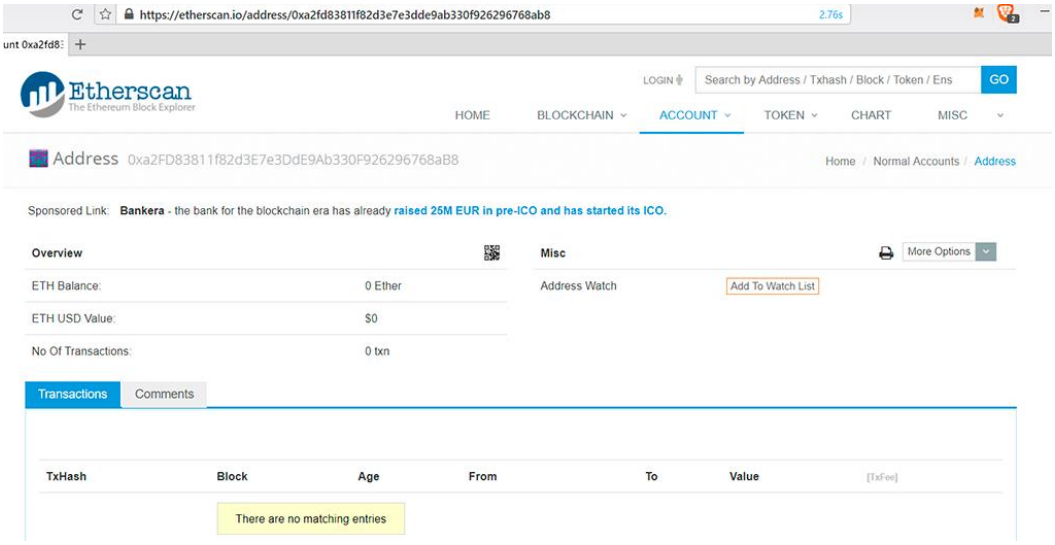

Рис. 3.13. Интерфейс сервиса Etherscan

3.3 Деплой смарт контракта

Для тестирования и деплоя смарт контрактов необходимо воспользоваться ранее установленным фреймворком Truffle, он скрывает часть низкоуровневой работы за абстракциями. Для инициализации проекта необходимо выполнить команду: «`$ truffle init`»

Создается только базовая структура из папок contracts, migrations и tests. В contracts и migrationsвы можно увидеть смарт контракт Migrations, который отвечает за логику деплоя (в терминах фреймворка — миграции).

Смарт контракт Migrations используется для того, чтобы сохранять какие из скриптов миграций уже выполнились. Если в процессе разработки добавлять новые контракты и новую логику деплоя, то предыдущий успешный прогресс передеплоивать не нужно.

Также можно вручную создавать файлы контрактов, миграций и тестов, однако в truffle есть и специальные команды:

- $ truffle create contract ExampleContract;
- $ truffle create migration ExampleMigration;
- $ truffle create test ExampleContract.

Для начала процесса деплоймента необходимо выполнить команду:

```
«$ truffle develop»
```

Результат выполнения команды представлен на рис. 3.14.

Данная команда поднимает тестовое окружение и дает доступ к нему через консоль. Далее необходимо выполнить команду для выполнения цикла команд для компиляции, миграции и тестирования:

```
«truffle(develop)> compile»
```

Результат выполнения команды представлен на рис. 3.15.

```
truffle(develop)> compile
Compiling .\contracts\Migrations.sol...
Compiling .\contracts\VotingContract.sol...
```

Рис. 3.15. Результат компиляции

```
«Truffle Develop started at http://localhost:9545/»
 (0) 0x627306090abab3a6e1400e9345bc60c78a8bef57
(1) 0xf17f52151ebef6c7334fad080c5704d77216b732
(2) 0xc5fdf4076b8f3a5357c5e395ab970b5b54098fef
(3) 0x821aea9a577a9b44299b9c15c88cf3087f3b5544
(4) 0x0d1d4e623d10f9fba5db95830f7d3839406c6af2
(5) 0x2932b7a2355d6fecc4b5c0b6bd44cc31df247a2e
(6) 0x2191ef87e392377ec08e7c08eb105ef5448eced5
(7) 0x0f4f2ac550a1b4e2280d04c21cea7ebd822934b5
(8) 0x6330a553fc93768f612722bb8c2ec78ac90b3bbc
(9) 0x5aeda56215b167893e80b4fe645ba6d5bab767de
Mnemonic: candy maple cake sugar pudding cream honey rich
smooth crumble sweet treat
truffle(develop)>
```

Рис. 3.14. Результат инициации деплоя

Для выполнения миграции необходимо выполнить команду:

«truffle(develop)> migrate»

Результат выполнения команды представлен на рис. 3.16:

```
truffle(develop)> migrate
Using network 'develop'.
Running migration: 1_initial_migration.js
  Replacing Migrations...
0xce8aea41f29a301ede9c03c609cb9ba1e4e7d375cbac759f977f4bf9e08
24541
  Migrations: 0x8cdaf0cd259887258bc13a92c0a6da92698644c0
Saving successful migration to network...
0xd7bc86d31bee32fa3988f1c1eabce403a1b5d570340a3a9cdba53a472ee
8c956
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying VotingContract...
0x90763d485f7060d7efa581c69590520673c18c861d76273079f858365f2
0c7c4
  VotingContract: 0x345ca3e014aaf5dca488057592ee47305d9b3e10
Saving successful migration to network...
0xf36163615f41ef7ed8f4a8f192149a0bf633fe1a2398ce001bf44c43dc7
bdda0
Saving artifacts...
truffle(develop)>
```

Рис. 3.16. Результат деплоя

На данном этапе деплоймент контракта в тестовую сеть закончен.

ЗАКЛЮЧЕНИЕ

Блокчейн—новая парадигма информационного мира. Она постучалась в двери в 2008 году, когда некто по имени Сатоши Накамото описал протокол электронных платежей для пиринговой сети. Так была заложена основа для технологии блокчейн. Это математический алгоритм, который позволяет безопасно и приватно обмениваться ценностями через пиринговые сети. Первой практической реализацией блокчейн стала сеть Биткоин.

Технологии блокчейн представляют собой кардинально новый подход к организации деловых операций. Они знаменуют новое поколение надежных и умных приложений для регистрации и обмена физическими, виртуальными, материальными и нематериальными активами. Благодаря ключевым понятиям криптографической безопасности, децентрализованному консенсусу и общему открытому реестру (должным образом контролируемому и ограниченному в видимости), блокчейн-технологии могут коренным образом изменить организацию нашей экономической, социальной, политической и научной деятельности.

В ходе выполнения выпускной квалификационной работы была спроектирована и разработана система автоматизированного голосования, использующая в качестве основы технологию блокчейн. В результате чего были решены следующие задачи:

- Произведен обзор актуальных технологий, используемых в разработке систем, основанных на блокчейн технологии.
- Произведена разработка приватного «токена» для подсчета голосов.
- Спроектирован и разработан умный контракт обеспечивающий работу системы.
- Настроена компиляция и процесс деплоймента системы в приватную сеть.
- Развернута защищенная приватная сеть.

В первой главе были рассмотрены основные концепции блокчейн технологии, а также платформ, построенных на её основе. Также была рассмотрена платформа Ethereum которая позволяет создавать умные контракты обеспечивающие надежные соглашения между двумя не доверенными лицами. Кроме того, в данной главе описывается концепция голосования с применением данной технологии, а также примеры применения блокчейн для голосования в других странах мира.

Во второй главе подробно описана структура блокчейн протокола, также в данной главе приводится описание разработки собственного «токена» для проведения и учета голосований. Кроме того, в данной главе приводится процесс написания умного контракта.

В третьей главе описаны настройка и тестирование системы, провайдера Metamask, а также пошаговое руководство по развертыванию приватной сети. Которую в последующем можно использовать для деплоймента «токена» и умного контракта. Данную сеть можно использовать для проведения голосования.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Creating a Private Chain/Testnet [Электронный ресурс], режим доступа - https://souptacular.gitbooks.io/ethereum-tutorials-and-tips-by-hudson/content/private-chain.html, свободный (дата обращения: 29.03.2018);
2. Ethereum [Электронный ресурс], режим доступа - https://ru.wikipedia.org/wiki/Ethereum, свободный (дата обращения: 29.05.2018);
3. OpenZeppelin [Электронный ресурс], режим доступа - https://github.com/OpenZeppelin/openzeppelin-solidity, свободный (дата обращения: 14.03.2018);
4. Solidity [Электронный ресурс], режим доступа - https://solidity.readthedocs.io/en/v0.4.24/, свободный (дата обращения: 17.02.2018);
5. Truffle framework [Электронный ресурс], режим доступа - http://truffleframework.com/docs/getting_started/project, свободный (дата обращения: 25.03.2018);
6. Antonopoulos, A. Mastering Bitcoin: Programming the Open Blockchain [Текст] / A. Antonopoulos. - O'Reilly Media, 2017. - 416 с.
7. Antonopoulos, A. Mastering Ethereum: Building Smart Contracts and Dapps [Текст]/ A. Antonopoulos, G. Wood. - O'Reilly Media, 2018. - 200 с.
8. Bheemaiah, K. The Blockchain Alternative [Текст] / K. Bheemaiah. - Apress, 2017. - 272 с.
9. Blockchain: A Practical Guide to Developing Business, Law, and Technology Solutions [Текст] / J. Bambara, P. Allen, K. Iyer, R. Madsen, S. Lederer, M. Wuehler. – McGraw-Hill, 2018. - 320 с.
10. Buterin, V. The Business Blockchain. Promise, Practice, and Application of the Next Internet Technology [Текст]/ V. Buterin, W. Mougayar. - John Wiley & Sons Limited, 2017. - 211 с.

11. Dannen, C. Introducing Ethereum and Solidity [Текст] / C. Dannen. - Apress, 2017. - 318 с.
12. Diedrich, H. Ethereum: Blockchains, Digital Assets, Smart Contracts, Decentralized Autonomous Organizations [Текст] / H. Diedrich. - CreateSpace Independent Publishing Platform, 2016. - 360 с.
13. Dingle, S. In Math We Trust: Bitcoin, Cryptocurrency and the Journey To Being Your Own Bank [Текст] / S. Dingle. - Tracey McDonald Publishers, 2018. - 172 с.
14. Drescher, D. Blockchain Basics [Текст] / D. Drescher. - Apress, 2017. - 276 с.
15. Gates, M. Blockchain: Ultimate guide to understanding blockchain, bitcoin, cryptocurrencies, smart contracts and the future of money [Текст] / M. Gates. - CreateSpace Independent Publishing Platform, 2017. - 125 с.
16. Laurence, T. Blockchain For Dummies [Текст] / T. Laurence. - John Wiley & Sons Limited, 2017. - 243 с.
17. Mahalik, A. Deconstructing Distributed Blockchain [Текст] / A. Mahalik. - Apress, 2017. - 183 с.
18. Mougayar, W. The Business Blockchain [Текст]/ W. Mougayar, V. Buterin. - John Wiley & Sons Limited, 2017. - 200 с.
19. Mukhopadhyay, M. Ethereum Smart Contract Development [Текст] / M. Mukhopadhyay. - Apress, 2017. - 249 с.
20. Norman, A. Blockchain Technology Explained: The Ultimate Beginner’s Guide About Blockchain Wallet, Mining, Bitcoin, Ethereum, Litecoin, Zcash, Monero, Ripple, Dash, IOTA And Smart Contracts [Текст] / A. Norman. - CreateSpace Independent Publishing Platform, 2017. - 126 с.
21. Peterson, T. Cryptocurrency: The Complete Beginner's Guide - Blockchain and Cryptocurrency, Technologies, Mining, Investing and Trading [Текст] / T. Peterson. - ADS, 2018. - 178 с.
22. Ramirez, R. Ethereum Bible: All You Need to Know About Ethereum [Текст] / R. Ramirez. - ADS, 2018. - 43 с.

23. Swan, M. Blockchain: Blueprint for a New Economy [Текст] / M. Swan. - O'Reilly Media, 2015. - 152 с.
24. Szucs, I. Blockchain Trends 2018 [Текст] / I. Szucs. - Издательские решения, 2017. - 11 с.
25. Takashima, I. Ethereum: The Ultimate Guide to the World of Ethereum, Ethereum Mining, Ethereum Investing, Smart Contracts, Dapps and DAOs, Ether, Blockchain Technology[Текст] / I. Takashima. - ADS, 2017. - 98 с.
26. Tapscott, D. Blockchain Revolution: How the Technology Behind Bitcoin and Other Cryptocurrencies Is Changing the World [Текст]/ D. Tapscott, A. Tapscott. - Portfolio, 2018. - 432 с.
27. Geth - интерфейс командной строки go-ethereum [Электронный ресурс], режим доступа - https://golos.io/geth/@idiatulla/geth-interfeis-komandnoi-stroki-go-ethereum, свободный (дата обращения: 10.04.2018);
28. Ethereum для начинающих [Электронный ресурс], режим доступа - https://ru.insider.pro/tutorials/2017-06-30/ethereum-dlya-nachinayushih-polnoe-rukovodstvo/, свободный (дата обращения: 26.05.2018);
29. Proof-of-Work vs. Proof-of-Stake: Как изменится Ethereum [Электронный ресурс], режим доступа - https://ru.insider.pro/tutorials/2017-07-14/proof-work-vs-proof-stake-kak-izmenitsya-ethereum/, свободный (дата обращения: 05.03.2018);
30. Блокчейн: возможности, структура, ЭЦП [Электронный ресурс], режим доступа - https://habr.com/post/348014/, свободный (дата обращения: 03.05.2018);
31. Блокчейн: организация сети, проверка подписи [Электронный ресурс], режим доступа - https://habr.com/post/348020/, свободный (дата обращения: 12.04.2018);
32. Введение в разработку умных контрактов Ethereum [Электронный ресурс], режим доступа - https://habr.com/post/335710/, свободный (дата обращения: 02.05.2018);

33. Выборы на блокчейне: возможно ли голосование без вбросов и фальсификаций? [Электронный ресурс], режим доступа - https://blockchain.ru/posts/vybory-na-blokcheine-vozmozhno-li-golosovanie-bez-vbrosov-i-falsifikatsii, свободный (дата обращения: 11.04.2018);
34. Как с помощью блокчейна защитить свои данные [Электронный ресурс], режим доступа - https://habr.com/company/acronis/blog/331326/, свободный (дата обращения: 18.04.2018);
35. Пишем смарт-контракт Ethereum [Электронный ресурс], режим доступа - http://inaword.ru/blokchejn/pishem-smart-kontrakt-ethereum-eto-prosto-chast-6-token-erc20-refaktoring/, свободный (дата обращения: 19.05.2018);
36. Руководство по Solidity [Электронный ресурс], режим доступа - https://github.com/ethereum/wiki/wiki/Руководство-по-Solidity, свободный (дата обращения: 27.04.2018);
37. Смарт-контракты [Электронный ресурс], режим доступа - www.tadviser.ru/index.php/Статья:Смарт-контракты_(Smart_contract), свободный (дата обращения: 18.05.2018);
38. Создаём собственный блокчейн на Ethereum [Электронный ресурс], режим доступа - https://habr.com/post/341466/, свободный (дата обращения: 16.03.2018);
39. Транзакции в сети Ethereum — GAS (газ) комиссии за переводы токенов [Электронный ресурс], режим доступа - https://mining-cryptocurrency.ru/ethereum-tranzakcii-komissii-gas/#i-3, свободный (дата обращения: 27.02.2018);
40. Raval, S. Децентрализованные приложения. Технология Blockchain в действии [Текст] / S. Raval. - Питер: O'Reilly, 2017. - 192 с.

ПРИЛОЖЕНИЕ

VotingContract.sol

```
pragma solidity ^0.4.13;

contract ERC20Basic {
    uint256 public totalSupply;
    function balanceOf(address who) constant returns (uint256);
    function transfer(address to, uint256 value) returns (bool);
    event Transfer(address indexed from, address indexed to,
uint256 value);
}

contract ERC20 is ERC20Basic {
    function allowance(address owner, address spender) constant
returns (uint256);
    function transferFrom(address from, address to, uint256
value) returns (bool);
    function approve(address spender, uint256 value) returns
(bool);
    event Approval(address indexed owner, address indexed
spender, uint256 value);
}

library SafeMath {

    function mul(uint256 a, uint256 b) internal constant returns
(uint256) {
        uint256 c = a * b;
        assert(a == 0 || c / a == b);
        return c;
    }

    function div(uint256 a, uint256 b) internal constant returns
(uint256) {
```

```
        // assert(b > 0); // Solidity automatically throws when
dividing by 0
        uint256 c = a / b;
        // assert(a == b * c + a % b); // There is no case in
which this doesn't hold
        return c;
    }

    function sub(uint256 a, uint256 b) internal constant returns
(uint256) {
        assert(b <= a);
        return a - b;
    }

    function add(uint256 a, uint256 b) internal constant returns
(uint256) {
        uint256 c = a + b;
        assert(c >= a);
        return c;
    }

}

contract BasicToken is ERC20Basic {

    using SafeMath for uint256;

    mapping(address => uint256) balances;

    function transfer(address _to, uint256 _value) returns
(bool) {
        balances[msg.sender] = balances[msg.sender].sub(_value);
        balances[_to] = balances[_to].add(_value);
        Transfer(msg.sender, _to, _value);
        return true;
```

```
    }

    function balanceOf(address _owner) constant returns (uint256
balance) {
        return balances[_owner];
    }

}

contract StandardToken is ERC20, BasicToken {

    mapping (address => mapping (address => uint256)) allowed;

    function transferFrom(address _from, address _to, uint256
_value) returns (bool) {
        var _allowance = allowed[_from][msg.sender];

        // Check is not needed because sub(_allowance, _value)
will already throw if this condition is not met
        // require (_value <= _allowance);

        balances[_to] = balances[_to].add(_value);
        balances[_from] = balances[_from].sub(_value);
        allowed[_from][msg.sender] = _allowance.sub(_value);
        Transfer(_from, _to, _value);
        return true;
    }

    function approve(address _spender, uint256 _value) returns
(bool) {

        require((_value == 0) || (allowed[msg.sender][_spender]
== 0));

        allowed[msg.sender][_spender] = _value;
```

```
        Approval(msg.sender, _spender, _value);
        return true;
    }

    function allowance(address _owner, address _spender)
constant returns (uint256 remaining) {
        return allowed[_owner][_spender];
    }

}

contract Ownable {

    address public owner;

    function Ownable() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner);
        _;
    }

    function transferOwnership(address newOwner) onlyOwner {
        require(newOwner != address(0));
        owner = newOwner;
    }

}

contract MintableToken is StandardToken, Ownable {

    event Mint(address indexed to, uint256 amount);
```

```
    event MintFinished();

    bool public mintingFinished = false;

    modifier canMint() {
        require(!mintingFinished);
        _;
    }

    function mint(address _to, uint256 _amount) onlyOwner
canMint returns (bool) {
        totalSupply = totalSupply.add(_amount);
        balances[_to] = balances[_to].add(_amount);
        Mint(_to, _amount);
        return true;
    }

    function finishMinting() onlyOwner returns (bool) {
        mintingFinished = true;
        MintFinished();
        return true;
    }

}

contract VotingTokenCoin is MintableToken {

    string public constant name = "Voting Coint Token";

    string public constant symbol = "VCT";

    uint32 public constant decimals = 18;

}
```

```
contract VotingContract is Ownable {

    using SafeMath for uint;

    VotingTokenCoin public token = new VotingTokenCoin();

    address[] candidates;

    address[] voters;

    uint start;

    uint period;

    function VotingContract() {
        start = 1500379200;
        period = 28;
    }

    modifier voteIsOn() {
        require(now > start && now < start + period * 1 days);
        _;
    }

    function finishMinting() public onlyOwner {
        token.finishMinting();
    }

    function addVoters (address[] newVoters) public onlyOwner {
        for(uint i = 0; i < newVoters.length; i++){
            token.balances.push(newVoters[i]);
            voters.push(newVoters[i]);
        }
    }
```

```
    function addCandidates (address[] newCandidates) public
onlyOwner {
        for(uint i = 0; i < newCandidates.length; i++){
            token.balances.push(newCandidates[i]);
            candidates.push(newCandidates[i]);
        }
    }

    function initVote () public onlyOwner {
        for(uint i = 0; i < voters.length; i++){
            token.mint(voters[i], 1);
        }
    }

    function vote(address voter, address candidate) returns
(bool){
        token.transferFrom(voter, candidates, 1);
        return true;
    }

    function getResults () public onlyOwner returns
(mapping(address => uint256)) {
        mapping (address => uint256) results;
        for(uint i = 0; i < candidates.length; i++){
            results[candidates[i]] =
token.balanceOf(candidates[i]);
        }
        return results;
    }

    function() external payable {
        createTokens();
    }
```

```
}
```

truffle.js

```
module.exports = {
    networks: {
        development: {
            host: "localhost",
            port: 8545,
            network_id: "*",
            gas: 5000000
        }
    },
    rpc: {
        host: "localhost",
        port: 8545
    },
    mocha: {
        useColors: true
    }
};
```

Migrations.sol

```
pragma solidity ^0.4.18;


contract Migrations {
    address public owner;

    uint public last_completed_migration;

    modifier restricted() {
```

```
        if (msg.sender == owner) _;
    }

    function Migrations() public {
        owner = msg.sender;
    }

    function setCompleted(uint completed) public restricted {
        last_completed_migration = completed;
    }

    function upgrade(address new_address) public restricted {
        Migrations upgraded = Migrations(new_address);
        upgraded.setCompleted(last_completed_migration);
    }
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) (Ф.И.О.)