LTO.network

Blockchain for Decentralized Workflows

www.lto.network

Аннотация

Цифровое преобразование и автоматизация бизнес-процессов дают большие преимущества относительно их производительности и сокращения затрат. Однако организации сопротивляются внедрению данных процессов в межорганизационное взаимодействие, отчасти из-за отсутствия доверия. Биткойн доказал, что блокчейн использует распределенные вычисления и криптографию для поддержания работы системы, которая не полагается на доверие.

LTO строится на этом же фундаменте с децентрализованным движком автоматизации бизнес-процессов, базируясь на одноранговой сети. Информация делится между сторонами, использующими приватные блокчейны (новая цепочка на каждый процесс) и хешируется в публичном блокчейне. Этот гибридный подход позволяет организациям соблюдать любые правила защиты данных и решает проблемы масштабируемости, обычно присущие блокчейн-проектам.

Вступление Цифровая революция привела к многочисленным изменениям, которые направлены на то, чтобы сделать нашу жизнь более эффективной[1]. Эта волна прогресса, в первую очередь, прошлась по потребительским и внутренним бизнес-процессам. Когда дело доходит до межорганизационных процессов, мы должны признать, что эти изменения менее радикальны. На замену письмам и факсам пришли е-мейлы, а пишущая машинка была заменена текстовыми процессороми (в графической обертке это уже текстовые редакторы). Но помимо этих поверхностных изменений, сами методы исполнения основных процессов практически не изменились.

Основной причиной отсутствия автоматизации является нежелание корпораций полагаться на внешние системы, которыми управляет контрагент[2], в то время как распределение информации играет важную роль в формировании отношений[3]. Там, где есть дисбаланс в возможностях, одна сторона может захватить контроль, заставить других использовать свою централизованно управляемую систему. Мы наблюдаем это при работе с правительством и, в некоторой степени, с корпорациями. В ситуации, когда ни одна сторона не требует контроля, потребности в автоматизации просто нет[4].

Для решения вопросов автоматизации для межорганизационных процессов люди экспериментировали с децентрализованными рабочими процессами на протяжении более чем двух десятилетий[5]. В этих исследованиях и экспериментах предполагается высокий уровень доверия и честной игры,основное внимание уделяется решению технологических задач. На самом деле это ложное предположение, поскольку отсутствие доверия препятствует успешному выходу пилотных проектов на рынок.

Еще одной причиной отсутствия автоматизации является взаимосвязь между эффективностью и коррупцией[6]. Традиционно крупным корпорациям и правительственным органам требуется большое количество людей для выполнения процесса. Большой бюрократический аппарат нужен для координации таких процессов. Это увеличивает количество взяток, уменьшая стимул к переходу на автоматизированные процессы. Однако, повышение эффективности снижает этот недостаток.

В данном документе будет показано, как обе проблемы можно решить, используя блокчейн, представив решение, в котором все стороны могут участвовать на равных основаниях.

Содержание

Part I. Живые контракты

Моделирование бизнес-процессов — это общая стратегия для любых средних или крупных организаций[7]. Создание визуального представления позволяет анализировать, улучшать, и автоматизировать рабочие процессы (figure 1). В отличие от процедур, которые прописаны на естественном человеческом языке или языке программирования, эти модели могут быть понятны как людям, так и компьютерам.

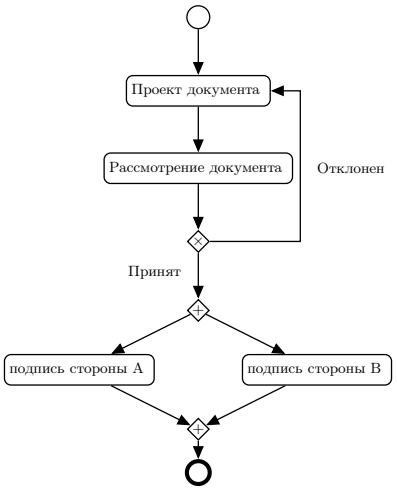


Рис. 1: Использование BPMN-диаграммы для визуализации рабочих потоков.

Для межорганизационного взаимодействия моделирование выполняется не только для улучшения коммуникаций. Вовлеченные стороны должны указать процесс, который будет служить в качестве обязательного соглашения[8]; на платформе LTO это называется Живой контракт (Live-контракт).

LTO создает специализированный под конкретный процесс приватный блокчейн, для каждого Живого Контракта. Такой блокчейн используется не как неизменный реестр, а как инструмент обеспечения всех сторон взаимосогласованной историей событий и согласованными общими состояниями.

1 Отличия живых контратков от смарт-контрактов

Живые контракты имеют цель аналогичную смарт-контрактам, реализованным в сети Ethereum[9]. Оба вида определяют и упрощают логику, которая может быть применена бездоверительным и проверяемым путем.

Однако философия этих цифровых контрактов существенно различается. Ethereum описывает смарт-контракты как криптографические «контейнеры», содержащие значение, которое будет разблокировано только при выполнении определенных условий[10].

Live-контракты не имеют прямого значения, но описывают, как две или более сторон должны взаимодействовать. Их назначение гораздо ближе к традиционному (бумажному) контракту.

1.1 Рикардианские контракты

Live-контракт соответствует определению рикардианского контракта [1][11]. В первую очередь, он легко читается как людьми, так и программами. Это новое свойство появляется из самого определения живого контракта. Это не версия договора на естественном языке для юридических целей и не версии на языке кода для исполнения программами.

1.2 Выполнение процесса

Выполнение процесса внутри блокчейна плохо подходит для многих случаев из реальной жизни. Смарт-контракты полагаются на проактивное принудительное исполнение, что означает, что нарушить соглашения невозможно, либо должна быть возможность прервать выполнение соглашения с каждой стороны [12].

В качестве примера возьмем соглашение о неразглашении. Блокчейн не может помешать участникам раскрывать информацию, а также не может заставить какую-либо сторону активно реагировать при возникновении нарушений. Для того, чтобы такой контракт работал в качестве самоисполняющегося соглашения[13], он должен удерживать некий штрафной депозит.Это обеспечивает заинтересованность каждой в разрешении конфликта при его возникновении.

Заморозка больших сумм в качестве штрафов при использовании контрактов для разрешения споров нецелесообразна для большинства организаций[14]. Кроме того, эффективность штрафов и аналогичных мер ограничиваются ценностью, поддерживаемой смарт-контрактом. Большинство бизнес-процессов требуют разрешения споров вне сети по средствам авторитетной третей сторону. Live-контракт облегчает разрешение спора. Он может включать переговоры о конфликтах, посредничество и даже арбитраж (третьей стороной или судьей).

Запуск процесса на платформе LTO формирует проверяемая история событий, уменьшая количество асимметричной информации. Распространение информации влияет на переговоры в случае спора[3] и влияет на оценку, проводимую независимой третьей стороной.

1.3 Пользовательский интерфейс

Ethereum предоставляет внутренний Тьюринг-полный язык написания скриптов, который может быть использован программистом для построения любых смарт-контрактов или типов транзакции, которые можно определить математически[10]. Данное свойство делает контракты очень абстрактными, поскольку состояние, содержащееся внутри действующего контракта, не имеет действительного значения.

Чтобы взаимодействовать с такими контрактами, необходимо создать пользовательский интерфейс для конкретного смарт-контракта или, точнее, интерфейс контракта[15]. Такие интерфейсы могут быть стандартизированы, как ERC-20[16] и ERC-721[16], с целью отказа от привязки интерфейса к логике контракта. Недостатком является то, что

1. Рикардианский контракт может быть определен как единый документ, который является a) договором, предлагаемым эмитентом держателям, б) ценным правом, удерживаемым держателем и управляемым эмитентом, в) легко читаемый людьми (например, договор на бумаге), d) читаемый программами (анализируемый как база данных), e) с цифровой подписью, f) передающий ключи и информацию о сервере и g) объединенный с уникальным и безопасным идентификатором.

такой подход ограничивает возможности при разработке контракта.

В живых контрактах информация имеет внутреннее действительное значение. Хотя это ограничивает варианты использования, данное свойство позволяет создать интерфейс, основанный исключительно на данных, представляющих контракт и его процесс. В результате любой рабочий процесс может быть оцифрован и выполнен в LTO без необходимости создания конкретного пользовательского интерфейса для каждого из них.

1.4 Инструкции и интеграции

Live-контракт содержит инструкции, предназначенные для конкретного человека или узла. Узел может действовать по инструкции или извещать пользователя, от которого ожидается выполнение действия. Такое действие может включать и получение информации из Интернета через HTTP API-вызов.

Логика умного контракта, реализованная в Ethereum и Hyperledger, способна только изменять состояние блокчейна. Смарт-контракт требует наличия оракулов, подтверждающих данные от внешнего источника в блокчейне. Этот оракул может воздействовать на состояние умного контракта, однако логика оракула не является частью контракта. Такая логика является частью Живого контракта, поэтому он может быть проверен и может оспариваться всеми участвующими сторонами.

2 Конечные автоматы

Live-контракт определяет рабочий процесс как диаграмму состояний конечного автомата (КА, FSM - finite state machine)[17]. Такой подход позволяет визуализировать его как блок-схему (figure 2). Таким образом, рабочий процесс становится понятным как людям, так и машинам.

2.1 Детерминированный конечный автомат

Любая блокчейн-логика должна быть детерминированной[18]. Где компьютерной программе могут потребоваться дополнительные усилия для выполнения вычислений, детерминированная КА (ДКА) останется детерминированной по определению.

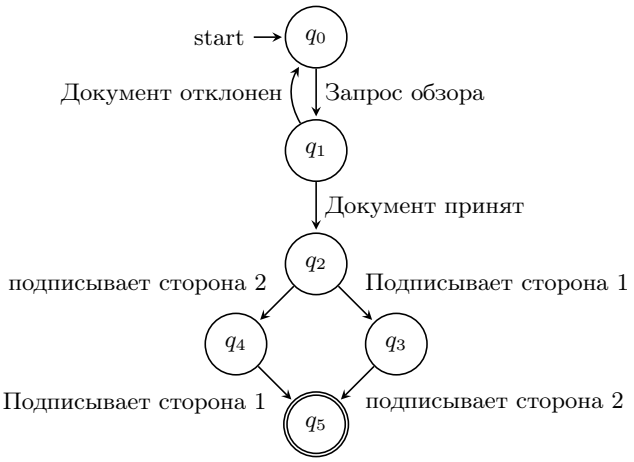


Рис. 2: Пример автомата конечного состояния, визуализированной в виде блок-схемы

2.2 Расширенный конечный автомат (РКА)

На рисунке 2 показана проблема, когда несколько действий в произвольном порядке приводят к одному состоянию. Такую модель можно представить как цепочку транзакций для каждого из возможных состояний, как это показано на 2. Однако, с таким подходом число состояний и переходов состояний будет расти экспоненциально количеству действий. Это не только делает визуализацию рабочего процесса менее ясной, но также делает определение рабочего процесса более сложным и подверженным ошибкам.

Вот почему вместо того, чтобы использовать обычный КА, Live-контракт использует Расширенный конечный атвомат[19] (РКА,EFSM), который позволяет использовать условные переходы между состояниями.

Figure 3 определяет тот же рабочий процесс, что и на рисунке 2 но с использованием РКА (EFSM).

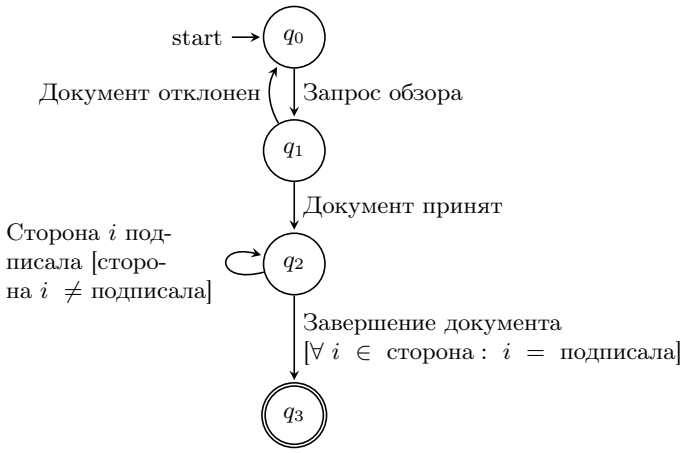


Рис. 3: Пример РКА: для совершения перехода из одного состояния в другое условия в скобках должны быть истинны

2.3 Коммуникационный конечный автомат

Конечные автоматы ограничены последовательным поведением; они не поддерживают выполнение параллельных процессов. Для представления параллельно выполняемых рабочих процессов каждая последовательность параллельных инструкций может быть представлена как отдельный КА (FSM). Коммуникационный конечный автомат (ККА) позволяет моделировать более сложные процессы путем объединения отдельных последовательностей событий.

Цепочку событий или эвентчейн (see section 9) можно представить как канал связи между двумя КА. Когда два процесса изолированы в различных эвентчейнах, канал связи между ними не является детерминированным, что, в свою очередь, делает всю систему недетерминированной[20]. Данное ограничение можно преодолеть подтверждением события, как показано в разделе 13.1.

2.4 Контракт как конечный автомат

Конечный автомат можно представить как соглашение между участниками путем формализации обязательств, разрешений и запретов, налагаемых сторонами друг на друга[21]. Контракты, подобные финансовым соглашениям[22] и сервисным контрактам[23] могут быть полностью оцифрованы в виде КА (FSM).

Однако представленные данные недостаточны для использования в качестве рабочих процессов, поскольку они

не могут определить механизм управления и взаимодействия в рамках процесса. Из сказанного выше можно сделать вывод, что моделирование КА станет гораздо более сложной задачей[24].

На практике КА лучше всего проявляет себя в качестве неполного контракта. И это совершенно не говорит о создании новой проблемы, поскольку пропуски могут быть устранены с помощью правил по умолчанию[25]. Система позволяет проводить пересмотр Live-контракта, либо разрешить конкретную ситуацию, либо решить проблему в целом, как показано в разделе8.

Также необходимо помнить о том, что не все действия в процессе представляют собой связывающий фактор. Например, на рисунке 1 принятие текста документа влечет за собой обязательства по исполнению требований документа; что происходит только тогда, когда документ подписан. Таким образом, действия можно классифицировать как носящие информационный характер и требующие выполнения[26].

3 Альтернативные методы моделирования

ККА (коммуникационные конечные автоматы) обычно используются для описания телекоммуникационных систем и других систем реального времени[27], но не бизнес-процессов.

Более общие представления рабочих процессов создают дополнительные трудности при моделировании децентрализованных межорганизационных процессов.

3.1 Сети Петри

Сети Петри[28] представляют собой графическое представление системы, в которой одновременно выполняются несколько независимых действий. Именно способность моделировать несколько параллельных действий отличает сети Петри от КА. В КА всегда есть одно «Текущее» состояние, определяющее, какое действие (действия) может произойти дальше. В сетях Петри может быть несколько состояний, и любое из них может измениться при изменении состояния сети Петри. Некоторые, или даже все из этих состояний могут развиваться параллельно, вызывая несколько независимых изменений в сети Петри одновременно[29].

Сети рабочих процессов (Workflow, WF-сети), которые могут использоваться для описания бизнес-процессов[30], являются подмножеством сетей Петри. WF-сети могут описывать весь процесс целиком, а не только одну из последовательностей его выполнения.

Расширенные КА используют глобальное состояние для хранения информации. Эта информация должна быть выделена для каждой последовательности выполнения. Если допустить возможность изменения данных, то это может быть использовано как показано в разделе5.1. В сетях Петри невозможно использовать глобальные состояния. Вместо этого, информация должна пройти через выполнение рабочего процесса.

При использовании коммуникационных КА (CFEMs), последовательности определяются как отдельные процессы. Это делает применение контроля доступа к части процесса простой задачей. Если рассматривать процесс как единое целое, то применение коммуникационных КА невозможно.

Сети Петри имеют интересное применение. Несколько исследований показывают, что они могут использоваться для представления бизнес-процесса. Учитывая, что коммуникационный КА можно смоделировать как сеть Петри, использование сетей рабочих потоков может быть предпочтительнее отдельных КА.

Из-за сходства между сетями Петри и КА, переключение на сети рабочих процессов (WF-nets) принципиально не изменит наш продукт, о чем и будет говориться далее.

3.2 BPMN

BPMN — это отраслевой стандарт обработки бизнес-моделей. Он будет стандартом моделирования в LTO. Однако существует ряд ограничений, которые особенно сложны для межорганизационных систем[31].

Язык исполнения бизнес-процесса (BPEL), как правило, связанный с BPMN, является недетерминированным Тьюринг-полным языком для веб-сервисов[32]. Это делает его непригодным для автоматизации в блокчейне

Предлагаемая ему альтернатива заключается в том, чтобы перенести модель BPMN в сеть Петри, которую, в свою очередь, будет переведена в смарт-контракт[33].

Хотя перевод на (Тьюринг-полный) умный контракт не требуется, переход с BPMN на сеть Петри (или Коммуникационный КА, CEFSM) может быть интересным решением с точки зрения поддержки текущего отраслевого стандарта.

3.3 DEMO

DEMO — акроним “Design & Engineering Methodology for Organizations” (Методика проектирования и разработки для организаций). Это методология для описания организаций и их бизнес-процессов на основе «коммуникативного действия». Она использует четыре модели для создания целостного представления, а именно Модель построения (CM - Construction Model), Модель процесса (PM - Process Model), Модель действия (AM - Action Model) и Модель факта (FM -Fact Model)[34].

Вместо того, чтобы рассматривать каждый шаг процесса отдельно, DEMO устанавливает обобщенный рабочий процесс для каждой выполняемой транзакции. В такой транзакции есть две роли: инициатор и исполнитель. Стандартная последовательность выполнения выглядит следующим образом: инициатор делает запрос, а исполнитель «дает обещание». Затем исполнитель выполняет действие и публикует результат; инициатор либо принимает его, завершая транзакцию, либо отклоняет ее[26].

Альтернативные потоки также моделируются, например, исполнитель отклоняет запрос, инициатор отзывает запрос, исполнитель не может «выполнить обещание» и т. д. При использовании других методологий моделирования такие альтернативные потоки часто не моделируются или моделируются частично, в то время как на практике, они всегда присутствуют.

Модель процесса (PM) объединяет все эти транзакции, чтобы смоделировать полный бизнес-процесс. Разница между этой моделью и рабочим процессом состоит в том, что в этой модели все работают параллельно, указывая взаимосвязи между транзакциями, где это необходимо. По сравнению с другими методами моделирования, такой его вид означает, что DEMO не даёт четкого понятия того, где именно мы находимся внутри процесса. Более того, взаимное исключение информации (не редактируйте документ, который готов к подписанию) не поддерживается. Наоборот, всё должно быть конкретно определено.

DEMO представляет собой хороший инструмент для построения высокоуровневой модели, в которой определяемый рабочий процесс может отлично настроен. Всё это должно приводить к созданию более полных контрактов, уменьшая зависимость от отклонений (см. раздел 8.2).

4 Сценарий

Рабочий процесс определяется как объект данных; the scenario. Он состоит из следующих элементов:

- q_x состояние с q_0 в качестве начального состояния,
- Q как множество всех возможных состояний $Q = \{q_0, ..., q_{n-1}\}$,
- σ_x как действие,
- Σ как множество всех возможных действий $\Sigma = \{\sigma_1, ..., \sigma_n\}$,
- δ в качестве функции перехода $\delta : Q \times \Sigma \to Q$,
- F как множество конечных состояний с $F \subset Q \mid F = \emptyset$,
- $\bar{I}$ как определение всех участников,
- $\bar{A}$ как определение всех активов,
- D как определение всех активов,

4.1 Состояния

Состояния в наборе Q обычно складываются из следующего:

- заголовок: короткий заголовок для состояния,
- набор действий с транзакцией как $\{(\overrightarrow{q_x}, \delta), ...\}$,
- описание: подробное описание состояния,
- карта или инструкции для конкретных участников.

Состояние описывает действия, которые могут быть выполнены в этом состоянии, и переходы из него. Это позволяет использовать конкретные действия в различных состояниях.

4.2 Действия

Сценарий определяет Σ, набор всех возможных действий, которые могут быть выполнены в рабочем процессе. Такими действиями могут являться, например, заполнение формы, рассмотрение документа и выполнение HTTP-вызова. Тип действия и свойства объекта прописываются в JSON-схеме.

Действие определяет, кто из участников, определенных в наборе I может его выполнить, и, опционально, - ограничения на выполнение. Эти ограничения делают сценарий Расширенным КА.

Когда действие выполняется, то происходит переход из одного состояния в другое. Действия подразделяются на выполняемые вручную, которые требуют вмешательства человека для выполнения, и системные действия, автоматически выполняемые системой.

При желании действия могут дополняться инструкциями для обновления списка участников и активов с использованием данных из таблиц.

4.3 Участники

Набор $\bar{I}$ определяет всех участников, которые могут играть свои роли в выполняемом процессе. Каждый участник определяется как объект с помощью JSON-схемы. И за счет этого участники, относящиеся к данному процессу, имеют свои свойства.

Участник в сценарии — это только статическое определение, экземпляр которого может быть создан в процессе.

4.4 Активы

$\bar{A}$ определяет все активы, доступные в процессе. Актив – это переменный объект данных. Свойства актива, относящегося к данному процессу, должны быть определены.

Обратите внимание, что сценарий только определяет структуру активов. Активы могут быть созданы только внутри процесса.

5 Объекты данных (Data-objects)

Помимо сценариев, могут быть определены другие типы объектов данных. Все объекты данных, включая сценарии, используют JSON-схему для определения типа. Примерами объектов данных являются формы, документы и шаблоны.

Объекты данных могут быть встроены в процесс или связаны с ним, но хранится независимо.

Связанные объекты идентифицируются SHA256 хэшем их JSON-представления. Чтобы JSON-кодирования данных всегда давало один и тот же результат, применяется детерминированный метод JSON-кодирования.

5.1 Неизменяемость

Объекты данных неизменяемы в том смысле, что когда если в него вносятся изменения, то создается новый объект данных. Если этот объект данных внедряется в процесс как актив, то старый объект заменяется на модифицированный. Примечательно, что, если объект данных доступен в нескольких процессах, изменения над объектом, произведенные в одном процессе, автоматически распространяются на другие процессы. Несоблюдение этого может привести к уязвимости. На рисунке 1 мы продемонстрировали процесс переговоров и подписание документа. Само собой, документ не должен меняться во время выполнения последовательности подписи.

5.2 Формы

Определения формы используют JSON-схему для определения структуры данных, которая должна возникнуть в результате заполнения формы. Опционально, используется дополнительная схема пользовательского интерфейса для указания того, как может отображаться соответствующее поле.

Для этого существует несколько похожих реализаций[35—38]. Наша цель – вести работу вместе с этими проектами, чтобы сформировать единый стандарт.

5.3 Документы

Цифровые рабочие процессы могут в значительной степени устранить необходимость в бумажной документации. Однако такие вещи, как соблюдение правовых норм, обратная совместимость и политика компании могут по-прежнему требовать использования документов. Определяя шаблоны как часть Живого Контракта, документы на естественном языке могут быть сгенерированы с использованием данные, собранные в процессе.

Мы рекомендуем использовать ODF-формат[39], который поддерживает поля и условные разделы для создания шаблонов.

5.4 Типы пользователей

Любая JSON-схема, которая определяет объект, может использоваться как тип объекта данных. Пользовательские типы влекут за собой риск неправильного выполнения рабочего процесса, поскольку другие стороны могут участвовать через узел, который не поддерживает данный тип. Данные с неизвестными типами будут храниться «как есть» и являются недоступными вне контекста процесса.

6 Личности

Личность определяет человека, команду или организацию в пределах Live-контракта. Личность всегда содержит следующую информацию:

- идентификатор,
- URI узла,
- пользовательская информация,
- знаковые клавиши,
- ключ шифрования.

Личность — это не то же самое, что и участники. Участник – абстрактная роль, например «студент»; однако, личность может иметь конкретное имя "Bruce Willis"или "Acme Corp".

Подписные ключи — это метки с одним или несколькими открытыми ключами, привязанными к конкретной личности. Ключ «пользователя» относится к личности и может быть использован только им/ей, для подписания действия. «Системный» ключ принадлежит узлу, который используется личностью. Он нужен для подписи автоматизированных действий. Другие типы ключей могут быть определены в пределах процесса.

Публичный (открытый) ключ шифрования может использоваться для шифрования данных, которые могут быть расшифрованы только данной личностью.

6.1 Приглашение личностей

Чтобы добавить стороны в процесс, сценарий должен определить действия для добавления других личностей. Если открытые ключи известны в пределах процесса, личность может быть добавлена напрямую.

Когда открытый ключ неизвестен, личность необходимо пригласить (figure 4). Это можно сделать с помощью любых средств, считающихся достаточно безопасными, включая электронную почту. Приглашающая система генерирует одноразовый ключ и отправляет его приглашенной личности. Приглашенная сторона должна заменить этот ключ своими безопасным «пользовательским» и «системным» ключами.

Прежде чем новая личность сможет полностью участвовать в процессе, может потребоваться дополнительная проверка подлинности. Это может быть как проверка по средствам SMS, так и даже нотариальное заверение для удостоверений личности.

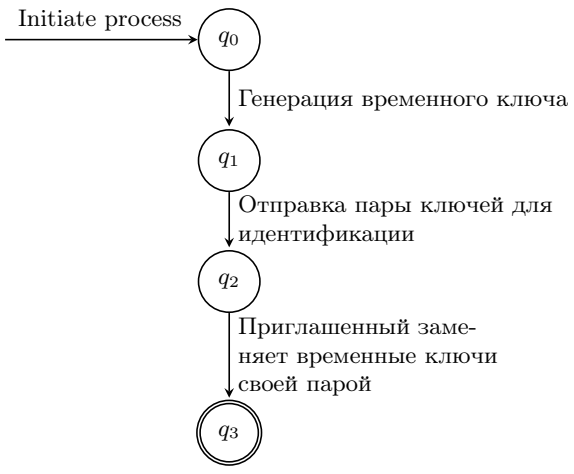


Рис. 4: Процесс приглашения личности

6.2 Изменение информации о личности

Личность имеет право изменять свою собственную информацию, за исключением идентификатора. Что позволяет одной из сторон переключиться на другой узел. В случаях, когда пользователю не разрешается переключаться между узлами, узел личности принимает решение об отклонении изменений. Удаление личности может быть выполнено путем очистки ключей идентификации, ключа шифрования и URI узла.

Обновление других личностей возможно только в том случае, если такое действие определяется в сценарии и разрешается в текущем состоянии.

7 Процессы

Если сценарий — это определение рабочего процесса без учета состояния, то процесс представляет собой спецификацию с заданным состоянием, состоящую из следующего:

- θ_x ответ, где $f : q_x \mapsto \theta_x$,
- Θ упорядоченный список всех ответов $\Theta = \{\theta_0, ...\theta_n\}$,
- q_t текущее состояние,
- I текущее состояние,
- A набор созданных активов.

7.1 Действия

Выполнение действия всегда дает отклик, ответ. Этот ответ должен быть подписан участником и подтвержден как новое событие. Узлы независимо определяют новое состояние на основе текущего состояния и того выполненного действия.

Сценарий определяется как детерминированный КА. Тем не менее, это касается только переходов состояний и проекции. В системах подобных Ethereum и Hyperledger, вся логика должна быть детерминированной, поскольку она выполняется всеми узлами и должна предоставлять одинаковые результаты вычислений по всем системам[40].

При использовании LTO только один узел или один участник выполняет действие, то есть действия не должны быть детерминированными. Как объясняется в разделе 1.4, для выборки данных из внешнего источника оракулы не нужны.

7.2 Действия, выполняемые вручную

Приложения, созданные на платформе LTO, должны информировать участников о действиях, которые они могут выполнять в текущем состоянии. Участник-человек подписывает свое собственное событие, прежде чем подтвердить его на собственном узле, который затем распространит его всем участникам.

7.3 Системные действия

Системные действия не требуют вмешательства человека, а выполняются узлами автоматически. Поэтому вместо человека, ответ подписывает узел.

Эти действия всегда выполняются одной системой и не должны быть детерминированными. Другие стороны процесса могут подтвердить или, при необходимости, отклонить ответ. Поскольку человек не вмешивается в действия системы, то действия подписываются самой системой, а не участниками.

Также возможно запланировать отложенное выполнение системного действия. Это прописывается в сценарии. Таким образом, появляется возможность взять тайм-аут или обратиться к внешнему источнику с предопределенной периодичностью.

Также возможно запланировать отложенное выполнение системного действия. Это прописывается в сценарии. Таким образом, появляется возможность взять тайм-аут или обратиться к внешнему источнику с предопределенной периодичностью.

7.4 Подпроцессы

Процесс, основанный на КА, может пребывать в конкретный момент времени только в одном состоянии. Подпроцессы позволяют live-контракту поддерживать несколько состояний и позволяют выполнять различные процедуры параллельно. Несмотря на то, что процессы подчиняются эвентчейну, данные каждого процесса по-прежнему остаются изолированными.

Чтобы задействовать подпроцессы, Live-контракт может содержать подзадачи, которые могут быть прописаны в основном сценарии.

7.5 Проекция

Помимо состояния КА (FSM), процесс также содержит другие данные о состоянии, такие как активы и участники. Полезная нагрузка каждого ответа может быть использована для обновления этих данных. Правила того, как полезная нагрузка обновляет данные, определены в сценарии. Обновление проекции детерминировано и, следовательно, применение данного набора ответов из сценария всегда будет давать ту же самую проекцию.

Проекцию можно использовать для установки параметров действия, а также для ограничений, определенных для действия Extended Finite State Machine.

7.6 Операторы данных

Операторы данных могут использоваться в сценарии для определения того, как проекция влияет на процесс. Эти операторы являются детерминированными функциями без сторонних эффектов. Они могут использоваться для арифметических или логических операций. Результат этих операций может храниться в проекции и могут использоваться в качестве основы для, например, перехода состояний.

7.7 Пассивное тестирование

Сценарий, содержащий цикл, состоящий исключительно из системных действий, может привести к бесконечному циклу, вызывая огромное количество транзакций. При проверке сценария, мы отклоняем его, если встречаем в нём подобную конструкцию.

Определение того, может ли программа работать бесконечно, называется проблемой остановки[41]. Была доказана неразрешимость проблемы для Тьюринг-полных систем; однако ее можно решить для КА [42]. Поскольку КА имеет конечное число путей перехода, все они могут быть проверены на наличие замкнутых циклов.

Пассивные тесты для моделей РКА осложняются наличием неосуществимых путей, что являляется проблемой для дальнейших исследований[19, 43]. Для простоты мы можем определить любой возможный путь КА как осуществимый, игнорируя определенные условия. При таком предположении, мы принимаем, что этот подход может вызвать ложноположительные результаты проверки.

8 Адаптивные рабочие процессы

Сценарий будет моделировать наиболее вероятные варианты развития процесса. Но невозможно заранее предвидеть все ситуации, а моделировать все возможные крайние случаи достаточно утомительно. Принятие подхода «код-есть-закон» сделает систему жесткой. Вместо этого Live-контракт поддерживает три метода решения таких задач.

- Комментарии,
- Отклонение,
- Обновления сценария.

8.1 Комментарии

Комментарии используются для связи с другими личностями. Они могут, например, использоваться для разрешения конфликтов или проводить дискуссии вне процесса. Использование комментариев вместо оффчейн-методов коммуникации гарантирует, что все переговоры регистрируются на блокчейне. Это также позволяет отслеживать, в процедуре определенные разговоры по времени.

Комментарии не ограничиваются текстовыми сообщениями. Также можно использовать изображения или документы для облегчения общения. Комментарии не являются частью процесса, а это означает, что добавление комментарий не вызывает переход состояния. Следовательно, всегда можно провести дискуссию по темам, которые не были предопределены в процедуре.

8.2 Отклонение

Любая сторона может предложить отклонение от основного информационного потока, определив частичный сценарий. Этот подпоток должен начинаться с одного из состояний существующего сценария и заканчиваться состоянием этого сценария. Потоки отклонений выполняются только один раз, после возвращения процесса в существующее состояние они перестают быть доступными.

Все стороны должны договориться об отклонении. Обратите внимание, что отклонения могут привести к появлению форков, которые могут быть разрешены только благодаря ручному вмешательству в разрешение конфликтов.

Отклонение может быть использовано для разрешения споров. Любая сторона может предложить оспорить правильность предыдущего события и представить решение о том, как это исправить.

Типичным случаем использования отклонений является договоренность об оплате. Организации, очевидно, не хотят, чтобы эта информация распространилась за пределы соглашения. Предопределенные подпотоки позволяют гарантировать такие договоренности, сохраняя их в секрете.

8.3 Обновления сценария

Для текущего процесса всегда может потребоваться изменение сценария; например, когда обновляется соглашение или принимается новый закон.

Сторона может предоставить новый сценарий для данного процесса через поток отклонения. Этот поток перемещает актуальное состояние из устаревшего сценария в новый. Поскольку обновление сценария является частью последовательности событий, наше решение остается детерминированным.

9 Эвентчейн или цепь событий

Чтобы определить состояние КА и проекции, нам нужно обработать набор ответов в заданном порядке. Вставка или удаление события, изменение порядка событий или изменение полезной нагрузки может привести к совершенно другому состоянию.

Для централизованных решений, контролирующая сторона несет ответственность за целостность данных. Все стороны полагаются на контролирующую, поскольку она представляет единственный источник истины. В децентрализованной системе власть и ответственность разделяют все стороны.

Чтобы способствовать этому, эвентчейн работает как специальный приватный блокчейн. Каждый ответ инкапсулируется в событие, которое можно рассматривать как блок с одним действием. Эти события образуют цепочку хэшей, которая распределяется между сторонами. Алгоритм консенсуса гарантирует согласие сторон над последовательностью событий.

9.1 Приватный блокчейн

В рамках сценария лица и организации определяются как личность. Личность может свободно выбирать узел, который будет принимать участие в процессе.

Для узла не требуется никаких разрешений для присоединения к сети. Не требуется никаких разрешений для запуска процесса и приглашения других участников. Данные распространяются только между участниками процесса. Таким образом, эвентчейн можно описать как приватный блокчейн со свободным доступом.

9.2 Криптографические подписи

Чтобы гарантировать, что никто не может фальсифицировать или подделывать события других, перед отправкой каждое событие подписывается с использованием асимметричной криптографии. Подписанное событие также служит своего рода распиской, позволяющей другим сторонам доказать, что действие было выполнено именно подписавшей личностью.

На платформе используются подписи ED25519[44]. Этот протокол цифровых подписей на основе эллиптических кривых широко применяется, поддерживается и исследуется такими организациями, как NIST[45] и ENISA[46]. Криптография с использованием эллиптической кривой позволяет ускорить однократную проверку сигнатуры и подписания без потери безопасности. Также она уменьшает необходимый размер ключей и подписей. Обратите внимание, что этот метод сам по себе не обеспечивает полной безопасности, поскольку участники все еще могут фальсифицировать или создавать свои собственные события. Другими словами, криптографические подписи не могут доказать, что событие не произойдет.

9.3 Хэшчейн

Каждое событие может быть однозначно идентифицировано с использованием его SHA-2 256 битного хэширования. Этот алгоритм, являющийся стандартом в индустрии, гарантирует быструю вычислимость и устойчивость к атакам нахождения прообразов криптографической хеш-функции и коллизиям[47]. Именно этот криптографический алгоритм хэширования рекомендуется NIST[48].

Вложение хэша предыдущего события в хэш следующего события создает цепочку хеширования, которая записывает хронологию событий. В сочетании с использованием криптографических подписей, цепь хеширования обеспечивает адекватное доказательство того, что конкретная последовательность событий привела к текущему состоянию[49].

10 Распространение

Вместо того, чтобы требовать от сторон получения информации от центрального сервера или друг от друга, каждая сторона несет ответственность за внесение события в систему для всех других вовлеченных сторон.

Системы должны быть всегда доступны, чтобы события не терялись. Использование развязок и очереди сообщений снижает вероятность потерь при временной недоступности узла. В типичном случае все стороны будут подключаться к узлу, которому они доверяют. Данный узел будет получать и обрабатывать события для этих сторон. Такой узел является частью более крупной системы (см. раздел 19.1).

Уделяя особое внимание организации и управлению, компании сами принимают решения о запуске узла. Для участия в процессе пользователи сами выбирают, подключаться ли к узлу своей организации или к общедоступному узлу.

Теорема CAP подразумевает, что при наличии разделения сети, нужно выбирать между согласованностью и доступностью. Используя развязки, мы жертвуем согласованностью для более высокой доступности. Узел, который поврежден или недоступен, не прерывает процесс для других участников и будет просто синхронизирован, когда снова станет доступным.

10.1 Приватный чейн

Цепочка событий или эвентчейн — это приватная цепочка, которая распределена только между узлами, выбранными личностями. Узлы не знают информации приватных сетей, частью которых они не являются.

Узел не только хранит множество эвентчейнов, но и взаимодействует с ними. В отличие от сайдчейнов, эвентчейны полностью изолированы. Эвентчейны не влияют друг на друга напрямую. Это позволяет использовать горизонтальное масштабирование, учитывая, что активность на эвентчейн – значение низкое, в пределах разумного.

10.2 Генезис

Любой может создать новую цепочку событий по своему усмотрению. Генезис-блок данной цепочки содержит идентификатор пользователя, который создает процесс, а следующий блок содержит сценарий. Другие личности будут приглашены в данный приватный блокчейн как часть сценария.

11 Механизм консесуса

LTO — это распределенная система, в которой все стороны могут участвовать через собственный узел. Узлы распространяют все события через своих пиров, которые, в свою очередь, обрабатывают эти события. Это означает, что существует короткий момент, когда состояние процесса между узлами отличается. Согласованность в конечном счёте[50] гарантирует, что в итоге, учитывая, что нового подтвержденного события не было, состояние процесса на всех узлах будет одинаковым.

Однако, иногда, новые события передаются до того, как согласованность достигнута. В этот момент, возможно ситуация, когда два или более узлов добавляют событие в

эвентчейн. В случае возникновения такой ситуации[51], все узлы считают, что их информация действительна; однако в целом система находится в противоречивом состоянии. В этом состоянии, узлы больше не принимают новые события друг от друга; Узлы должны иметь возможность прийти к консенсусу, а не остановить работу сети.

Распределенные приложения используют для этого разные алгоритмы консенсуса. В общем, это консенсус типа Byzantine fault tolerance (BFT).Ранние версии BFT-консенсуса плохо масштабировались [52]. Изобретение алгоритмов консенсуса, обеспечивающих лучшее масштабирование, таких как «доказательство работы» proof-of-work [53],"доказательство доли"proof-of-stake [54] и "доказательство авторизации"proof-of-authorization [55], позволили создать распределенные сети с большим количеством участников; это явление также называется технологией распределенного реестра (DLT - Distributed Ledger Technology)

Хотя перечисленные алгоритмы консенсусов намного лучше, чем традиционные BFT подходы, им требуется относительно большое количество участников для обеспечения безопасной работы сети (для PoW > 1000). Традиционный BFT подход основывается на том, что не более [(n-1)/3] участников являются скомпрометированными. Это означает, что в системах с менее чем четырьмя участниками, один злоумышленник в сети может оказать влияние на работу системы. А для защиты сети от двух злоумышленников требуется не менее 7 участников.

Эвентчейн — это приватный блокчейн с относительно небольшим числом участников, часто их менее семи, что означает их высокую уязвимость при использовании BFT-консенсуса. Вместо того, чтобы доверять большинству голосов, узлы проверяют корректность их состояния, если не доказано иное.

11.1 Вероятность конфликта

Эвентчейны полагаются на оптимистичный сценарий параллельной обработки событий; большое количество конфликтов является серьёзным испытанием для достижения консенсуса, который может стать относительно медленным, поскольку, возможно, придется долго ждать нового сгенерированного блока.

Мы обозначаем распределенный эвентчейн следующим образом:

- Пусть N - множество сущностей $\{n_1, n_2, n_3, \dots\}$ добавляющих события в эвентчейн.
- Пусть C_n - эвентчейн, последовательность, состоящая из событий $(e_1, e_2, e_3, \dots)$, принадлежащих сущности n, и пусть C будет множеством всех копий цепочки событий $\{C_n | n \in N\}$.
- Далее определим конфликт или ветвь как $\exists\, i, j \in N : i \neq j,\ C_{i_0} = C_{j_0}, C_i \not\subset C_j,\ C_j \not\subset C_i$.

Чтобы конфликт произошел случайно, две стороны должны добавить блок к своей цепочке, прежде чем они получат обновление от другой цепочки.

Давайте допустим, чтобы кто-то распространил обновление по цепи P (x). Вероятность этого события зависит от количества блоков, которые были добавлены к цепочке в течение заданного интервала времени, времени, необходимого для распространения этого блока по остальной сети, а также количества личностей, присоединившихся к эвентчейну в рамках текущего временного интервала. Предполагая, что все в равной степени добавляют события в сеть, можно получить формулу (1)

$$P(x) = \frac{f \cdot t}{n} \tag{1}$$

где:
f = Общее количество транзакций в данном временном отрезке
n = Общее количество активных участников
t = время, необходимое для распространения блока в остальной части сети

Данную вероятность можно использовать для вычисления вероятности возникновения конфликта. Эта вероятность определяется путем вычитания шанса невозникновения конфликта из 1. Когда нет конфликта, это означает, что либо никто не добавил событие в отведенный промежуток времени, вероятность которого рассчитывается с использованием формулы (2),

$$(1 - P(x))^n \tag{2}$$

либо в случае, если только один узел добавил событие в эвентчейн, вероятность этого рассчитывается с использованием формулы (3).

$$P(x) \cdot (1 - P(x))^{n-1} \cdot n. \tag{3}$$

Поэтому вероятность конфликта рассчитывается по формуле (4).

$$P(c) = 1 - (1 - P(x))^n - P(x) \cdot (1 - P(x))^{n-1} \cdot n. \tag{4}$$

С сетевой задержкой в 1200 мс мы видим, что вероятность конфликта составляет $< 2\%$:

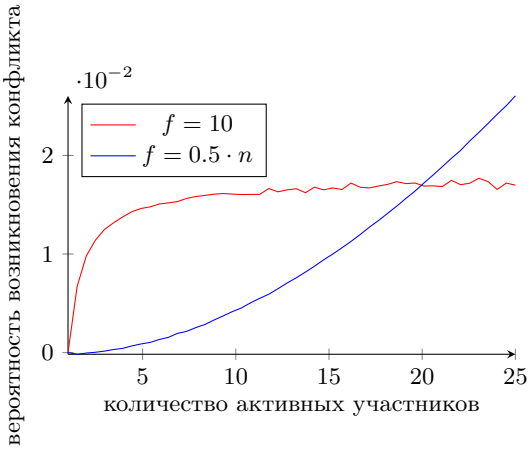


Рис. 5: This plot shows how the chance of a conflict occurring increases when the number of participants increases. However, it stabilizes if the total number of transactions stays the same.

Figure 5 shows that the chance of a conflict occurring is more or less constant when the number of participants increases but the total number of transactions stay the same. However, when the number of participants increases usually the number of transactions increase as well. For example, more participants may lead to more comments. When this is the case, the chance of a conflict increases exponentially with the number of participants.

LTO работает с очень небольшим количеством активных участников в одном эвентчейне, что уменьшает вероятность

конфликта. С числом участников сети 5 или более, значение вероятности становится нерелевантным. С более чем 10 участниками, основываясь на значениях сетевой задержки и частоты транзакций, вероятность конфликта становятся практически линейной функцией.

Если использовать индивидуальное разделение эвентчейнов с разделением сообщений, частота транзакций станет очень низкой. Это снизит вероятность конфликта.

11.2 Валидация ветвей

Только узел может доказать, что другая сторона знала о цепочке событий вплоть до последнего момента. После последней фиксации состояния любая сторона может создать ответвление в цепи. Если сторона пытается разветвить цепочку до точки, в которой произошло последнее событие, то такая ветвь автоматически отбрасываются всеми (другими) сторонами, а событие регистрируется.

Прежде чем принимать новые события из конфликтующей ветви, они проверяются наравне с любым полученным набором событий. Событие должно быть правильно подписано одной из личностей и должно быть надлежащим образом закреплено. Если временная метка события отличается больше чем на 300 секунд от метки времени якоря, событие может быть отклонено.

11.3 Порядок анкоринга (закрепления)

Узлы должны привязывать события к глобальному блокчейну. Порядок блоков устанавливается майнингом, порядок транзакций внутри, полученных майнингом блоков, также фиксирован.

Это позволяет нам использовать порядок якорей в глобальном блокчейне для определения последовательности событий. В случае конфликта должен быть принят блок, который был закреплен первым. Консенсус в приватном эвентчейне достигается посредством консенсуса в публичном блокчейне. В публичном блокчейне консенсус достигается анонимным сотрудничеством между большим числом участников, использующих вариацию PoS-консенсуса.

11.4 Приоритет

Может потребоваться указать приоритет действий или участников, так что его блок будет учтен первым, даже если появился последним. Можно настроить такие приоритеты в сценарии.

Некоторые типы событий, такие как комментарии, естественно, имеют более низкий приоритет.

Использование приоритетов исключает атаки, при которых участник может ответить на событие, создав новую ветку, которая, в последствии, отменит это событие. Приоритеты должны быть использованы в тех случаях, когда они не создают проблемы.

11.5 Незакрепленные события

Когда блок получен, но еще не закреплен, один из участников может принять решение принять блок в любом случае. Это происходит только в случае, если принятие блока не создает конфликт. Если Привязка блока было просто отложено, то принятие блока предотвратит ненужные задержки в самом процессе. С другой стороны, если блок никогда не будет закреплен, никаких реальных проблем не возникает. Если все принимают блок, процесс может продолжаться как обычно, а блок может быть закреплен позже.

11.6 Объединение ветвей

При возникновении форка большинство приложений выбирают одну цепочку для продолжения работы и игнорируют все, что происходит на других ветвях. В блокчейне, подобном биткойну, все транзакции в конце концов будут включены в блоки каждой ветви.

В эвентчейне события сами создают хэшчейн. Выбор одной из ветвей приведет к потере информации о выполненном действии. Вместо этого, когда узлу становится известно о другой ветви, которая имеет приоритет над его собственной ветвью, он должен расположить события, которые хранит локально, поверх другой цепочки. Это похоже на действие переадресации при использовании git[56].

11.7 Форки

Несмотря на то, что существует гарантированный способ достижения консенсуса, участвующая сторона может решить игнорировать другие цепочки и поддержать разделение цепи. Для большинства блокчейн-приложений, таких как Ethereum[57], нет оснований вмешиваться в это, поскольку ценность имеет только участие в основной цепочке.

Живые контракты - инструмент для оцифровки и частичной автоматизации бизнес процессовпроцессов. Несмотря на то, что блокчейн позволяет существовать форкам, эти процессы обычно отсутствуют. В случае возникновения форка, стороны могут начать вторичную процедуру разрешения конфликта, запускаемую вручную.

12 Приватность

LTO построен для выполнения процессов между сторонами. Кроме этих сторон никому не нужно знать о процессе или даже о самом взаимодействии сторон.

Публичные блокчейны поддерживают существование анонимных аккаунтов; тем не менее, такие аккаунты функционируют как псевдонимы. Любая транзакция может раскрыть личности участников, использующих аккаунт, показав полную историю транзакций для него. Смарт-контракты требуют, чтобы данные были общедоступными, поскольку это необходимо для каждого узла. Что касается блокчейна, то все участники оповещаются о действиях друг друга.

Специальные приватные блокчейны позволяют использовать уникальный случайный набор участников. Это позволяет взаимодействовать без предоставления информации общественности и её подтверждения. Такие блокчейны могут быть полностью удалены после завершения выполнения процесса.

12.1 Связанные данные

Каждая сторона подключается через свой собственный узел или узел, которому сторона доверяет. У каждого узла есть приватный сервис хранения, где пользователи могут хранить данные. Пользователи имеют полный контроль над данными, располагающимися в подобных хранилищах, аналогично данным, хранящимся на таких сервисах как DropBox. Пользователи могут удалить свои персональные данные в любое время. Данные не передаются и не обрабатываются без соглашения об обработке данных, которое явно подтверждается пользователем.

Когда действие приводит к связанным данным, эти данные не распространяются напрямую другим сторонам; в блокчейн добавляется только соответствующий хэш. LTO

предотвращает использование хэша для всего, кроме проверки полученных данных, помещая их в пакет вместе с меткой времени и некоторыми случайными данными.

Хэш, созданный для формирования пакета, никогда не появится в блокчейне более одного раза.

Когда организация указывает, что она хочет выполнить действие, требующее связных данных, узел этой организации автоматически сформирует запрос. Узел владельца данных проверит, является ли указанное действие валидным и, таким образом, проверит, может ли данный пользователь при текущем состоянии выполнять подобное действие.

12.2 GDPR

С появлением нового закона GDPR[58] (General Data Protection Regulation - Положение об общем регулировании персональных данных) в Европе, был поднят спор о том, что многие приложения на блокчейне не соответствуют требованиям GDPR [59]. На то есть две основные причины:

- неизменяемая природа блокчейна находится в противоречии к праву как изменять, так и удалять данные,
- не существует специального назначенного контроллера над данными, поскольку работа идет в распределенной среде.

Связанные данные означают, что узел, выбранный стороной для организации взаимодействия, будет действовать как контроллер над данными этого пользователя. Все другие стороны всегда функционируют как процессоры данных. Запросы данных автоматически форматируются для создания корректного соглашения об обработке данных, которое включает цель и время, требуемые для обработки данных.

Если потребуется, специальный блокчейн при этом позволяет удалить всё содержимое цепи.

Проще говоря, функции конфиденциальности LegalThings позволяют приложениям соответствовать GDRP без лишних костылей.

12.3 Доказательство нулевого знания

Сценарий может потребовать от одной стороны доказать другой стороне, что ее знание несет особую ценность. Доказательство нулевого знания (zk (zero-knowledge)-доказательство) представляет собой способ сделать это без передачи какой-либо информации, кроме той, что сторона обладает этой ценностью.

LTO поддерживает zk-proofs через интерактивную систему проверки доказательств. Две стороны, доказывающая и проверяющая, обмениваются сообщениями с целью убеждения проверяющего в честности (полноте) доказывающего и разоблачающий нечестного доказывающего проверяющим ([60].

Zk-proof для живого контракта - это всегда действие между двумя сторонами. Нет необходимости в zk-proofs без взаимодействия, таких как zk-SNARKS, которые всё ещё считаются экспериментальными.

13 Последовательности общего типа

13.1 Взаимодействие цепей между собой

Некоторым процессам для собственного исполнения, возможно, потребуется взаимодействовать с другими процессами; например, в случае необходимости одному процессу получить разрешение другого процесса на продолжение или получить результат процесса разрешения конфликта. Запрос данных из другого процесса выполняется, согласно схема, приведенной на рисунке 6.

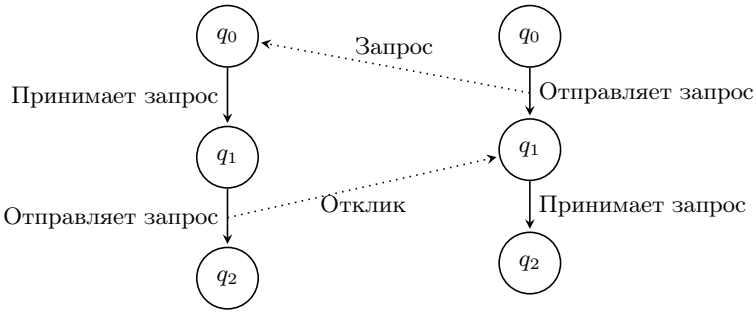


Рис. 6: Пример взаимодействия двух процессов.

13.2 Синхронизация, заданная в явном виде

В некоторых случаях для выполнения процесса может быть особенно критичным событие, которое передается узлам сети слишком поздно. Для таких случаев в сценарий может быть встроена явная синхронизация. Она требует от всех сторон подтверждения текущего состояния, перед тем как продолжить процесс. Это последовательность внутри КА, которая предназначена для предотвращения форков для событий, предшествующих конкретному событию. В случае возникновения спора или конфликта, такое подтверждение может быть использовано для выбора цепи, которая будет использоваться для продолжения выполнения процесса.

Явная синхронизация работает только в том случае, если все стороны признают текущее состояние процесса. В случае, если сторонам не хватает стимулов для продолжения или для разделения процесса, предлагается иное решение. В этом случае сторона, которая требует продолжения процесса объявляет о своем намерении всем другим сторонам. Если сторона, получившая такое сообщение, замечает, что заявленное действие является недействительным в ее блокчейне, она может транслировать эту информацию остальным участникам сети. Это означает, что произошел форк цепи и должен быть применен стандартный алгоритм разрешения конфликтов. Чтобы убрать влияние задержки сети во время трансляции сообщений о форках, время ожидания соответствует установленному времени отмены сообщения.

Part II. Глобальный блокчейн

Глобальный блокчейн LTO является публичным блокчейном со свободным доступом, задачей которого является верификации информации. Он существует для поддержки live-контрактов и приватных эвентчейнов. Глобальный блокчейн определяет особенности закрепления и цифровых личностей, которые могут взаимодействовать через эвентчейны, блокчейны и приложения.

Заверяющие транзакции могут проводиться практически на любых блокчейнах. Тем не менее, на блокчейнах, оптимизированных для финансовых транзакции или общей логики, заверяющий тип транзакций является дорогостоящим и неэффективным[61]. Кроме того, варианты оптимизации, такая как шардинг и усечение блока, могут негативно влиять на обеспечение целостности релевантной информации[62, 63].

Глобальный блокчейн принадлежит к семейству Nxt[64]. Уникальной особенностью этого блокчейна является то, что транзакции основаны на типах, которые не требуют какой-либо обработки скриптов или обработки входов/выходов транзакций со стороны узлов сети. Это уменьшает размер блокчейна, повышает эффективность и позволяет использовать методы агрегирования, которые особенно важны для заверяющих транзакций.

Вместо того, чтобы использовать Nxt напрямую, мы используем форк платформы WAVES[65] в качестве основы. На этой платформе реализован ряд улучшений, таких как протокол NG (section 15.6), от внедрения которого наша сеть только выиграет. Существующие типы транзакций, ориентированные на цифровые активы, включая «цветные» монеты, удаляются или деактивируются и заменяются заверяющими типами транзакций.

14 Цетрализованный и децентрализованный анкоринг

В других решениях для блокчейн-привязки используется централизованный подход, когда все хеши собираются в течение определенного периода времени. Из этих транзакций создается Merkle tree (дерево хешей), которое помещается в одну транзакцию на стороннем блокчейне, таком как биткойн. В связи с этим отсутствует прямая обратная связь от системы и прежде чем будет получено окончательное подтверждение[66] от центрального узла, может пройти несколько часов.

Глобальный блокчейн LTO является децентрализованным решением, где каждый узел контролирует все транзакции. Как только закрепленная транзакция транслируется, она мгновенно видна, и уже примерно через 3 секунды она предварительно подтверждается с помощью протокола NG (section 15.6). В блоке транзакция появляется в течении минуты.

Узлы могут отслеживать все привязанные хэши или сами узлы. При необходимости они могут создавать подтверждение самостоятельно (section 16.1). То сть, нет необходимости в централизованном обслуживании.

15 Алгоритм консесуса

Глобальный блокчейн работает как обычный публичный блокчейн. Для валидации транзакций и проверки подлинности блока узел сети выбирается в качестве генератора. Для определения генератора мы используем алгоритм консенсуса с выделенным доказательством важности или LPoI. Генератор должен быть вознагражден комиссией за валидацию транзакций в произведенном блоке.

Доказательство важности является вариация алгоритма доказательства доли Proof-of-Stake (PoS), где вероятность быть выбранным для создания блока определяется на основе количества токенов, которые имеет узел сети. С Доказательством важности, вероятность увеличивается на основе частоты использования сети её узлами[67, 68].

Экономика токена, вытекающая из данного алгоритма консенсуса подробно описана в разделе «Экономика токена LTO»[69].

15.1 Лизинг

NXT, Waves и другие блокчейны этого семейства используют LPoS в качестве алгоритма консенсуса[64, 70]. При использовании лизинга токенов, их держатель передает право на создание блока выбранному узлу. Такие узлы могут работать как майнинговый пул, пропорционально распределяя вознаграждения среди тех, кто отдал свои токены в лизинг.

Сети типа NXT восприимчивы к атакам, когда злоумышленник контролирует по крайней мере 1/3 от всех активных балансов[71]. Что является серьезной проблемой, поскольку проекты, которые реализовали алгоритм LPoS склонны к высокому уровню централизации. Два Nxt-узла контролируют более 50 процентов сети [72]. В случае с Waves два узла контролируют более 1/3 сети, а пятерка самых богатых узлов - более 50 процентов[73].

В разделе 18 мы обсудим важность высокой степени децентрализации для нашей сети. Для предотвращения появления узлов, с большой долей в лизинге, существует ограничение на использование токенов отданных ноде в лизинг. Каждому узлу необходимо владеть не менее чем 10процентов токенов от общего количества токенов ноды. Доказательство важности учитывает это, ставя таким образом узлы, которые являются пассивными стейкерами, в менее привелигированное положение.

15.2 Коэффициент вероятности получения возграждения или Raffle factor

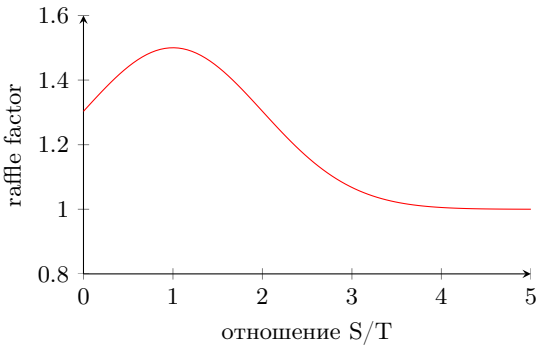
Для того чтобы рассчитать коэффициент использования сети узлом, который увеличивает вероятность создания блока, используется отношение баланса стейкинга к количеству транзакций (отношение S / T).

$$\text{ST ratio} = \frac{\text{Стейкнутые токены в \% от общего количества}}{\text{Сгенерированные транзакции в \% от общего количества}}$$

Отношение S / T связано с «коэффициентом вероятности получения вознаграждения» или раффл-фактором. Раффл-фактор является математической формулой, которая определяет вероятность того, что узел будет выбран для создания нового блока. Чем более сбалансировано отношение S/T (ближе к 1.0), тем выше данный коэффициент. Если отношение S/T разбалансировано (узел не вносит никаких транзакций), раффл-фактор будет равен 1.0.

$$\text{raffle factor } r = 1 + (0.5 \cdot e^{-0.5 \cdot (\text{ST ratio} - 1)^2})$$

Эта формула приводит к стандартному показателю отклонения колоколообразной кривой.

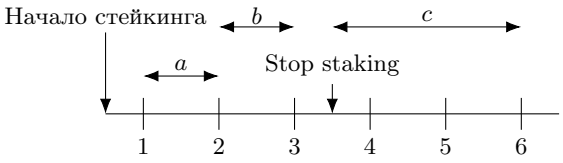


Максимальное значение данного коэффициента составляет 1.5, что является серединой между минимумом и абсолютным максимум - 2.0. Чтобы получить дополнительные X токенов от стейкинга с помощью увеличивая вашей значимости, вам необходимо потратить сумму в размере $2 \cdot X$ от суммы транзакционных комиссий. Важность инфляции подробно объясняется в разделе 18.1.

Чтобы полностью понять концепцию коэффициента вероятности получения вознаграждения (раффл-фактора) и его влияния на экономику токена, пожалуйста, ознакомьтесь с документом «Токеномика LTO»[69].

15.3 Вероятность генерации блока

Вероятность быть выбранным для создания блока равна $P(forge) = S \cdot r$. Количество обработанных транзакций T вычисляются в течении этого времени. При вычислении $P(forge)$, значение S должно быть постоянным за тот же период времени, чтобы предотвратить вероятность злоупотребления.



a = Момент, в который вычисляется P(forge)

b = Момент генерации блока

c = Время, в течение которого токены заблокированы

Рис. 7: Временная шкала стейкинга токенов и создания блоков. Время измерено в количестве суммарных блоков.

15.4 Справедливый PoS

Формула, которая решает, какой узел имеет право создать новый блок основана на алгоритме Fair Proof of Stake[74], созданном WAVES. Это улучшение оригинального Nxt PoS-алгоритма, который в оригинале придаёт слишком большую ценность крупным стейкхолдерам..

Для дальнейшего понимания указанного алгоритма, пожалуйста, прочитайте статью «Fair Proof-of-Stake»[74].

Чтобы преобразовать данный алгоритм из PoS в PoI, формула применяет раффл-фактор к балансу стейкинга, что создает эффективный баланс $b_i \cdot r$.

- T_i время генерации блока для i-го аккаунта,
- X_n подпись генерации,
- r raffle factor,
- b_i доля в процентах стейка от общего количества токенов в стейке,
- Λ_n базовая цель,
- $T_{min} = 5$ постоянная для задержки между блоками,
- $C_1 = 70$, постоянная, определяющая форму распространения задержки,
- $C_2 = 5E17$ постоянная величина для корректировки базовой цели.

$$T_i = T_{min} + C_1 \cdot log(1 - C_2 \cdot \frac{log(X_n / X_{max})}{r \cdot b_i \cdot \Lambda_n})$$

Если вы получите новый блок до того, как отведенное время будет превышено, такой блок должен быть добавлен в цепочку, а затем должно быть рассчитано новое время задержки. Предыдущее T_i (время генерации) становится не актуальным. Каждый узел сам вычисляет T_i, эта информация не предоставляется генератором. Это означает, что нет смысла использовать неправильный баланс стейкинга в расчетах.

15.5 Генерационная подпись

Хэш блока никогда не учитывается во время расчёта времени задержка T_i. Этот хэш основан на содержимом блока, которое определяется узлом, создающим блок. Если бы данный хэш имел значение при определении следующего узла – создателя блока, то стало бы возможным легко манипулировать значением задержки. Узел смогла бы создать несколько разных блоков, но транслировала бы в сеть только тот, который имеет самое низкое значение задержки.

Чтобы предотвратить подобные ситуации, используется только подпись, сформированная при генерации. Эта подпись является вторичным хэш-чейном, т.к. использует только предыдущую генерационную подпись и открытый ключ генератора. В Nxt, такой процесс полностью детерминирован, что делает его восприимчивым к тем, кто хочет извлечь выгоду из данного свойства. [71].

Для борьбы с этим, и уменьшением вероятности появления форков, Fair PoS использует генерационную подпись, использованную 100 блоков назад. Любое изменение баланса на узле или изменения условий лизинга изменяет T_i для заданного X_n. Тем не менее, любая группа, контролирующая не менее 1/3 токенов, имеет 30% преимущество.

PoI требует, чтобы баланс стейкинга и раффл-фактор были постоянными в течение фиксированного периода блоков. Такой подход сделал бы сеть еще более уязвимой для подобных атак.

В качестве решения, генерация не является хешем, который может быть публично рассчитан. Наоборот, вместо этого узел должен захэшировать подпись предыдущего генеротора и подписывать этот хэш своим закрытым ключом. Это и является генерационной подписью. В отличие от Nxt и Waves, узел может вычислять T_i только для себя, что делает невозможным досрочное определение следующего генератора.

15.6 NG-протокол

Протокол NG был представлен как одно из решений проблемы масштабируемости биткойна. Хотя он никогда не был реализован в биткойне. Протокол Waves NG активирован в основной сети Waves с декабря 2017

В протоколе NG генерируются два типа блоков, микроблок и ключевой блок. Узел, который ранее был выбран, может продолжать валидацию транзакций, создавая микроблоки в среднем каждые 3 секунды. Когда выбирается новый узел, он создает ключевой блок из предыдущих обращении микроблоков. Транзакции в микроблоках можно

считать довольно безопасными для транзакции с низким уровнем риска, таких как привязка.

Получение транзакционных сборов разделяется между узлом, который создал микроблок и узлом, который создал ключевой блок с долями 40% - 60%. Это разделение всегда должно быть в пользу создателя ключевого блока. В противном случае был бы стимул для игнорирования уже созданных микроблоков и создания новых микроблоков для себя.

В открытом стресс-тесте Waves NG доказала свою способность обрабатывать до 6000 Tx/мин с пиком 17000 Tx/мин[75]. Потенциально протокол NG может обрабатывать до 1000 Tx/sec или 60 000 Tx/мин. NG-протокол снижает значения задержек и является ключевым компонентом для имплементации новых решений по оптимизации.

16 Типы транзакций

Глобальный блокчейн LTO использует предопределенные типы транзакций. Это позволяет использовать более компактные блоки и устраняет необходимость написания скриптов. Список типов транзакций может быть расширен в будущем в случае необходимости. Типы транзакций, которые возможны в сети в настоящее время, представлены в нижеследующем перечне:

- Привязка (анкоринг): используется для проверки транзакций из приватных блокчейнов,
- Issue a certificate: used to declare relationships between identities,
- Выдача/отклонение сертификатов: используется для оглашения взаимодействий между личностями,
- Трансфер токенов: используется для отправки токенов другим личностям,
- Стейк токенов: используется того, чтобы дать возможность участнику стейкать или отдавать в лизинг токены,
- Отмена стейкинга: используется для остановки стейкинга или лизинга токенов,
- Установка транзакцию сценария: используется для настройки смарт-аккаунтов.

16.1 Привязка (анкоринг)

Анкоринг — это метод принятия хэша документа или других данных и их хранение внутри блокчейн-транзакции. Цель состоит в том, чтобы сделать невозможным для любого, включая создателя, проводить документ задним или будущим числом[76].

Каждое событие приватного эвентчейна привязано к глобальному блокчейну. Приложения третей стороны могут использовать глобальнаый блокчейн для привязки документов с целью доказательства их существования. По нашим оценкам 99% всех транзакций в глобальном блокчейне могут быть транзакциями привязки. Учитывая, что большинство транзакций предназначены для привязки, их агрегация уменьшает количество памяти необходимое для хранения блокчейна.

При создании блока узел генерирует древо хэшей (Merkle tree)[77] из транзакции в порядке, представленном в списке. Только корень древа хэшей добавляется к блокчейну. В рамках процесса валидации, каждый узел воссоздает данное древо хэшей.

Узлы могут индексировать каждый хэш привязки. Однако для уменьшения размера требуемого пространства памяти, большинство узлов должны выбрать для извлечения дерева хэшей путь из собственных привязанных транзакций. Этот путь формирует цифровую квитанцию, которая может храниться с исходными данными (например, такими как событие).

16.2 Аутентификация и авторизация

Методы идентификации, такие как имя пользователя и проверка пароля, требуют централизованной системы. В полностью децентрализованной системе, мы полагаемся на криптографические подписи для обеспечения аутентификации. Хотя информация не передается между эвентчейнами, стороны по-прежнему могут быть идентифицированы по цепям, с учетом пары ключей, используемых для подписи.

В более широком смысле стороны могут подписывать любую информацию таким образом. Это очень похоже на Сертификаты PKI. Опора на центральные органы для выпуска и аннулирования сертификатов препятствует принятию PKI-сертификатов в качестве замены стандартной аутентификации.

При использовании блокчейна пары открытый/закрытый ключ могут быть созданы без потребности в центральном органе. Пара ключей формирует уникальную сущность, на которую можно ссылаться через адрес, полученный из хэша открытого ключа.

16.3 Сертификаты

Транзакция сертификата позволяет каждой личности передавать информацию о другой личности, ссылаясь на ее адрес. В отличие от токенов, предоставление и отзыв учетной записи полностью под контролем стороны, выдающей сертификат.

Сертификату может быть присвоен определенный тип, который выбирается стороной, выдающей сертификат. Хотя это и не требуется, рекомендуется, чтобы сертификат был подтвержден получателем, перед распространением другим участникам сети.

16.4 Доверительная цепочка

Хотя публичный адрес с приватным ключом является методом аутентификации, он не предоставляет решения для авторизации. Сертификаты могут использоваться для определения отношений/взаимодействий между сущностями.

Этот подход похож на сеть доверия (WoT). WoT имеет ряд недостатков по сравнению с PKI, которых нет у нашей платформы.

В блокчейне, установка и отмена отношений или пометка личности как скомпрометированной простая, мгновенная и безотзывная. Блокчейн-транзакции имеют временную метку, что позволяет проверить существование отношений в определенной точке времени.

Вместо простого установления личности мы устанавливаем специфические отношения. Транзакция не подтверждает или не отрицает любую другую информацию о личности, за исключением существования отношений, устраняющих необходимость физической встречи с другой стороной.

В данном контексте нам важно только найти доверительную цепочку между двумя личностями на основе этих отношений. Такой подход имитирует цепочку доверия, как это было сделано с проверкой PKI, но без централизованного влияния. Вместо абсолютного корневого сертификата, блокчейн адрес либо нашей организации, либо организации, с которой мы работаем, становится доверительным корневым узлом.

16.5 Смарт-аккаунты

По умолчанию, любая транзакция для учетной записи должна быть подписана с использованием закрытого ключа, связанного с этой учетной записью. Waves представляет концепцию смарт-аккаунтов, позволяющую любому настраивать эту логику[78].

Для этого логика может быть описана с использованием полного не-Тьюринг языка. Такой скрипт используется только для верификации и отклонения транзакции для конкретной учетной записи. Он не может инициировать другие транзакции. Таким образом, этот тип смарт-контракта не предотвращает агрегацию. Для защиты от таких ситуаций на смарт-аккаунты LTO были наложены дополнительные ограничения.

В LTO не существует транзакций данных, а другие транзакций недоступны из скриптов. Умные учетные записи могут использоваться для создания аккаунта с большим количеством подписей путем указания альтернативных открытых ключей, которые должны быть использованы для подписи транзакций. Вместо указания этих ключей напрямую, смарт-аккаунт указывает, что транзакция может быть подписана кем угодно, кто имеет специальный сертификат.

Существуют и другие ограничения, которые также необходимо учитывать. Учетная запись может быть заблокирована, разрешается только трансфер токенов через определенное количество блоков. При этом разрешается только трансфер определенного количества или трансфер только на определенный аккаунт.

При запуске узла рекомендуется использование мультиподписи. Учетная запись, связанная с узлом, обычно содержит большое количество токенов для стейкинга или выполнения привязки. Приватный ключ к этой учетной записи известен узлу. С использованием мультиподписи, получение этого ключа не даст прямого доступа к токенам учетной записи узла.

Операции привязки не учитываются смарт-аккаунтами. Они всегда должны быть подписаны закрытым ключом учетной записи. Эта логика существует для того, чтобы позволить возможную будущую оптимизацию, например, горизонтальное масштабирование узлов глобального блокчейна.

17 Сводные (summary) блоки

Привязка (анкоринг) - это метод с низким уровнем воздействия и вмешательства, который привносит дополнительный уровень безопасности в блокчейн. Мы ожидаем, что другие приложения, которые не используют живые контракты, также начанут использовать возможности привязки.

Одним из аспектов блокчейна, представляющий проблему для масштабируемости - тот факт, что размер блокчейна будет продолжать расти[79]. Размер предъявляет определенные требования к оборудованию для хранения копий. Что также определяет нагрузку на новые узлы, которые должны будут воспроизвести всю цепь целиком. Чтобы уменьшить скорость роста цепи, используются сводный блоки.

17.1 Размер ключевого блока

В таблице 3 показана структура ключевого блока. Глобальная цепочка должна быть масштабируемой до 50 *millions* транзакций в день. Это примерно в пять раз больше ожидаемого использования. Размер такого ключевого блока определяется данными блока и данными транзакции (5).

$$Keyblock\ size = d + t \tag{5}$$

где:
d = данные блока,
t = транзакционные данные.

Перед вычислением ожидаемого размера блока следует обозначить следующее допущения:

- 99.98% транзакций - это анкоринговые (table 5). Это основное предназначение глобального блокчейна,
- Оставшиеся 0.02% - это транзакции с сертификатами (table 7),
- Все другие транзакции являются редкими и ими можно пренебречь,
- Все транзакции распределены равномерно по блокам,
- В среднем генерируется один ключевой блок в минуту,
- Каждый день создается около 1440 ключевых блоков.

Данные блока составляют 277 байтов (table 3). В соответствии с ранее сделанными предположениями, размер данных транзакции может быть рассчитан с использованием уравнения (6).

$$Transaction\ data\ size = n \cdot (0.9998 \cdot a + 0.0002 \cdot c) \tag{6}$$

with:
a = размер транзакции привязки (table 5),
c = Размер транзакции сертификата (table 7),
n = количество транзакций на блок.

Это все приводит к общему размеру блока в 3.8MB.

17.2 Рост без агрегации

С 3.8MB на блок и 1440 блоками в день, блокчейн будет расти на 5.47GB в день и 2TB в год, если он будет работать непрерывно на полную мощность.

Ожидаемый объем составляет около 10 *million* транзакций, что увеличивает блокчейн на 1.1GB в день. В конечном результате это дает 3.65 *billion* транзакций или 400GB в год.

Для биткоина с его 340 *million* транзакций в общей сложности[80], требуется около семи дней для синхронизации от момента создания, в зависимости от скорости сети и оборудования.

С миллиардами транзакций глобального блокчейна получим ожидание синхронизации неделями или даже месяцами.

Одной из целей агрегирования транзакций является требование только 20 минут синхронизации в год, но опять же, в зависимости от скорости сети и оборудования. С 365 сводный блоками в год, узел должен иметь возможность обрабатывать сводный блок за 3 секунды.

17.3 Протокол Segregated witness

Segregated witness - это стратегия, используемая в биткойне для сокращения размера данных в блоке[81] путем разделения транзакций на две части: данные, которые необходимо обработать, и данные, которые используются для проверки транзакции. Эта вторая часть называется witness-данными, которая среди прочего содержит подписи.

Финализация — это гарантия того, что блоки, которые находятся достаточно глубоко в цепи, никогда не будут

удалены из блокчейна. Невзирая на вероятностную финализацию или протокол финализации, witness-данные не имеют смысла, если блок гарантированно невозможно отменить. Узлы могут бесплатно удалять данные свидетеля для блоков, которые прошли стадию финализации для уменьшения размера блокчейна.

Мы встроили логику segregated witness в концепцию сводных (summary) блоков.

17.4 Агрегация

Сводный блоки – это специальные блоки, которые создаются каждые 1440 блоков (примерно один раз в день). Они содержат агрегированные значения всех блоков начиная с предыдущего сводный блока. При воспроизведении цепи, необходимо использовать только сводный блоки, чтобы достигнуть текущего состояния. Тогда только блоки, которые были созданы после последнего сводный блока, необходимо воспроизвести для достижения полной синхронизации с сетью. Это значительно уменьшает время воспроизведения.

Второй и последний сводный блок, и все блоки до него являются окончательными. Узлы не будут рассматривать форки сети до этого момента, независимо от самой длинной цепи. Это означает, что только транзакции от предыдущих 1440 до 2880 ключевых блоков должны быть сохранены. Удаление данных транзакций из ключевых блоков позволяет уменьшить их размер с 11.3MB to 277 байт, что делает их объем пренебрежимо малым в сравнении с summary-блоками

17.5 Отличие от простого уменьшения размера блокчейна

На первый взгляд, этот подход выглядит похожим на уменьшение размера блокчейна, поскольку поддерживается только ограниченный набор транзакций. Опасность с уменьшением размеров заключается в том, что когда вводится фальсифицированное состояние, это угрожает основополагающим законам блокчейна.

Опасность связана с передачей по сети текущего состояния блокчейна. С использованием протокола segregated witness, опираясь на концепцию финализации, транзакции принимаются и без проверки подлинности подписи. Однако каждый узел все равно должен принять все транзакции, начиная с генезис-блока, для расчета текущего состояния. Фактические данные транзакции не используются для расчета подписи блока, но скорее сохраняются как приложение для следующего блока (table 2). В качестве событий без транзакций ключевые блоки являются частью блокчейна и не могут быть проигнорированы. Если транзакции принимаются без валидации, их объединение несет лишь небольшой риск.

17.6 Размер сводных блоков

Сводный блоки содержат всю информацию о не привязанных транзакциях и агрегированную версию всех сборов транзакционных сборов и движениях других токенов. Это довольно большие блоки, особенно в сравнении с ключевыми блоками. Чтобы уменьшить объем используемой памяти, информация о транзакционных сборах и передаче токенов уменьшены до изменения баланса у участника (table (4)). Сводный также содержит не агрегируемые транзакции, такие как транзакции выдачи/получения сертификатов, стейкинга и выполнения скриптов. Для вычисления примерного ожидаемого размера сводного блока допустим следующие предположения:

- В общей сложности ожидается 200000 участников,
- Каждый день создается один сводный (сводный) блок,
- Исходя из предположений в предыдущем разделе, можно полагать, что сводная информация изменений баланса будет являться единственной значительной частью сводного блока.

Используя эти предположения, может быть рассчитан размер сводного блока. При использовании уравнения 7 для вычисления размера сводного блока, итог составит около 10.3MB.

$$Summary\ block\ size = Transaction\ summary = p \cdot e \quad (7)$$

где:
p = Количество участников в последних 1500 блоках,
e = Размер сводной записи изменений балансов (table 4.

17.7 Общий размер

Общий размер блокчейна состоит из двух частей: статической и растущей. Статическая часть состоит из последней тысячи ключевых блоков. Они в любом случае будут содержать прикладные служебные данные (5).

$$Static\ part = n \cdot k \quad (8)$$

где:
n = Количество ключевых блоков, хранящихся в данных транзакции,
k = Размер ключевого блока (5).

При использовании уравнения 8 для вычисления величины вероятности, получается, что общий размер блокчейна составляет около 11.3GB. Поскольку только последние 1000 блоков хранятся полностью, включая транзакции, это значение может немного отличаться, но не будет заметно выше.

Растущая часть состоит из сводных блоков (7) и того, что осталось от ключевых блоков после того, как данные о транзакциях удалены. Для определения роста размера данных за год мы будем использовать уравнение (9).

$$Growing\ part = n \cdot k + m \cdot s \quad (9)$$

with:
n = Количество ключевых блоков за год,
m = Количество сводных блоков за год,
k = Размер ключевого блок без транзакционных данных,
s = Размер сводного блока (7).

Следуя ранее сделанным предположениям, согласно этому уровнению, можно увидеть, что блокчейн будет расти в среднем на 3.7GB в год.

17.8 Исторические ноды

Для удаления старых транзакций не требуются ноды. Поддерживая все транзакции, History-ноды могут подтвердить корректность блокчейна, при необходимости. Исторические ноды не смогут пройти проверку при использовании фальсифицированных данных, так как необходимо, чтобы блоки History-ноды соответствовали блокам других узлов. Сеть может полагаться на относительно небольшое количество узлов истории.

Запуск узла истории не дает каких-либо преимуществ и не увеличивает вероятность создания новых блоков. Его запуск может быть произведен только в интересах сообщества, или как источник пассивного дохода.

18 Риски уязвимости сети

18.1 Важность инфляции

Одно из основных сомнений в отношении алгоритма PoI лежит в возможности намеренноого увеличения важности из-за внедрения в сеть фиктивных транзакций. Мы можем рассчитать прибыль/убыток от спам-транзакций исходя из формулы максимального значения Raffle factor, приведенной ниже:

- Raffle factor; r,
- Процент стейкнутых токенов; b_i,
- СТоимость транзакции; c,
- Общее количество транзакций в сети; n,
- Спам-транзакции; τ,
- Вознаграждение; p,
- Прибыль/убыток от спама; $\Delta p = p_{r_{max}} - p_{r=1}$.

$$p = (r \cdot b_i \cdot n \cdot c) - (\tau \cdot c) \tag{10}$$

$$r = 1, \tau = 0 \rightarrow p = b_i \cdot n \cdot c \tag{11}$$

$$r = r_{max}, \tau = b_i \cdot n \rightarrow (r_{max} - 1) \cdot b_i \cdot n \cdot c \tag{12}$$

Что дает

$$\Delta p = ((r_{max} - 2) \cdot b_i \cdot n \cdot c) \tag{13}$$

При заданных

$$b_i > 0, n > 0, c > 0, \Delta p < 0 \rightarrow (r_{max} - 2) < 0 \tag{14}$$

$$r_{max} < 2 \tag{15}$$

Исходя из этих уравнений, видно, что получить прямую выгоду от спам-транзакций невозможно, поскольку Raffle factor близок к значению 2.0. В этом случае транзакции становятся почти бесплатными. Повышение приоритета в сети в этом случае вызовет риск атаки 51%, потому что ее будет дешевле осуществить. Поэтому максимальное значение Raffle Factor, установленное на значении 1,5, обеспечивает защиту – попытка разгона инфляции для удешевления атаки становится слишком затратной.

18.2 Атака Nothing at stake

Принцип «Nothing at stake» предполагает, что узлы скорее будут продолжать работу в любых форк-цепях, нежели продолжат самую длинную цепь, поскольку это не несет никаких негативных последствий[82]. Если все узлы будут вести себя именно так, то атакующему потребуется небольшой процент токенов, чтобы заставить сеть переключиться на другую цепь; это атака 1%.

Такая ситуация называется «Трагедией общин» (Tragedy Of The Commons)[83]. Все стороны стремятся получить индивидуальную прибыль от недостатков системы. Но, если все поступают подобным образом, то в итоге никто не получит от этого прибыли. Наоборот, такие действия приводят к доскредитации сети, о соответственно – и к уменьшению ценности основного токена этой сети.

Алгоритм справедливого PoS от Waves сильно усложнил возможность доминирования побочной цепи над материнской[74]. Вероятность получения прибыли от такого недобросовестного поведения участниками сети, удерживающими сравнительно небольшую часть токенов – маловероятна.

Злоумышленнику элементарно нужно будет помимо создания еще и поддерживать измененные версии нод. То есть, это означает огромные расходы в сочетании с маленькой веротяностью получения выгоды. И этих факторов хватает для предотвращения атак подобного типа[84].

18.3 Централизация LPoS

Проекты, реализовавшие алгоритм LPoS, имеют тенденцию к гиперцентрализации. Этот эффект можно объяснить вознаграждением за токены. Грамотно собранное и настроенное железо с почти 100% временем безотказной работы не пропустит возможность создания блока, принося больше вознаграждений за установленное количество токенов в стейкинге. Это привлекает держателей токенов к лизингу токенов на таких узлах и усиливает эффект, поскольку уменьшает накладные расходы и позволяет получать более высокие выплаты узлам, которым отдаются средства в лизинг.

Ограничение количества токенов на узел является ошибочным методом, т.к. создает возможность для Sybil-атак[85]. В системе репутации без необходимости регистрации в сети, один узел может замаскироваться под несколько, используя ложные идентификаторы для обхода данного ограничения. Либо злоумышленник может просто запустить много узлов.

Из-за характера нашего решения большинство транзакций будет подписано парой ключей, связанной с узлом. Сеть также будет состоять из относительно большого числа нод. Это не влияет на PoS-сети; однако в PoI дает пользователям платформы преимущество перед простыми держателями токенов (подробнее об этом – в Token Economy Paper) Дополнительной мерой защиты является ограничение использования токенов, полученных в лизинг; каждому узлу необходимо иметь не менее чем 10% токенов от количества, находящегося у него в лизинге.

18.4 DDoS-атаки

Из-за ограниченной масштабируемости любой публичный блокчейн довольно легко перегрузить большим количеством транзакций. Транзакционные сборы обеспечивают первичную защиту от таких видов атак, но транзакции все еще необходимо проверять, чтобы сделать вывод о недостаточности средств. Кроме того, можно уронить сеть за счет спам-транзакций при наличии достаточного объема ликвидности, как это было с сетью Ethereum в июле 2018 года.

Узлы имеют возможность автоматически увеличивать транзакционные сборы в случае угрозы спам-атаки. В идеале, узлы получают от стейкинга столько же, сколько они тратят на транзакции. Когда вознаграждение за транзакционные сборы растет за счет спам-токенов, то автоматически срабатывает противодействие подобным атакам.

18.5 Уязвимость SHA-2

В 2017 году SHA-1 подверглась уязвимости, когда в исследовательской лаборатории Google удалось найти коллизию, при которой из-за уязвимости в хэш-алгоритме получилось создать 2 PDF-документа с одинаковыми хэшами, но разным содержимым[86]. Если алгоритм SHA-2 256bit так же уязвим, то это может иметь плачевные последствия для анкоринга, основанного на дереве хэшей.

Если обнаружена коллизия, можно утверждать, что документ с коллизией был заверен. Хуже того, учитывая корень древа хэшей и рандомный хэш, можно было бы создать

целое действительное дерево хэшей. Это может позволить хакеру проходить проверку любых документов. Однако это все равно будет непростой задачей, поскольку каждая ветвь дерева должна сочетаться с двумя хэшами, которые вместе составляют ровно 32 байта.

В то время, как для принудительной работы конкретного SHA-1 все равно потребуется слишком много вычислений в рамках временного цикла, то «парадокс дней рождения» приводит к значительно меньшему количеству вычислений, требуемых для поиска коллизии. «Парадокс дней рождения» также применим и к публичному блокчейну LTO, так как существует множество корней дерева хэшей, каждый из которых имеет максимальное количество путей дерева хэшей.

Для предотвращения этого в случае возникновения спорных ситуаций, проверяющая сторона может обратиться к историческим нодам. Однако из-за этого меньше ресурсов нод уходит на анкоринг.

Вместо этого можно добавить вторичное дерево хэшей, где SHA-2 хэш хэшируется другим алгоритмом, таким как SHA-3 или Blake2. Простое двойное хеширование не слишком полезно там, где можно фальсифицировать дерево хэшей, поскольку для эксплоита хватит уязвимости во внешнем алгоритме. Если же присутствует два древа хэшей, то подобрать уязвимости для обоих алгоритмов теоретически почти невозможно. Однако даже в этом случае есть риск нахождения коллизии для обоих деревьев одного блокчейна – как раз из-за «Парадокса дней рождения».

Part III. Платформа

19 Архитектура

19.1 Микро-архитектура

Ноды LTO разработаны с использованием архитектуры микросервисов. Это означает, что все функциональные возможности узла разделены на микросервисы, причем каждая служба ответственна только за небольшую часть всего узла. Есть несколько преимуществ использования данного шаблона проектирования, которые перечислены ниже:

- Отказоустойчивость: если какая-то служба откажет, то не обязательно заденет или помешает другим службам.
- Масштабируемость: все сервисы внутри узла распараллелены и, следовательно, могут работать на разных машинах. Это делает механизм действительно удобным для горизонтального масштабирования. Масштабирование автоматизировано, подробнее - в разделе 24
- Гибкость: некоторые функции выполняются лучше, если прописаны на определенных языках программирования. Каждый сервис может быть разработан на необходимом ему языке программирования.
- Качество кода: разделяя узел на маленькие надежные модули, разработчикам становится проще читать и просматривать его код. Это приводит к улучшению качества кода.

Микросервисы сгруппированы в контейнер Docker. Такие контейнеры запускаются с использованием платформы управления контейнерами Kubernetes; это описано в разделе 24. Каждый микросервис предназначен для автономной работы. Это означает, что он не имеет общих зависимостей, поэтому каждый контейнер имеет свои собственные базы данных или очереди событий. Микросервисы также предназначены для работы без учета состояния сети для того, чтобы быть легко масштабируемыми.

19.2 Уровни приложений и службы

Как было описано в разделе 19.1 узел разбивается на несколько сервисов. Эти службы сгруппированы в четыре разных уровня:

- UI слой: состоит из приложений пользовательского интерфейса, которые взаимодействуют с прикладным уровнем (уровнем приложений)
- Уровень приложений: содержит все службы, которые обрабатывают действия, вызванные событиями из эвентчейна,
- Слой приватной цепи: отвечает за распараллеливание работы узла,
- Открытый сетевой уровень или уровень публичной цепи: управляет службой общедоступных публичных блокчейнов. Наш глобальный публичный блокчейн оптимизирован для хранения хэшей. Каждый узел индексирует все хэши, благодаря чему они легко верфицируемы.

20 UI-уровень (пользовательский интерфейс)

Уровень пользовательского интерфейса содержит два интерфейса, которые позволяют пользователям легко разрабатывать и редактировать live-контракты. Первый является средством просмотра блокчейнов, которое позволяет пользователям подключаться к определенному узлу и просматривать все цепочки. Пользователь может просматривать только цепи, частью которых он является. Второй интерфейс - приложение интерактивной среды, которое позволяет пользователю разрабатывать сценарии для live- контрактов. Оно визуализирует сценарий на диаграмме состояний, а также содержит иные инструменты визуализации и проверки.

21 Уровень приложений

21.1 Веб-сервер

Приложение веб-сервера служит своеобразной прослойкой между интерфейсом и приложениями внутри узла. Веб-сервер выполняет две функции:

- Аутентификация всех запросов к сервисам
- Адресация запросов к конкретным сервисам.

21.2 Модуль управления рабочими процессами

Фактически создание и исполнение live-контрактов выполняется модулем управления бизнес- процессами. Если событие, полученное сервисом эвентчейна содержит действие в live-контракте, то оно будет направлено в модуль управления бизнес-процессом. Затем служба бизнес-процесса выполнит действие, которое приведет к переходу процесса в новое состояние и созданию новой проекции бизнес-процесса. Эта проекция сохраняется в базе данных MongoDB.

22 Слой приватного блокчейна

Слой приватного блокчейна рапараллеливает работу узла. Распараллеливание обеспечивает стабильность системы, даже в случаях плохой связи или высокой нагрузки. Очередь сообщений представляет собой уровень связи для приватной цепи. Технология, предоставляющая очередь сообщений называется RabbitMQ. Она представляет собой легким диспетчером сообщений, который идеально подходит для доставки сообщений не только внутри узла, но и другим узлам. RabbitMQ имеет функцию под названием «Shovel», которая динамически устанавливает соединение с другим RabbitMQ диспетчерами и обменивается сообщениями. Этот механизм используется для передачи событий от одного узла другому.

Следующие три службы, о которых написано ниже, управляют всеми входящими и исходящими событиями.

22.1 Служба эвентчейна

Служба, управляющая приватными цепочками — это сервис эвентчейна. Данная служба обрабатывает все входящие события. При обработке этих событий выполняются следующие шаги:

- Валидируются входящие события с помощью проверки корректности подписей и проверки корректной работы цепи.
- Проверяется, соответствует ли цепочка той, что хранится локально на узле. Если нет, сервис, по мере возможности, выполняет разрешение конфликтов.
- Если событие(я) отправляются личностью, принадлежащей этому узлу, он выполнит принятое событие(я). В противном случае будет только хранить их в базе данных

- Если новая личность добавляется из другого узла, вся цепочка перенаправляется на этот узел
- Все новые события передаются связанным узлам.

Для хранения в службе эвентчейнов используется база данных MongoDB.

22.2 Служба очереди событий

У сервиса очереди событий небольшая задача расставлять события в очереди событий. Он делает это как для служб внутри узла (например, для сервиса управления бизнес-процессом или для сервиса эвентчейна), так и для внешних пользователей.

22.3 Служба доставки событий

Все сообщения в очереди событий обрабатываются сервисом доставки событий. Он собирает все сообщения и направляет их сервису событий. Если сервис событий обрабатывает сообщение, оно будет помечено как обработанное; в противном случае оно будет перемещено в очередь зависших сообщений.

23 Уровень публичного блокчейна

23.1 Служба анкоринга

Сервис анкоринга (закрепления) – это сердце уровня публичного блокчейна. Сервис анкоринга будет форком сети платформы NXT с использованием протокола NG. Также он будет расширен для возможности обработки как «обычных» транзакций, так и дата-транзакций.

Эти дата-транзакции (транзы с данными) будут использоваться для хранения хэшей событий приватных чейнов, как описано в разделе 16.1. Все эти хэши будут собираться ежедневно и детерминированно объединяться в древо хэшей. Таким образом, дата-транзы можно будет удалять из хранилища, чтобы уменьшить объем, но люди все еще смогут проверять существование хэшей событий.

Чтобы была возможность определить, тот ли определенный хэш сохранился, все хеши будут индексироваться. Это сделает процесс верификации более быстрым, потому что не будет необходимости производить поиск по всем данным о транзакциях.

24 Управление контейнерами

Поскольку узел содержит несколько микросервисов, для управления контейнерами требуется специальная платформа. Платформы управления контейнерами выполняют такие задачи как резирвирование хостов, создание экземпляров контейнеров, перезапуск контейнеров, в которых произошел сбой, и масштабирование кластеров путем добавления или удаления контейнеров. Различные инструменты управления контейнерами могут использоваться для этих целей. Например Docker Swarm, Mesos, Nomad или Kubernetes. Первоначально файл конфигурации будет создан на платформе Kubernetes. Каждая служба будет настроена на работу со своим собственным Pod и с собственным менеджером нагрузки. Это делается для того, чтобы отдельные сервисы могли масштабироваться независимо друг от друга. Pod автоматического горизонтального масштабирования используется для управления масштабированием сервисов [87].

Список литературы

[1] Hannah Ritchie Max Roser. Technological Progress. https://www.ft.com/content/cb56d86c-88d6-11e7-afd2-74b8ecd34d3b. Accessed: 03-06-2018. 2017.

[2] Christine Legner и Kristin Wende. “The challenges of inter-organizational business process design – a research agenda”. в: (2007).

[3] Benjamin E. Hermalin и Michael L. Katz. “Moral Hazard and Verifiability: The Effects of Renegotiation in Agency”. в: (1990).

[4] Audun Jøsang. “The right type of trust for distributed systems”. в: Proceedings of the 1996 workshop on New security paradigms. ACM. 1996, с. 119—131.

[5] Israel Z Ben-Shaul и Gail E Kaiser. “A paradigm for decentralized process modeling and its realization in the oz environment”. в: Proceedings of the 16th international conference on Software engineering. IEEE Computer Society Press. 1994, с. 179—188.

[6] Ray Fisman и Roberta Gatti. “Bargaining for bribes: The role of institutions”. в: International handbook on the economics of corruption (2006), с. 127—139.

[7] Jörg Becker, Michael Rosemann и Christoph Von Uthmann. “Guidelines of business process modeling”. в: Business Process Management. Springer, 2000, с. 30—49.

[8] Manfred Reichert, Thomas Bauer и Peter Dadam. “Enterprise-wide and cross-enterprise workflow management: Challenges and research issues for adaptive workflows”. в: (1999).

[9] Vitalik Buterin. “Ethereum White Paper: A Next-Generation Smart Contract and Decentralized Application Platform”. в: (2013).

[10] Vitalik Buterin и Karthik Gollapudi. A Next-Generation Smart Contract and Decentralized Application Platform. https://github.com/ethereum/wiki/wiki/White-Paper/f18902f4e7fb21dc92b37e8a0963eec4b3f4793a. Accessed: 22-05-2018.

[11] Ian Grigg. “The Ricardian Contract”. в: (2004).

[12] Nick Sabo. “Formalizing and Securing Relationships on Public Networks”. в: (1997).

[13] Telser. “A Theory of Self-Enforcing Agreements”. в: (1980).

[14] David Joulfaian Douglas Holtz-Eakin и Harvey S. Rosen. “Sticking it out: entrepreneurial survival and liquidity constraints”. в: (1993).

[15] Toshi wallet now supports ERC20 tokens and ERC721 collectibles. https://blog.toshi.org/toshi-wallet-now-supports-erc20-tokens-and-erc721-collectibles-e718775895aa. Accessed: 30-08-2018.

[16] ERC-20 Token Standard. https://eips.ethereum.org/EIPS/eip-20. Accessed: 30-08-2018.

[17] Daniel IA Cohen и Daniel IA Cohen. Introduction to computer theory. т. 2. Wiley New York, 1991.

[18] Marko Vukolić. “Rethinking permissioned blockchains”. в: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts. ACM. 2017, с. 3—7.

[19] AbdulSalam Kalaji, Rob Mark Hierons и Stephen Swift. “A search-based approach for automatic test generation from extended finite state machine (EFSM)”. в: Testing: Academic and Industrial Conference-Practice and Research Techniques, 2009. TAIC PART’09. IEEE. 2009, с. 131—132.

[20] M.G. Gouda, E.G. Manning и Y.T. Yu. “On the Progress of Communication between Two Finite State Machines”. в: (1984).

[21] G Pace и J Schapachnik. “Contracts for Interacting Two-Party Systems”. в: (2012).

[22] Mark D. Flood и Oliver R Goodenough. “Contract as Automaton: The Computational Representation of Financial Agreements”. в: (2015).

[23] Davide Basile, Pierpaolo Degano и Gian-Luigi Ferrari. “Automata for Service Contracts”. в: (2014).

[24] Pierpaolo Degano Davide Basile и Gian-Luigi Ferrari. “From Orchestration to Choreography through Contract Automata”. в: (2014).

[25] Ian Ayres и Robert Gertner. “Filling Gaps in Incomplete Contracts: An Economic Theory of Default Rules”. в: (1989).

[26] Jan L.G. Dietz. “Understanding and Modeling Business Processes with DEMO”. в: (1999).

[27] YoungJoon Byun, Beverly A. Sanders и Chang-Sup Keum. “Design Patterns of Communicating Extended Finite State Machines in SDL”. в: (2001).

[28] Petri. “Kommunikation mit Automaten”. в: (1962).

[29] Dennis Kafura. Notes on Petri Nets. http://people.cs.vt.edu/kafura/ComputationalThinking/Class-Notes/Petri-Net-Notes-Expanded.pdf. Accessed: 01-09-2018.

[30] Wil M.P. van der Aalst. “The Application of Petri Nets to Workflow Management”. в: (1998).

[31] Jan Recker и др. “How good is bpmn really? Insights from theory and practice”. в: (2006).

[32] Wil M.P. van der Aalst и др. “Life After BPEL?” в: (2005).

[33] Luciano García-Bañuelos и др. “Optimized Execution of Business Processes on Blockchain”. в: (2017).

[34] Jan L.G. Dietz. “DEMO: Towards a discipline of organisation engineering”. в: (1999).

[35] JSONForms - React. https://jsonforms.io/. Accessed: 30-08-2018.

[36] JSONForm - Bootstrap 3. https://github.com/jsonform/jsonform. Accessed: 30-08-2018.

[37] Mozilla react-jsonschema-form. https://github.com/mozilla-services/react-jsonschema-form. Accessed: 30-08-2018.

[38] Angular Schema Form. http://schemaform.io/. Accessed: 30-08-2018.

[39] Open Document Format. http://www.opendocumentformat.org/. Accessed: 30-08-2018.

[40] Sindhu Sajana и Sethumadhavan. “On Blockchain Applications: Hyperledger Fabric And Ethereum”. в: (2018).

[41] Stephen A Cook. “The complexity of theorem-proving procedures”. в: Proceedings of the third annual ACM symposium on Theory of computing. ACM. 1971, с. 151—158.

[42] Daniel Brand и Pitro Zafiropulo. “On communicating finite-state machines”. в: Journal of the ACM (JACM) 30.2 (1983), с. 323—342.

[43] Robert M Hierons AbdulSalam Kalaji и Stephen Swift. “New approaches for passive testing using an Extended Finite State Machine specification”. в: (2003).

[44] O Sury и R Edmonds. Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC. тех. отч. 2017.

[45] NIST. Transition Plans for Key Establishment Schemes using Public Key Cryptography. https://csrc.nist.gov/News/2017/Transition-Plans-for-Key-Establishment-Schemes. Accessed: 13-07-2018. 2017.

[46] Daniel J. Bernstein и др. “High-speed high-security signatures”. в: Journal of Cryptographic Engineering 2.2 (2012), с. 77—89. ISSN: 2190-8516. DOI: 10.1007/s13389-

012-0027-1. URL: https://doi.org/10.1007/s13389-012-0027-1.

[47] Henri Gilbert и Helena Handschuh. “Security analysis of SHA-256 and sisters”. в: International workshop on selected areas in cryptography. Springer. 2003, с. 175—193.

[48] NIST. NIST Policy on Hash Functions. https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions. Accessed: 13-07-2018. 2015.

[49] Bruce Schneier и John Kelsey. “Cryptographic Support for Secure Logs on Untrusted Machines.” в: USENIX Security Symposium. т. 98. 1998, с. 53—62.

[50] Peter Bailis и Ali Ghodsi. “Eventual consistency today: Limitations, extensions, and beyond”. в: Queue 11.3 (2013), с. 20.

[51] Andrew S Tanenbaum и Maarten Van Steen. Distributed systems: principles and paradigms. Prentice-Hall, 2007.

[52] Miguel Castro и Barbara Liskov. Byzantine fault tolerance. US Patent 6,671,821. 2003.

[53] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf. Accessed: 17-05-2018.

[54] Aggelos Kiayias и др. “Ouroboros: A provably secure proof-of-stake blockchain protocol”. в: Annual International Cryptology Conference. Springer. 2017, с. 357—388.

[55] POA Network. Proof of Authority: consensus model with Identity at Stake. https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256. Accessed: 17-05-2018.

[56] Git Documentation. Git Branching - Rebasing. https://git-scm.com/book/en/v2/Git-Branching-Rebasing. Accessed: 09-08-2018.

[57] Aaron van Wirdum. “Rejecting Today’s Hard Fork, the Ethereum Classic Project Continues on the Original Chain: Here’s Why”. в: Bitcoin Magazine 20 (2016).

[58] European Parliament. REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL: on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN. Accessed: 12-07-2018. 2016.

[59] Olly Jackson. “Is it possible to comply with GDPR using blockchain?” в: International Financial Law Review (2018).

[60] S Goldwasser, S Micali и C Rackoff. “The knowledge complexity of interactive proof systems”. в: (1989).

[61] Blockchain costs per transaction. https://www.blockchain.com/charts/cost-per-transaction. Accessed: 05-09-2018.

[62] Emanuel Palm. Implications and Impact of Blockchain Transaction Pruning. 2017.

[63] Wenting Li и др. “Towards scalable and private industrial blockchains”. в: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts. ACM. 2017, с. 9—14.

[64] Serguei Popov. “A probabilistic analysis of the nxt forging algorithm”. в: Ledger 1 (2016), с. 69—83.

[65] Waves platform. WAVES whitepaper. https://blog.wavesplatform.com/waves-whitepaper-164dd6ca6a23. Accessed: 16-07-2018. 2016.

[66] Chainpoint Node API: How to Create a Chainpoint Proof. https://github.com/chainpoint/chainpoint-node/wiki/Chainpoint-Node-API:-How-to-Create-a-Chainpoint-Proof. Accessed: 05-09-2018.

[67] Gleb Kostarev. Review of blockchain consensus mechanisms. https://blog.wavesplatform.com/review-of-blockchain-consensus-mechanisms-f575afae38f2. Accessed: 13-07-2018. 2017.

[68] NEM. NEM technical reference. https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf. Accessed: 13-07-2018. 2018.

[69] LTO. “LTO Token Economy”. в: (2018).

[70] Waves Platform. Blockchain Leasing For Proof Of Stake. https://blog.wavesplatform.com/blockchain-leasing-for-proof-of-stake-bac5335de049. Accessed: 13-07-2018. 2018.

[71] mthcl. “The math of Nxt forging”. в: (2014).

[72] Waves generators. http://dev.pywaves.org/generators/. Accessed: 28-08-2018.

[73] Nxt Blockchain Explorer. https://nxtportal.org/monitor/. Accessed: 28-08-2018.

[74] Kofman Begicheva. “Fair Proof of Stake”. в: (2018).

[75] Waves-NG stress test: results in! https://blog.wavesplatform.com/waves-ng-stress-test-results-in-44090f59bb15. Accessed: 05-09-2018.

[76] S. Haber и W.S. J Stornetta. “How to time-stamp a digital document”. в: (1991).

[77] Ralph C. Merkle. “Method of providing digital signatures”. пат. США 4309569. 5 янв. 1982.

[78] A. Begicheva и I. Smagin. “RIDE: a Smart Contract Language for Waves”. пат. 2018.

[79] Saifedean Ammous. “Blockchain Technology: What is it good for?” в: (2016).

[80] Blockchain number of transaction. https://www.blockchain.com/en/charts/n-transactions-total. Accessed: 05-09-2018.

[81] BIP 141: Segregated Witness (Consensus layer). https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki. Accessed: 05-09-2018.

[82] Problems Ethereum. https://github.com/ethereum/wiki/wiki/Problems. Accessed: 30-08-2018.

[83] G Hardin. “The Tragedy of the Common”. в: (1969).

[84] Nothing considered a look at nothing at stake vulnerability for cryptocurrencies. https://pivx.org/nothing-considered-a-look-at-nothing-at-stake-vulnerability-for-cryptocurrencies/. Accessed: 30-08-2018.

[85] John R Douceur. “The sybil attack”. в: International workshop on peer-to-peer systems. Springer. 2002, с. 251—260.

[86] Marc Stevens и др. “The first collision for full SHA-1”. в: (2017).

[87] Kubernetes. Kubernetes, Horizontal Pod Autoscaling. https://github.com/kubernetes/community/blob/master/contributors/design-proposals/autoscaling/horizontal-pod-autoscaler.md. Accessed: 03-08-2018.

#	Название поля	Размер
1	Версия	1
2	Временная метка	8
3	Подпись родительского блока	64
4	Длина суммирующего блока	4
5	Базовая цель	8
6	Подпись генератора	32
7	Список хэшей транзакций	32
8	древо хэшей анкоринга	32
9	Генерация публичных ключей	32
10	Подпись блока	64

Таблица 1: Структура ключевого блока

#	Название поля	Размер
1	Количество транзакций (x)	4
2	Транзакция #1 байтов	113
...	...	...
2 + (K - 1)	Транзакция #K байтов	113

Таблица 2: Составляющие ключевого блока

#	Название поля	Размер
1	Версия	1
2	Временная метка	8
3	Подпись родительского блока	64
4	Длина суммирующего блока	4
5	Базовая цель	8
6	Подпись генератора	32
7	Список хэшей транзакций	32
8	Транзакция #1 байтов	TODO
...	...	...
$8 + (K - 1)$	Транзакция #K байтов	TODO
$9 + (K - 1)$	Сводная запись изменений балансов #1	40 (Table 4)
...	...	...
$9 + (K - 1) + (N - 1)$	Сводка изменений балансов #N	40 (Table 4)
$10 + (K - 1) + (N - 1)$	Публичный ключ генератора	32
$11 + (K - 1) + (N - 1)$	Подпись блока	64

Таблица 3: Структура сводного (summary) блока

#	Название поля	Размер
2	Адрес кошелька	32
3	Изменение баланса	8

Таблица 4: Сводка баланса

#	Название поля	Размер
1	Тип транзакции	1
2	Хэш анкоринга	32
3	Плата	8
4	Временная метка	8
5	Подпись	64

Таблица 5: Структура анкоринговых транзакций

#	Название поля	Размер
1	Тип транзакции	1
2	Временная метка	32
3	Адрес получателя	32
4	Сумма	8
5	Плата	8
6	Временная метка	8
7	Подпись	64

Таблица 6: Структура транзакции перевода

#	Название поля	Размер
1	Тип транзакции	1
2	Id	32
3	Адрес отправителя	32
4	Адрес получателя	32
5	Дата окончания срока	8
6	Тип сертификата	32
7	Плата	8
8	Временная метка	8
9	Подпись	64

Таблица 7: Структура транзакции выпуска сертификата

#	Название поля	Размер
1	Тип транзакции	1
2	Id	32
3	Новая дата истечения срока действия	8
4	Плата	8
5	Временная метка	8
6	Подпись	64

Таблица 8: Структура транзакции обновления сертификата

#	Название поля	Размер
1	Тип транзакции	1
2	Адрес отправителя	32
3	Адрес получателя	32
4	Сумма	8
5	Плата	8
6	Временная метка	8
7	Подпись	64

Таблица 9: Структура транзакции лизинга

#	Название поля	Размер
1	Тип транзакции	1
2	Адрес отправителя	32
3	Адрес получателя	32
4	Сумма	8
5	Плата	8
6	Временная метка	8
7	Подпись	64

Таблица 10: Структура транзакции отмены лизинга