Санкт–Петербургский государственный университет

Подсевалов Иван Алексеевич

Выпускная квалификационная работа

Разработка и тестирование блокчейн систем для горизонтально-интегрированных структур

Уровень образования: магистратура

Направление 02.04.02 «Распределённые вычислительные технологии»

Научный руководитель:

д.ф.-м.н.,

профессор

Богданов Александр Владимирович

Санкт-Петербург

2021 г.

Содержание

Введение

В последние годы тенденции цифровой трансформации очевидны. Повышение конкурентности, увеличение доли “цифровой” составляющей в бизнес-процессах способствуют усилению горизонтальной интеграции участников различных сегментов рынка с целью повышения эффективности и результативности межорганизационных процессов. Более тесное сотрудничество предоставляет новые возможности и взаимную выгоду для вовлеченных сторон, а также может улучшить их операционные и бизнес-показатели. В данном случае, вопросы прозрачности, конфиденциальности и приватности данных, которые возникают при внедрении горизонтальной интеграции, заслуживают отдельного внимания. Экосистемы должны быть подготовлены для совместного использования общих данных среди бизнес-партнеров или участников горизонтально-интегрированной структуры. А именно, возникает необходимость решить проблему совместимости систем, то есть их способность обмениваться информацией.

Взрывной рост генерируемых данных, тот объем, с которым приходится работать при современных задачах, снижает эффективность традиционных баз данных. Централизованные решения уже кажутся не столь рациональными; в такой ситуации стоит акцентировать свое внимание и отдать предпочтение DLT (Distributed Ledger Technology). Последнее имеет ряд преимуществ:

- во-первых, безопасность. Отсутствует централизация данных, за контроль и валидацию данных отвечают непосредственно участники сети;
- во-вторых, эффективность. Одноранговый обмен данными делает систему быстрой и высокоэффективной;
- в-третьих, функциональность. Технология распределенного реестра очень универсальна и находит применение во многих секторах. Различные отрасли, от здравоохранения до финансов, используют блокчейн для дальнейшего продвижения на рынке [1].

Вопрос доверия в децентрализованных системах, в частности в распределенных реестрах, стоит крайне остро. Депозиты заменяют другие проявления доверия. Вместо явного ранжирования (например, количество звезд отдельного пользователя на GitHub, Amazon или величина кармы на Habr), агенты будут доверять друг другу только в том случае, если будет предоставлено достаточное количество активов для подтверждения операции. Ключевая идея состоит в том, что, если контрагент совершает нежелательные действия, система может наказать обманывающую сторону и возместить причиненный ущерб агенту, пострадавшему от этих действий.

Возникает вполне закономерный вопрос: все ли проблемы решены для блокчейн систем? Не совсем. Определить, какой депозит требуется агенту - непростой вопрос. Чем больший депозит должен предоставить агент, тем большую безопасность имеет протокол, так как это увеличивает потенциальное наказание в случае нежелательных действий. Однако появляется потенциальная проблема: чем больше требуется депозит, тем меньше множество агентов, у которых достаточно активов, чтобы претендовать на участие. Более того, в большинстве протоколов необходимо учитывать два основных источника неопределенности.

Во-первых, относительная величина депозита по отношению к риску может меняться с течением времени, то есть напрямую зависит от события. Например, для криптовалюты Dai [2] требуется обеспечение в долларовом эквиваленте не ниже 150% получаемой суммы Dai для учета внезапных скачков цены базового эфира по отношению к цене в долларах США. А это значит, чтобы получить 100 Dai, агенту придется заложить криптоактив, эквивалентный 150 долларам США.

Во-вторых, необходимо учитывать личную информацию. Например, агент может иметь неизвестные скрытые мотивы или какие-то внешние стимулы, например, подкуп.

Для отслеживания зависимостей от событий или личной информации отдельного агента требуется чрезмерное обеспечение протоколами. Проще говоря, если риск зависит от одного из описанных выше источников, разработчик протокола зачастую умножает значение риска на коэффициент F, который учитывает оба источника неопределенности. Это относится к

большинству протоколов DeFi (децентрализованные финансовые сервисы), таких, как Dai или Compound [3]. В других протоколах, например TrueBit или PoS, риск не совсем ясен. В этих случаях нужно произвести оценку необходимого депозита. В любом случае, существует внутренняя дилемма между агентами, предоставляющими услуги (они же обязаны вносить депозит), и агентами, получающими услуги.

Децентрализованные платформы на основе блокчейн технологий в ближайшем будущем будут стремиться решить проблему доверия, в том числе путем добавления прозрачности. Самым ярким примером такой платформы является Ethereum, который предоставляет возможность разработки децентрализованных приложений (DApps) с помощью высокоуровневого объектно-ориентированного языка программирования Solidity. К сожалению, в рамках данной системы нет простого способа добавить значимый уровень приватности и конфиденциальности разрабатываемым приложениям. Рассмотрим классический пример - закрытый аукцион. Без каких-либо затруднений мы можем программно реализовать механизм открытых аукционов, но все ли так просто в случае с закрытыми аукционами? Логика, которая лежит в основе закрытых аукционов - отсутствие знаний у любого участника торгов о ставках других участников. Обычно процесс происходит в несколько этапов: сбор запечатанных предложений, вскрытие собранных предложений и определение победителя путем сравнения предложений. И здесь сразу возникает вопрос: как реализовать подобный механизм в блокчейн системе, где значения транзакций являются общедоступными?

Существует несколько способов решения данной проблемы. Во-первых, можно отправлять несколько заявок, часть которых является фиктивной. Во-вторых, служба имен Ethereum (ENS) предлагает пользователям отправлять только хеш своих ставок для регистрации имени у регистратора и вносить больше эфира, чем истинное значение ставки. Однако, ни один из приведенных выше механизмов не обеспечивает необходимый уровень приватности и конфиденциальности, и при этом ложится дополнительной нагрузкой на пользователя. В-третьих, можно прибегнуть к использованию криптографических решений для запуска произвольных смарт-контрактов

с сохранением конфиденциальности, которые, к сожалению, не полностью децентрализованы или слишком дороги для простых контрактов [4, 5].

В действительности, стремление к конфиденциальности не ограничивается аукционами. Если бы существовал механизм конфиденциальной отправки внутренней валюты в той же системе Ethereum, возможно, разработчики могли бы использовать его, чтобы добиться необходимого уровня конфиденциальности в своих приложениях. К сожалению, почти все известные подходы для обеспечения конфиденциальности транзакций [6, 7, 8] относятся к UTXO-модели представления балансов, где входами для новой транзакции являются неизрасходованные выходы предыдущих транзакций. Модель UTXO плохо подходит для приложений, которым необходимо поддерживать какое-то внутреннее состояние, о чем говорит [9] Виталик Бутерин, основатель Ethereum. Поэтому платформы смарт-контрактов работают, используя модель на основе учетных записей. Подробнее о моделях представления балансов говорится в разделе “Модели представления балансов”.

Вышеописанные проблемы, которые свойственны горизонтально-интегрированным структурам, характерны и для вертикально-интегрированных структур, которым также уделено внимание в рамках данной работы. Прежде чем перейти к обзору существующих практик, сформулируем цель и задачи работы.

Постановка задачи

Цель данной работы состоит в разработке приватной и конфиденциальной системы для горизонтально-интегрированных структур, основанной на блокчейн технологии, которая удовлетворяет свойству дифференциальной приватности. Процесс разработки включает в себя исследование ключевых вопросов, касающихся безопасности, конфиденциальности, приватности и дифференциальной приватности, а также основных механизмов и архитектурных решений, существующих в этой области на настоящий момент.

Особенностью работы является использование наиболее актуальных математически обоснованных практик и современных технологий для максимизации эффекта синергии.

Сформулируем задачи, выполнение которых позволит достичь поставленных целей:

- рассмотрение и изучение основных систем горизонтально-интегрированных структур;
- рассмотрение и изучение ключевых вопросов приватности и конфиденциальности, которые свойственны горизонтально-интегрированным структурам;
- исследование архитектурных решений и ключевых протоколов дифференциальной приватности;
- создание собственной приватной сети типовых узлов для исполнения приватных и конфиденциальных транзакций внутри сети.

Схематично модель можно представить следующим образом (Рис. 1). В рамках приватной сети запускаются докер контейнеры (с использованием собственного описанный образа), в каждом контейнере запускается один узел блокчейна Ethereum. Отдельно реализуется самовыполняющийся контракт, который публикуется в сеть. Все транзакции участников сети совершаются, при помощи смарт-контракта, используя собственный внутренний токен. Конечный пользователь запрашивает данные у серверной

программы, которая, в свою очередь, получает данные из приватной сети, от конкретного узла. В качестве ответа клиент получает зашумленные данные, которые удовлетворяют свойству дифференциальной приватности. Более подробно процесс описан в главе "Эксперимент".

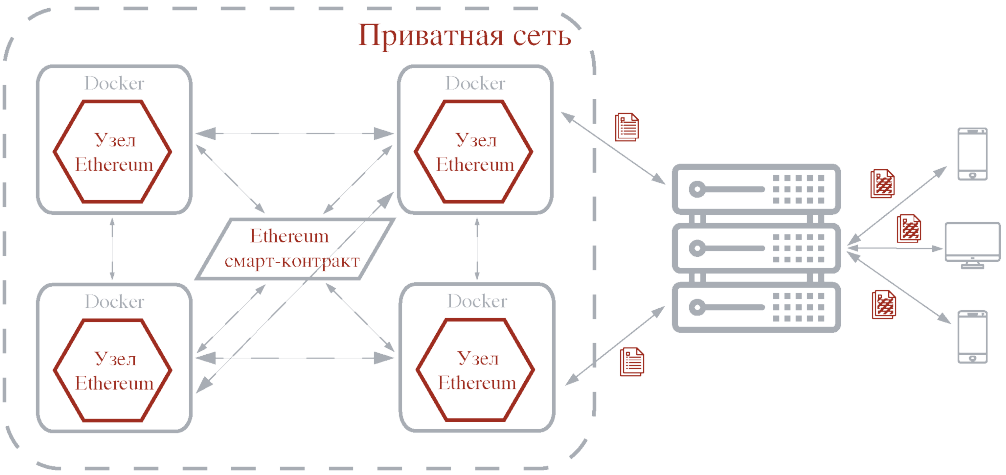

Рис. 1: Модель системы

Приступим к исследовательской части работы, а именно, к обзору существующих практик.

Обзор литературы

В данном разделе рассматриваются и обобщаются последние научные исследования и разработки в области приватности, конфиденциальности, а также дифференциальной приватности. Обзор затрагивает рассмотрение ключевых решений, основанных на технологиях распределенных реестров и блокчейн. В обзоре учитывались онлайн ресурсы, такие как Computer Communications, Digital Communications and Networks и другие.

Архитектурные положения. Система BGX/DGT является одним из случаев одновременно децентрализованного и распределенного приложения, теоретические положения подобного рода приложений неоднократно рассматривались разными авторами. Следует особо отметить работу Нараяна Прасти "Блокчейн. Разработка приложений"[10].

DGT представлена программным обеспечением типовых узлов, образующих сеть по определенным правилам (пример сети из двух кластеров представлен на Рис. 2). В этом смысле DGT следует воспринимать, как интеграционную платформу для сбора распределенных и децентрализованных данных, выполнения с ними граничных вычислений и создания объединенного распределенного реестра.

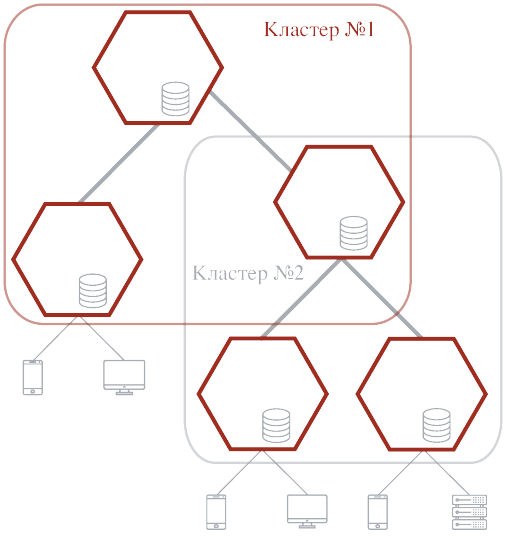

Рис. 2: Сеть BGX

Горизонтально-интегрированные структуры можно рассматривать как объединения неких бизнес-сущностей, связанных общими интересами, в

некотором смысле - как саморегулируемые объединения. Для выполнения своих задач и достижения целей, компании, входящие в такие объединения, должны следовать общим для всех правилам работы, оформленным в виде набора программ, имеющим какие-то общие обязательные признаки. Если рассматривать объединения компаний, имеющих своими товарами и услугами какие-то нематериальные активы, то в качестве основы для создания подходящих систем могут рассматриваться уже существующие системы на базе технологии блокчейн. Таким образом, мы будем рассматривать горизонтально-интегрированные структуры, представляющие из себя децентрализованное приложение (ДП) [11].

Многими авторами отмечаются обязательные признаки ДП. Так, например, Сирадж Равал выделяет [12] следующие:

1. открытый исходный код;
2. внутренняя валюта;
3. децентрализованный консенсус;
4. отсутствие центральной точки отказа.

Для того, чтобы придать децентрализованному приложению дополнительную функциональность в соответствии с воображаемым стандартом Blockchain 3.0 [13], следует дополнить этот ставший классическим набор признаков двумя новыми: наличием API, который унифицирует взаимодействие узлов сети, и достаточной производительностью. Авторы Wei Cai, Zehua Wang, Jason B. Ernst, Zhen Hong , Chen Feng , Victor C. M. Leung [11] отмечают желательные дополнительные признаки:

- улучшенная производительность;
- возможность офлайн транзакций;
- разумная стоимость содержания системы;
- гибкая поддержка системы;

- упрощенная идентификация.

Одним из обязательных признаков рассматриваемого ДП является наличие внутренней валюты; например, для системы BGX/DGT выбрана оригинальная двухзвенная комбинация из якорной валюты - это Ethereum и внутренней DGT Coin. Использование внутренней white-label [14] валюты позволяет производить внутренние расчеты между участниками бизнес-объединения.

В технологии Blockchain 1.0 используется консенсус, основанный на выполненной работе (PoW), что совершенно не подходит в случае распределенного приложения ввиду его исключительной энергозатратности и неудовлетворительной скорости. Для распределенного приложения актуальными могут быть вариации классического практического алгоритма Byzantine fault tolerance (PBFT) [15].

Хранение данных, с точки зрения высокой производительности и безопасности, логично использовать в структурах ациклического направленного графа DAG с использованием IPFS (межпланетной файловой системы) [16]. Использование IPFS обусловлено необходимостью организации децентрализованной сети с длительным сроком хранения данных и отсутствием единого центра управления данными. С использованием клиента IPFS пользователь получает возможность добавить в DAG некие данные, которые идентифицируются полученным при добавлении хешем. В системе используется граф на основе дерева Меркла - MerkleDAG (Рис. 3), который можно рассматривать как связанный список или дерево. При добавлении данных система генерирует пару из открытого и закрытого ключей SHA-256 и возвращает их пользователю. Данные в IPFS формируют общий merkleDAG, включающий все узлы. Все данные в IPFS общедоступны в рамках своей сети, а наличие закрытого ключа может служить доказательством права собственности на данные [13].

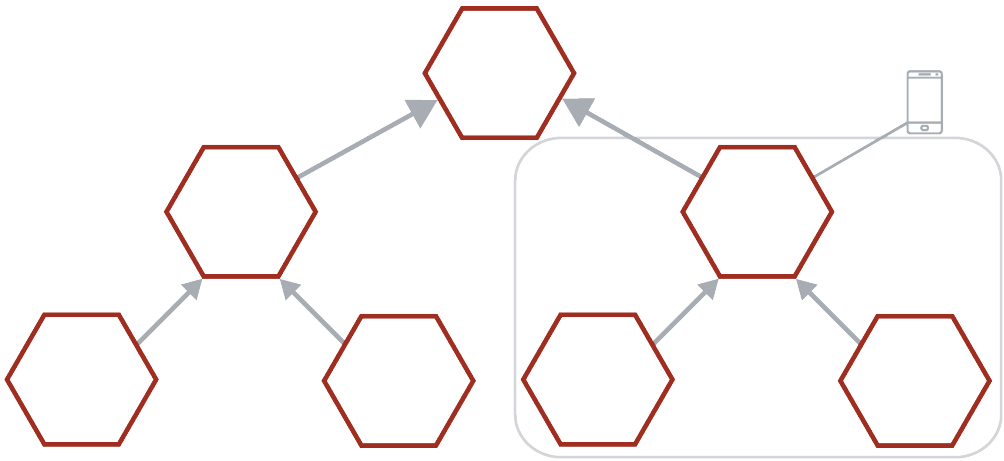

Рис. 3: Дерево Меркла (DAG)

В большинстве централизованных приложений пользователь может подтвердить личность отправкой контрольных кодов, переходом по ссылке и т.д. Этот процесс обозначается термином “Know Your Customer” ("знай своего клиента"). В данной системе модуль KYC может использоваться в комплекте с платежным шлюзом (Payment Gateway).

Теперь перейдем к подразделу дифференциальная приватность.

Дифференциальная приватность. Понятие дифференциальная приватность (ДП) было введено в 2006 году C. Dwork, F. McSherry, K. Nissim и A. Smith [18]. ДП является совокупностью методов на основе зашумления выходных данных посредством рандомизации или случайности. Общая идея, лежащая в основе этого механизма, заключается в том, что информация должна распространяться независимо от того, является ли конкретный агент злоумышленником или нет. Стоит сразу отметить, что это не какой-то конкретный процесс, а, скорее, свойство, которым может обладать процесс. Свойство дифференциальной приватности гарантирует, что результат анализа будет почти неотличим вне зависимости от того, есть ли данные отдельного агента в наборе или нет. Визуально это свойство можно представить следующим образом (Рис. 4):

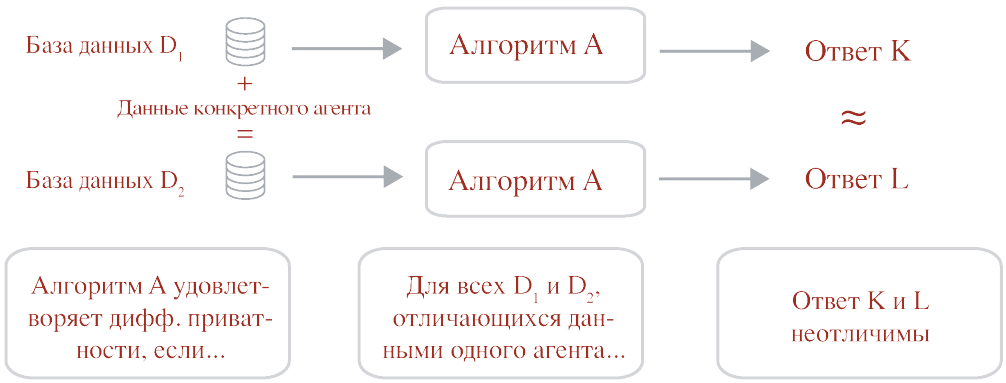

Рис. 4: Дифференциальная приватность

При работе с приватными данными существуют различные классические подходы, которые стремятся решить проблему приватности. Но, к сожалению, ни один из подходов не дает желаемого результата. Самое распространенное решение — это деидентификация, то есть удаление всей идентифицирующей информации из исходных данных. Сразу стоит отметить, что анонимизированные данные являются целью атак, которые называются “атаки связывания”. Другое решение, которое пользуется популярностью у статистических баз данных — это агрегирующие запросы (например, вычисление среднего). Агрегирование защищает приватность лишь тогда, когда диапазон данных, по которым делается выборка, достаточно велик.

Существует ряд популярных примеров, которые ярко описывают атаки (деанонимизация, атака связывания, атака реконструкции) на статистические базы данных [19, 20, 21, 22].

L. Sweeney в 2000 году показал, что для однозначной идентификации 87% населения США, достаточно знать всего три значения: почтовый код (ZIP), дату рождения и пол человека. Исследователь потратил 20 долларов США для приобретения регистрационного списка избирателей, выгрузил базу данных пациентов, которая распространяется для исследователей бесплатно, и связал все данные. В итоге, удалось идентифицировать медицинскую карту губернатора Массачусетса [23].

В 2011 году исследователи H. Zang и J. Bolot показали [24], что можно однозначно идентифицировать людей по анонимным необработанным данным об их местоположении, которые могут быть собраны с помощью обыч-

ного смартфона. В 2013 году исследователи de Montjoye, Yves-Alexandre, César A. Hidalgo, Michel Verleysen и Vincent D показали [25], что данные о мобильности имеют такую же идентифицирующую ценность, что и отпечаток пальца человека.

В 2013 году исследователи из MIT показали [26], что фамилию человека можно однозначно установить из личных геномов по запросу из базу данных генетической генеалогии. Они демонстрируют, что метаданных, таких как возраст и состояние здоровья, вполне достаточно для идентификации людей.

Примером деанонимизации является исследование [27] A. Narayanan и S. Vitaly Shmatikov, которые в 2008 году смогли частично деанонимизировать базу данных Netflix, связав рейтинги открытых аккаунтов IMDB и предложенный тренировочный набор данных от Netflix для улучшения их рекомендательной системы.

Наиболее подробно вопрос дифференциальной приватности раскрыт в разделе “Дифференциальная приватность”. Теперь перейдем к рассмотрению горизонтально-интегрированных структур.

Глава 1. Горизонтально-интегрированные структуры

В информационных технологиях последнего времени произошло значительное изменение возможностей, как чисто технических, так и математических.

К этим новым возможностям можно отнести:

- появление новых СУБД, как правило это БД "ключ-значение";
- дополнение СУБД технологиями резервирования и репликации (SMR);
- взрывное развитие криптографических методов защиты информации.

Эти изменения вызвали, в свою очередь, изменения во многих сферах организации и ведения бизнеса, что привело к небывалому росту интереса к новым технологиям. Эти процессы, безусловно, разогрели интерес к разработке многочисленных программных продуктов, основанных на технологиях распределенных реестров.

Прежде чем переходить к распределенным системах, стоит упомянуть другие варианты систем, а именно централизованную и федеративную.

1.1 Централизованная система

В настоящее время централизованная модель является наиболее распространенной. Централизованно организованные системы непосредственно управляют работой отдельных частей программ, а весь поток информации движется через общий центр. Работа отдельных компьютеров с клиентскими программами напрямую зависит от способности центра принимать и отправлять назад информацию, а также осуществлять централизованное управление.

Ключевыми особенностями подобной системы являются: централизованное конфигурирование, централизованное шифрование, масштабирование и другие. Главным недостатком является то, что пользователь полностью зависит от центрального узла, а это значит, что пользователь не

может настроить систему под себя и в случае неполадок на центральном узле, система может стать неработоспособной в принципе.

Между централизованной и распределенной системами существует промежуточное решение, которому посвящен следующий подраздел.

1.2 Федеративная система

Промежуточным можно назвать подход, при котором система является и децентрализованной и распределенной, можно назвать ее федеративной. В этом случае отдельные компьютеры в системе отдают часть своей самостоятельности в обмен на объединение в кластеры (федерации). При этом на уровень федерации передается и часть приватности и конфиденциальности.

Как продвинутую парадигму можно рассматривать распределенную систему, где все приватные данные находятся у клиента, на уровне клиентских приложений. При этом подходе соблюдается максимальная приватность, но, очевидно, такой подход приводит к излишней изолированности.

1.3 Распределенная система

Всего десять с небольшим лет назад появление первой блокчейн платформы Bitcoin дало старт истории первого программного продукта, рассчитанного на работу в течение ближайших ста с лишним лет, если судить по организации учета времени внутри системы. У пользователей такой платформы появился интерес и к наращиванию функциональности системы распределенного реестра, что привело к появлению программной среды, призванной расширить возможности реестров. В свою очередь появление возможности организации контрактной среды привело к появлению интереса к организации бизнес-сообществ, объединенных общими интересами и специфическими запросами. Такие сообщества могут организовывать как открытые, так и закрытые “клубы по интересам”, однако их будет объединять желание развивать бизнес-процессы в безопасной и доверительной среде. Стремление к объединению близких по роду деятельности бизнесов можно классифицировать как процесс горизонтальной интеграции, когда

организации образуют некий горизонтальный кластер, в то время как множество таких кластеров могут организовывать вертикально организованные структуры.

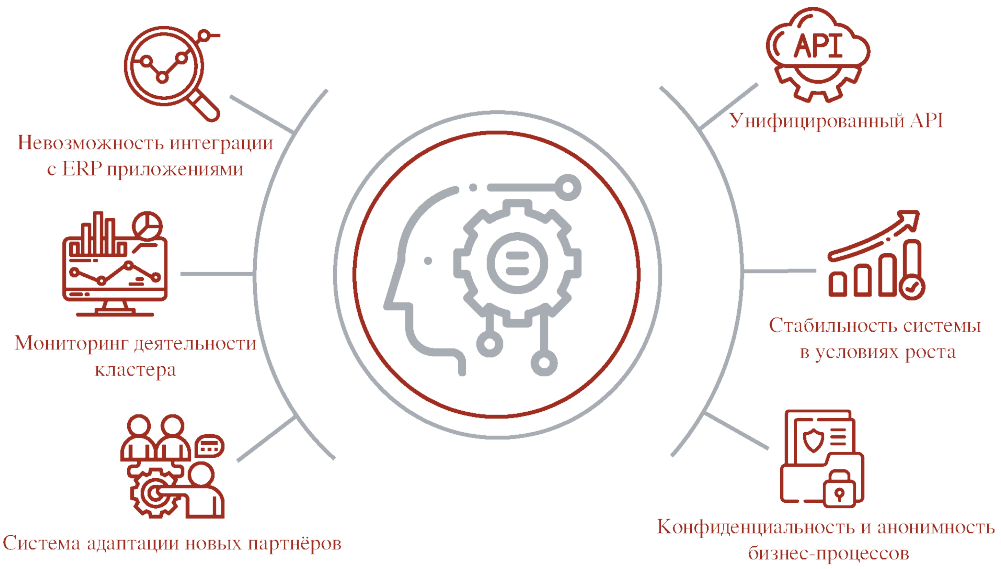

Рис. 5: Ключевые признаки горизонтально-интегрированных структур

Горизонтально-интегрированные структуры можно охарактеризовать несколькими общими признаками (Рис. 5):

- невозможность использовать какую-то общую информационную систему (ERP);
- необходимость мониторинга общей деятельности кластера;
- система одобрения вступления новых членов в систему;
- организация унифицированного API;
- стабильность системы в условиях роста;
- возможность организации конфиденциальности бизнес-процессов и анонимности при раскрытии статистической информации.

К таким сферам можно отнести логистику, т.е. управление цепями поставок, управление грузоперевозками, банковское дело (системы межбанковских платежей), системы распределения заказов и работ (тендерные и государственные закупки и конкурсы).

Просуммировав все плюсы от внедрения таких систем, можно заключить, что потенциальными сферами их применения являются все те системы, которые связаны с обработкой цифровых или оцифрованных активов, имеющих высокую ценность, и при этом при обычной реализации имеют жестко централизованную архитектуру.

Замена подобной архитектуры горизонтально-интегрированных систем на децентрализованную или распределенную способна поднять на качественно новый уровень решение вопросов обеспечения непрерывности и устойчивости бизнеса.

Несомненно представляет интерес возможность эмиссии участниками горизонтально-интегрированных структур собственных криптовалют или, лучше сказать, токенов (фишек, расписок). Выпуск токенов – это весьма удобная форма для поддержания какой-либо общей деятельности, в первую очередь это, конечно, содержание общей вычислительной инфраструктуры.

Понятно, что для того, чтобы обычное централизованное приложение сохраняло работоспособность в течение длительного времени, оно должно участвовать в генерации прибыли. У горизонтально-интегрированных структур, которые работают на основе децентрализованных приложений (ДП), нет очевидного хозяина, но все равно узлы ДП нуждаются в постоянно действующих сетевых ресурсах. Именно здесь уместно использование внутренней валюты. Количество токенов, получаемых узлом, определяется используемым протоколом консенсуса.

Естественно, использование определенного количества токенов для поддержания структуры должно быть экономически оправдано. Проблему можно решить побуждая пользователей системы расплачиваться внутренними токенами, решается проблема устойчивости системы.

Переходим к рассмотрению моделей представления балансов.

Глава 2. Модели представления балансов

В настоящее время системы распределенного реестра базируются на двух основных схемах представления балансов участников: на UTXO-схеме (или Unspent Transaction Output) или схеме аккаунтов (account-based model).

Если представить систему распределенного реестра как конечный автомат, то, в соответствии с определением, система имеет состояние. Если она построена с учетом необходимости хранения команд или действий пользователя, в каждый отдельный момент эта информация называется внутренним состоянием системы. Таким образом распределенный реестр можно рассматривать как конечный автомат, который записывает какие-то происходящие события или действия пользователя. С записью каждого нового блока в реестр происходит переход системы в новое состояние в соответствии с внутренней логикой.

Каждый распределенный реестр, независимо от того, использует он UTXO-модель или модель аккаунтов, следует этому алгоритму. Действия пользователя, а в основном это какие-то транзакции, распространяются по сети и, объединяясь в блоки, постоянно записываются в реестр. Балансы сторон, участвующих в транзакциях, изменяются при переходе реестра в новое состояние. Разница между моделями UTXO и account-based заключается в способе учета этих балансов.

В UTXO модели, использующей схему данных реестра в виде ориентированного ациклического графа, все выходы транзакций (как израсходованные, так и неизрасходованные) представляют вместе глобальное состояние реестра. В модели аккаунтов глобальное состояние реестра - это набор аккаунтов и соответствующие им балансы. Рассмотрим модели более внимательно, а также приведём своего рода сравнение данных моделей. Начнем с модели UTXO.

2.1 UTXO-модель

Модель UTXO (Рис. 6) использует схему данных реестра в виде ориентированного ациклического графа (DAG), описывающего перемещение активов между участниками деловой деятельности.

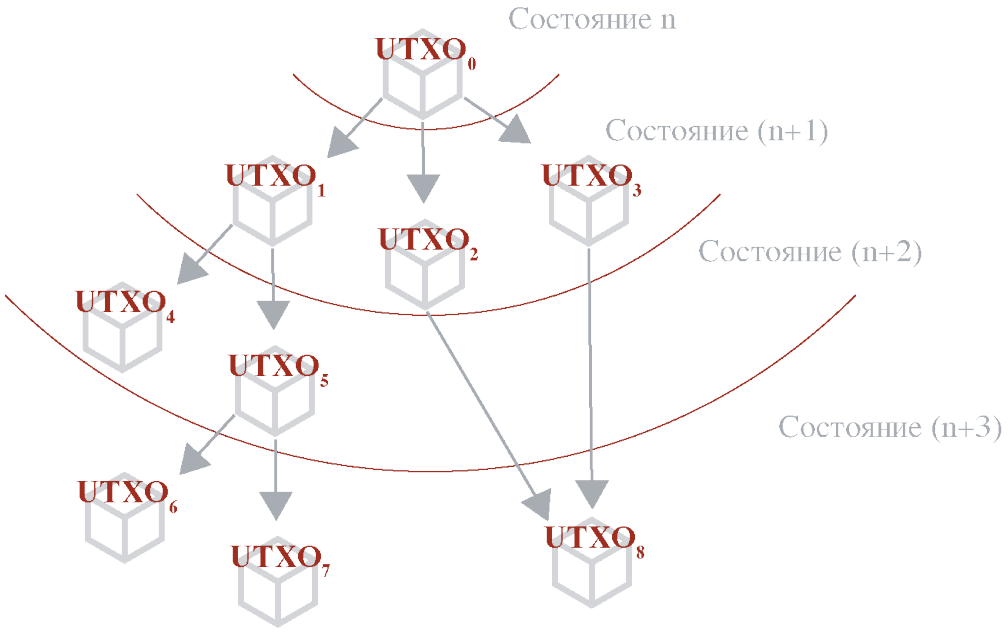

Рис. 6: Модель UTXO

Каждое внутреннее состояние представляется одним блоком в распределенном реестре и результат каждой транзакции отображается в виде узла графа, и в свою очередь одна или более новые транзакции являются выходами из предыдущего результирующего узла. Модель UTXO основана на подсчете подмножества тех вершин графа, которые еще не стали входом для каких-то новых транзакций. Чтобы узнать баланс конкретного пользователя системы, нужно отследить и просуммировать все его непотраченные (unspent) выходы транзакций.

Самой первой системой распределенного реестра, построенной по модели UTXO, стала Bitcoin. В дальнейшем большинство криптовалютных платформ тоже стало базироваться на основе UTXO-модели.

UTXO модель не содержит понятия счетов или кошельков на уровне протокола. Эта модель целиком базируется на концепции отдельных транзакций, сгруппированных в блоки. Выходы транзакций должны быть потрачены в следующей транзакции целиком, “сдача” с транзакции должна быть размещена по контролируемому адресу. Так как в этой модели нет концепции счетов/кошельков, вся ответственность за ведение своего баланса ложится на клиента, в клиентском приложении должен быть организован учет по всем контролируемым адресам, из чего и получается текущий

баланс пользователя.

2.2 Модель аккаунтов

Модель транзакций на основе учета аккаунтов (счетов), представляющих балансы активов, весьма похожа на модель учета счетов и транзакций в традиционных банках. Глобальное состояние в модели аккаунтов можно представлять как глобально обновляемую вместе с транзакциями таблицу аккаунтов и балансов на этих счетах (Рис. 7).

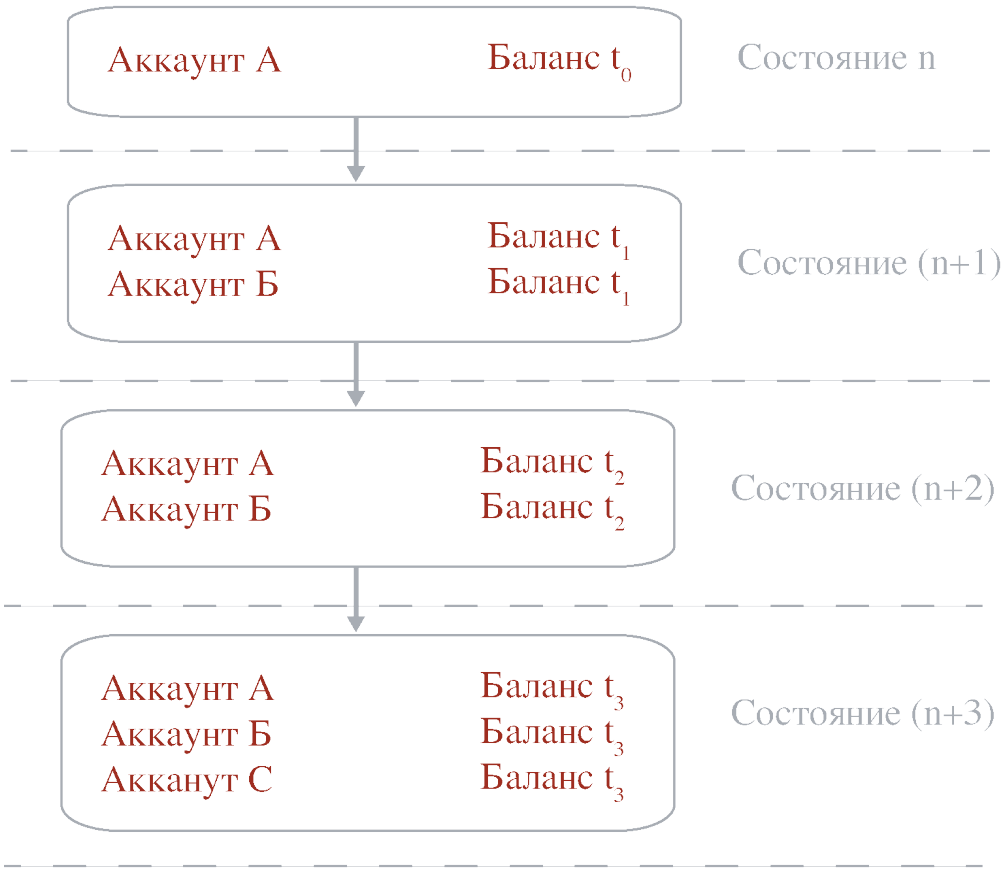

Рис. 7: Модель аккаунтов

Одной из известных систем, использующих эту модель, является блокчейн Ethereum. В нем используется два вида аккаунтов:

- частные счета клиентов, контролируемые через систему приватных ключей;
- счета, контролируемые кодом смарт-контрактов.

Когда пользователь впервые создает Ethereum-кошелек, одновременно создается аккаунт в глобальном состоянии и распространяется на все ноды сети. Аналогично, размещение смарт-контракта приводит к созданию аккаунта, контролируемого смарт-контрактом. Смарт-контракт сам по себе может иметь какие-то монеты, которые могут перераспределяться в соответствии с внутренней логикой контракта. Каждый аккаунт в Ethereum имеет баланс и "внутреннее" содержимое. Изменение баланса отправителя происходит одновременно с изменением баланса получателя.

2.3 Сравнивая модели

У каждой модели есть свои плюсы и минусы, сильные и слабые стороны. Сравнивать их следует с нескольких точек зрения.

Концептуально, UTXO-модель это модель основанная на системе проверок отправленных пользователем транзакций. В случае успешной проверки использования нераспределенных выходов и подписей, происходит обновление глобального состояния.

Модель аккаунтов, с другой стороны, основана на системе вычислений. Пользователь отправляет в систему некие инструкции для изменения глобального состояния. Сама система после этого вычисляет новое состояние с учетом необходимости преодоления определенных сложностей, связанных с масштабированием и распределенностью данных.

Если говорить о конфиденциальности, у каждой модели есть свои плюсы, однако обе модели в версии "как есть" не содержат встроенных механизмов обеспечения конфиденциальности в условиях открытости распределенного реестра. Любой злоумышленник может загрузить себе копию реестра со всеми транзакциями. Изучив данные транзакций в реестре, можно отследить связи между транзакциями и аккаунтами. Схема предусматривает постоянное добавление новых транзакций, при этом все старые не удаляются, то есть достаточно нарушить конфиденциальность в одной точке и далее отследить всю историю. Кроме этого, существует опасность использования косвенной информацию для определения личности владельцев аккаунтов.

В UTXO модели применима техника постоянной смены адресов, что требует от злоумышленника дополнительных усилий для раскрытия личностей отправителя/получателя.

Известным решением по обеспечению конфиденциальности транзакций для модели аккаунтов является использование встроенных миксеров, когда к реальным адресам "подмешиваются" адреса, не несущие смысловой нагрузки. Требования к защищенности модели аккаунтов должны быть очень высокие, так как однократное раскрытие личности отправителя/получателя приводит к полной потере анонимности пользователя аккаунта.

Модель аккаунтов предоставляет очевидные преимущества, когда мы рассматриваем ее с точки зрения использования смарт-контрактов. Эта модель интуитивно более понятна, логика программ более ясная и менее затратная с точки зрения использования вычислительных мощностей. С другой стороны, в UTXO модели гораздо проще реализуются параллельные вычисления, так как они могут базироваться на разных выходах транзакций. В то же время в модели аккаунтов вычисления с использованием одного и того же аккаунта должны выполняться, очевидно, последовательно.

В общем, UTXO модель лучше работает с простыми однократными транзакциями, модель аккаунтов лучше обслуживает сложную логику. Сразу возникает желание использовать некую гибридную логику, которая пользуется преимуществами и той и другой модели.

Существующие в настоящее время гибридные системы пытаются соединить блочную технологию с виртуальной машиной типа Ethereum, при этом прослеживается тенденция ухода от консенсусной модели PoW, с использованием которой построен Bitcoin. В модифицированной модели UTXO в качестве неизрасходованных входов можно использовать транзакции, которые впоследствии уничтожаются, а в результате создаются новые нераспределенные выходы.

Очевидным выглядит желание разработчиков гибридных систем использовать стратегию, ориентированную на преимущественное применение мобильных устройств.

Глава 3. Приватность и конфиденциальность

Технологии блокчейн появились в ответ на общественный запрос на создание доверительной среды для совместного ведения бизнеса большим кругом юридических и физических лиц, которые частично или полностью не доверяют друг другу, но заинтересованы в решении общих задач. Важнейшей отличительной особенностью систем распределенного реестра от ранее известных информационных систем (ИС) является то, что ИС строятся вокруг общего бизнес-процесса, но при этом нельзя рассчитывать на честность и аккуратность участников этих процессов.

Архитектуру ныне существующих систем управления распределенным реестром можно изобразить в виде модели, у которой четко просматриваются три уровня: транспортный (сетевой), уровень хранения данных (реестр и операции с ним) и прикладной (смарт-контракты). Естественно, обеспечение безопасности в работе с данными на каждом уровне обеспечивается по-своему. Рассмотрение вопросов, связанных с защитой транспортного уровня, выходит за рамки данной работы.

Первое, что стоит рассмотреть - пользовательскую приватность.

3.1 Пользовательская приватность

В настоящий момент публичные блокчейн-системы декларируют абсолютную анонимность транзакций, и за пределами рассматриваемой системы это действительно в большинстве случаев именно так. И в нашем случае остальные участники сети не должны иметь возможности отследить кого-либо через его транзакции, а означает, что приватность и данных и пользователя следует рассматривать комплексно. Это предполагает рассмотрение максимального количества возможных сценариев работы системы. Особенно важным является рассмотрение процедуры проведения расчетов за пределами рассматриваемой сети, например, в случае обмена токенов на обычные валюты, что означает потенциальное раскрытие части приватной информации. Например, при создании учетной записи в сети BGX/DGT пользователю не нужна идентификация, но при конвертации в фиатные деньги обменные биржи запрашивают подтверждение личности.

В такой ситуации выхода за пределы приватности следует применять KYC / AML процедуры (Рис. 8).

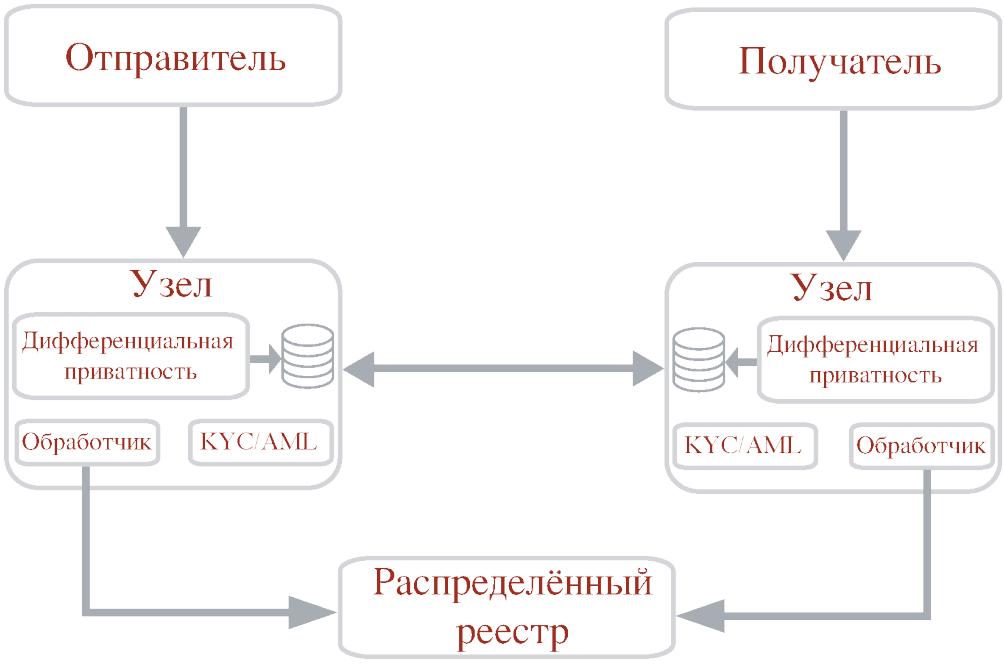

Рис. 8: Схема типичных узлов

Политика приватности, например, сети BGX/DGT основана на нескольких важных принципах:

- транзакции в системе полностью отделены от персональных данных пользователя и профиль пользователя размещен в отдельном хранилище виртуальных ссылок, то есть данная информация не попадает в узлы, за исключением выделенных серверов в кластере, которые обслуживают персонализацию;
- управление конфиденциальной информацией в узлах не может быть унифицировано из-за разного подхода к приватности для разных горизонталей, однако для одного уровня (в рамках одной юрисдикции) это можно декларировать;
- при проведении выборок чувствительной информации применяется подход дифференциальной приватности (Рис. 8), когда к открытой информации подмешивается некий шум, который не влияет на основную информацию, но препятствует идентификации пользователя;

- если узел содержит сервер KYC / AML, транзакции проходят обработку в этом сервере и получают дополнительную подпись, гарантирующую прохождение соответствующей процедуры.

Наравне с пользовательской приватностью стоит рассмотреть вопрос конфиденциальности транзакций.

3.2 Конфиденциальность транзакций

Транзакционный механизм является центральным элементом практически любой децентрализованной системы и охватывает несколько общих этапов обработки. Первым шагом является формирование конечной транзакции; на данном этапе происходит получение необходимой информации, основных атрибутов и генерация необходимой формы на основе полученных данных. Следующим шагом является валидация, то есть проверка правильности созданной транзакции; валидация осуществляется как на клиентской стороне, так и на серверной. После успешной валидации происходит проверка основных условий: наличие достаточного количества средств для совершения транзакции, отсутствие двойных трат или любых иных мошеннических действий. По прохождении всех этапов транзакция сохраняется и распространяется внутри сети. Жизненный цикл транзакции можно увидеть на Рис. 9.

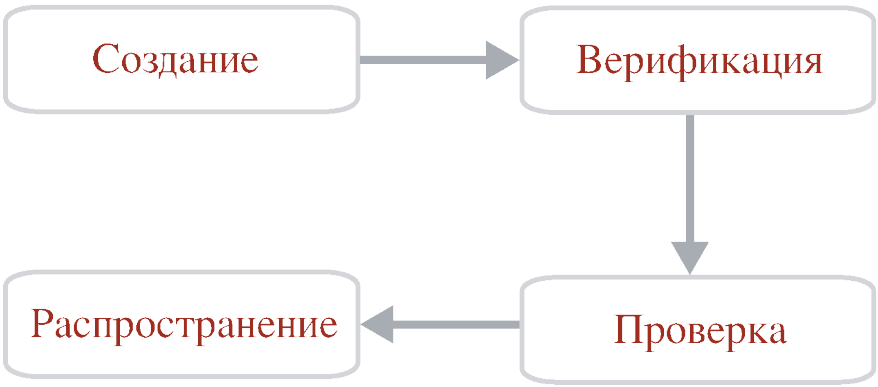

Рис. 9: Жизненный цикл транзакции

Рассмотрение конфиденциальности транзакций следует начать с простого деления реестров на два типа: первый - это система нераспределен-

ных выходов транзакций (UTXO), второй - система аккаунтов (account-based).

В свое время первая появившаяся UTXO-система Bitcoin обещала полную приватность, однако период анонимности достаточно быстро закончился, и началась затяжная борьба за конфиденциальность. Известно, что простейший способ обеспечить конфиденциальность транзакций в UTXO-системе для любого участника – это создавать множество адресов своих электронных кошельков, в идеале – открывать новый кошелек для каждой новой транзакции. Однако, наличие большого количества кошельков весьма неудобно для пользователя. Для решения вопросов конфиденциальности в реестрах, использующих UTXO-систему, в прошлом было разработано несколько программных продуктов, например, Dash, Zcash, Zcoin, Monero.

Большинство решений для достижения анонимности и конфиденциальности для блокчейн-платформ открытого типа построено на нескольких основных принципах: это, например, перемешивание адресов, введение в систему промежуточной стороны, которая обеспечивает защищенную обработку транзакций во внутренних токенах, использование специальных криптографических методов (например, доказательств с нулевым разглашением), использование одноразовых адресов кошельков и заверение транзакций посредством кольцевой подписи.

Для обеспечения конфиденциальности транзакций для UTXO-модели на блокчейн-платформе закрытого типа чаще всего вводятся системы балансов, которые проще обслуживать. Такие решения предусматривают кодирование токенов специальным способом, позволяющим совершать анонимные транзакции.

Аналогичным образом идет развитие программных решений, направленных на обеспечение приватности и конфиденциальности транзакций для модели аккаунтов. Как известно, баланс аккаунта обновляется при добавлении связанной с ним транзакции в реестр, и в любой момент времени баланс – это суммарный результат всех операций с аккаунтом. То есть нам необходимо защитить анонимность аккаунта в течение всей его жизни, что, очевидно, весьма непросто. Понятно, что потеря приватности отправите-

ля или получателя в единственной транзакции приводит к полной потере анонимности владельца аккаунта.

В настоящее время система Zether обещает достаточно интересное решение, позволяющее обеспечить как конфиденциальность содержания транзакций, так и приватность участников транзакций. Решение ориентировано на блокчейн-платформу Ethereum и предполагает выпуск в обращение специальных Zether-токенов (ZTH). Для операций с ними разработаны специальные смарт-контракты и используются специальные аккаунты. Аккаунты идентифицируются при помощи открытых ключей схемы открытого шифрования Эль-Гамаля и, таким образом, не ассоциированы с адресами аккаунтов платформы Ethereum. Владельцы аккаунтов могут конвертировать "родную"криптовалюту платформы Ethereum (эфир) в эти токены и обратно при предъявлении соответствующих ключей аккаунта. Аккаунты также имеют механизмы защиты от повторной траты токенов, в некотором смысле подобные аналогичным механизмам платформы Ethereum, а именно последовательную нумерацию транзакций на каждом аккаунте и подписание содержания каждой транзакции ее инициатором. Чтобы перевести токены с баланса одного аккаунта на баланс другого аккаунта, их необходимо зашифровать при помощи открытых ключей обоих аккаунтов, приложив криптографические доказательства утверждений двух типов: во-первых, доказательство того, что в обоих шифртекстах зашифрована одна и та же сумма активов, вовлеченных в транзакции, и, во-вторых, того, что на балансе отправителя имеется достаточно активов для совершения транзакции. Для реализации столь нетривиальных доказательств авторы предлагают оригинальную гибридную конструкцию, составленную из сигма-протоколов и доказательств типа Bulletproofs, названную ими "Σ-Bullets".

Важным свойством, которым должна обладать практически любая современная система - дифференциальная приватность, которой посвящён следующий раздел.

Глава 4. Дифференциальная приватность

Чуть ранее в работе уже говорилось, что дифференциальная приватность - свойство, которым может обладать процесс. Также было упомянуто, что свойство дифференциальной приватности гарантирует, что результат анализа будет почти неотличим вне зависимости от того, есть ли данные отдельного агента в наборе или нет. Исходя из этого, можно рассчитать и доказать, что процесс удовлетворяет требованиям дифференциальной приватности.

Для контроля и регулирования требуемого уровня приватности вводится параметр ϵ, который называют потерей приватности (privacy loss). Здесь же стоить ввести еще одно значимое определение - смежность баз данных. Базы данных D_1 и D_2 называются смежными, если они отличаются ровно одной строкой. Чем меньше значение параметра ϵ, тем менее различимы две смежные базы данных и более защищены данные отдельных пользователей (Рис. 10).

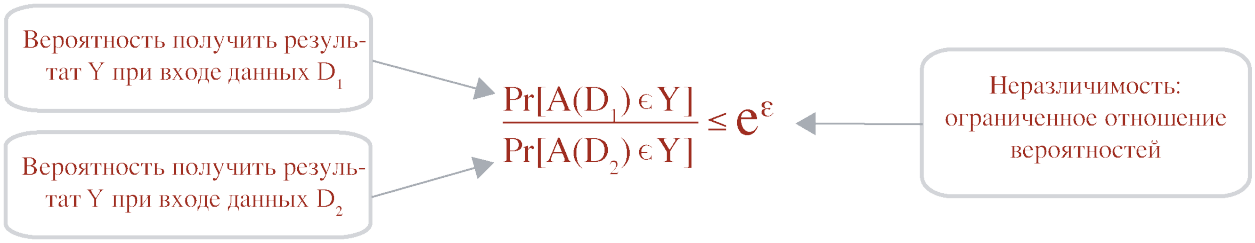

Рис. 10: Формальное определение дифференциальной приватности

В большинстве случаев мы можем соблюсти свойство дифференциальной приватности, добавляя случайный шум к исходным данным. Основная проблема, которая возникает при использовании ДП, - определить, куда и сколько шума стоит добавить. Существуют различные подходы для решения данной задачи, так называемые механизмы зашумления, которые мы рассмотрим чуть позже. Наиболее популярные - это механизм Лапласа и экспоненциальный механизм [18, 28].

Очевидно, что данные с повышенной степенью конфиденциальности требуют больше шума для удовлетворения свойства дифференциальной приватности, то есть определенного значения ϵ. Также очевидно, что чем больше шума мы добавляем в результирующие данные, тем сильнее мы

снижаем полезность полученных данных. Здесь нужен некий компромисс между приватностью и полезностью данных.

4.1 Архитектурные решения

Архитектуры систем, обладающих свойством дифференциальной приватности, должны одновременно учитывать как приватность, так и безопасность. Приватность следит за тем, что стоит исключить из исходных данных, а безопасность, в свою очередь, контролирует доступ к данным, но не дает гарантий касательно их содержания.

При разработке систем, удовлетворяющих свойству дифференциальной приватность, важно понимать, от чего мы стараемся защититься, понимать модель угроз. Также важно понимать, на каком этапе могут возникнуть проблемы. Если злоумышленник сможет целиком скомпрометировать сервер с конфиденциальными данными, то необходимо подготовить систестему, чтобы она смогла этому противостоять.

На сегодняшний день существуют несколько архитектурных решений, остановимся на каждом из них чуть подробнее.

4.1.1 Центральная модель

Наиболее известная модель дифференциальной приватности - центральная модель [28]. В данной модели хранилище данных или агрегатор имеет доступ к реальным данным. Также, каждый пользователь без дополнительного шума отправляет свои данные в хранилище. А уже агрегатор берет эти данные и преобразует их с помощью дифференциально приватного механизма. Принцип работы можно увидеть на Рис. 11.

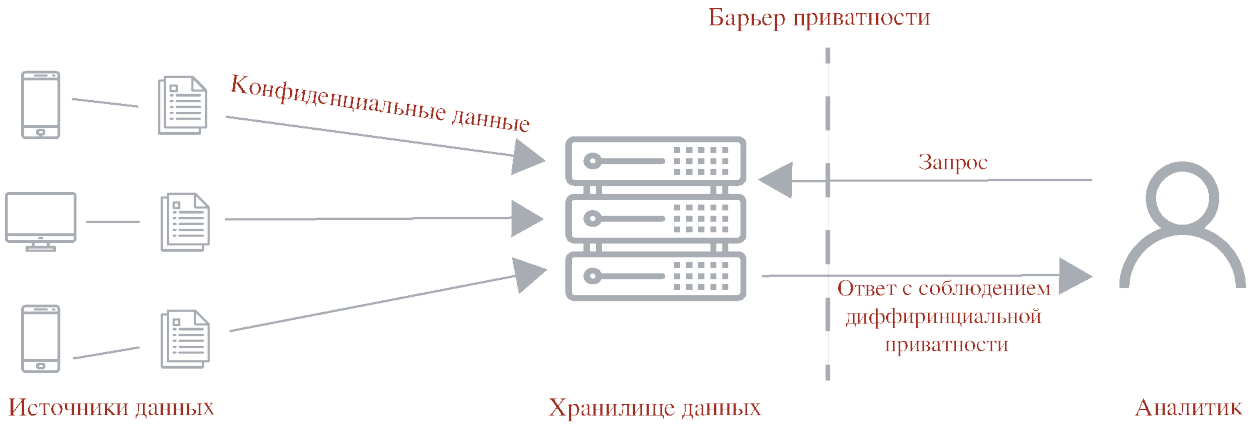

Рис. 11: Центральная модель дифференциальной приватности

Механизм дифференциальной приватности применяется только один раз, в конце процесса, после чего агрегатор может поделиться этими данными с третьими лицами (например, аналитиками). Эта модель имеет существенное преимущество, а именно - точность. В центральной модели зачастую не нужно добавлять много шума, чтобы выдать достаточно полезные данные с низким значением ϵ.

Безусловно, у этого есть обратная сторона медали. Агрегатору нужны реальные данные (в том числе конфиденциальные), а пользователи (источники данных) должны доверять агрегатору, чтобы делиться с ним данными. Достаточно сложно добиться пользовательского доверия к хранилищу данных, что, в свою очередь, является проблемой. Кроме того, в центральной модели все данные собираются в одном месте, что существенно увеличивает риск утечки данных.

4.1.2 Локальная модель

Альтернативой централизованной модели дифференциальной приватности служит локальная модель (Рис. 12). В этой модели все еще есть хранилище данных, но в данном случае оно не оперирует исходными данными. Вместо этого каждый источник данных на своей стороне производит вычисления - зашумление. После чего эти данные отправляются в хранилище данных.

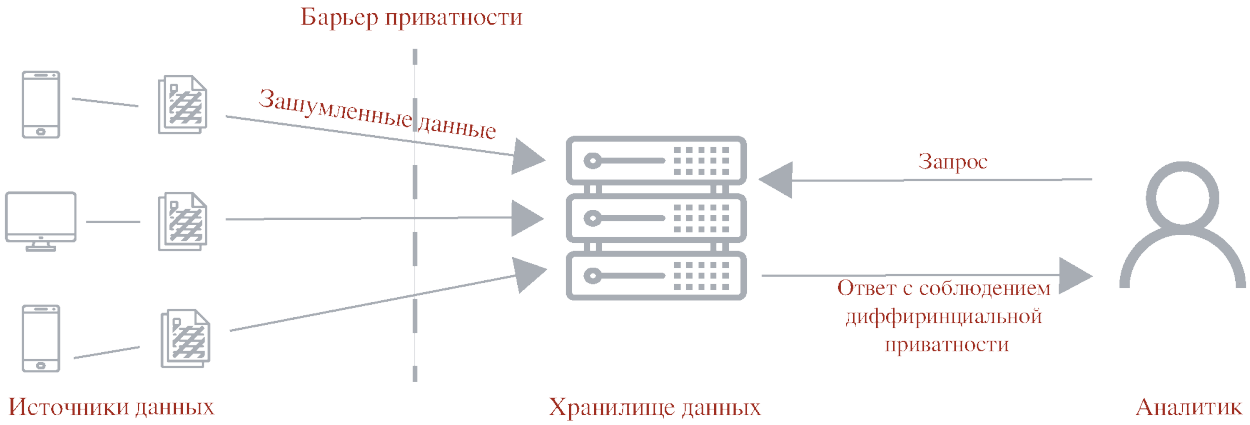

Рис. 12: Локальная модель дифференциальной приватности

После сбора зашумленных данных агрегатор может своими внутренними алгоритмами вычислить некоторую статистику и предоставить ее третьим лицам. Последнее не обязано быть дифференциально приватным, так как данные изначально анонимны. Теоретически, хранилище может предоставлять доступ ко всем данным.

Большим преимуществом данной модели является то, что она не требует доверия к хранилищу данных. Поскольку каждый пользователь использует дифференциально приватный механизм на своей стороне, данные по-прежнему будут в безопасности, даже если хранилище данных является вредоносным. Подобное свойство делает модель подходящей для случая, когда сложно добиться доверия от хранилища данных. Это основная причина, по которой локальная модель дифференциальной приватности была выбрана для таких систем как Google RAPPOR [29] и система сбора данных Apple [30].

Но и эта модель имеет свои недостатки. Поскольку каждый пользователь должен добавлять шум к своим данным, общий шум намного больше по сравнению с центральной моделью. Зачастую для получения конечных полезных данных требуется гораздо большая выборка, чем в той же центральной модели. Чтобы смягчить эту проблему, на практике зачастую используют большее значение параметра ϵ.

4.1.3 Гибридная модель

Центральная и локальная модели имеют как преимущества, так и недостатки, и сейчас основные усилия направлены на то, чтобы взять от них все лучшее. Новый дизайн архитектуры называется ESA: Encode, Shuffle, Analyze. Подобная архитектура реализована, например, в Prochlo [31]. Если в прошлых вариантах мы имели только двух участников, то в данном случае появляется третье звено, которое на рисунке (Рис. 13) обозначено как “перемешиватель”.

Пользователь добавляет шум на исходные данные (аналогично первым двум моделям) и направляет их перемешивателю. Перемешиватель - это некий промежуточный процесс, который удаляет все имеющиеся идентификаторы и группирует схожие данные вместе, после чего сгруппированные данные отправляются в хранилище данных. Анализатор фактически расшифровывает данные и вычисляет интересующую нас статистику.

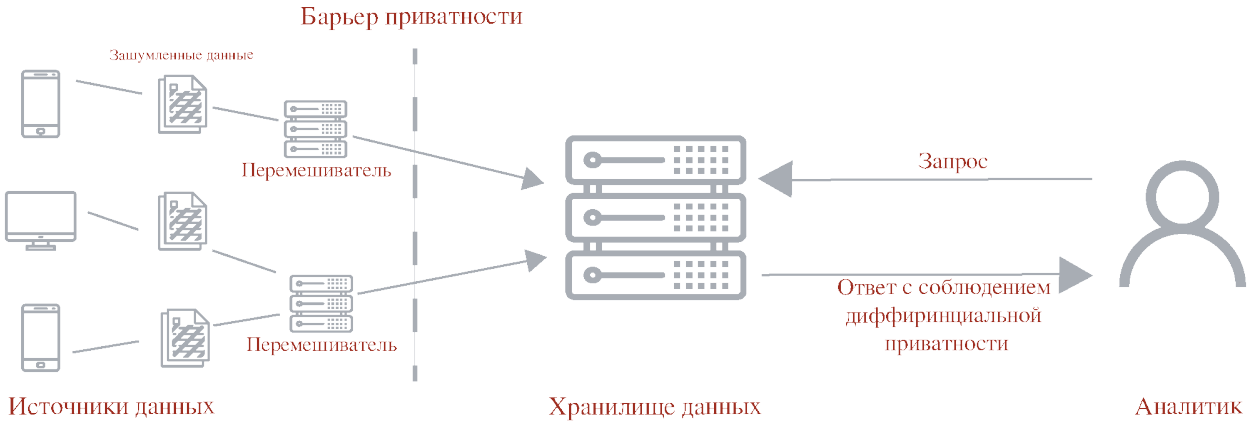

Рис. 13: Гибридная модель дифференциальной приватности

В гибридной модели данные шифруются дважды, на двух уровнях. Перемешиватель может расшифровать данные только первого слоя. Он содержит в себе идентификаторы пользователей, а также идентификатор группы. Идентификатор группы необходим для общего описания данных, но не их фактического значения. Примером идентификатора группы может служить, например, причина отказа транзакции, а фактические данные - реальные причины.

Затем перемешиватель группирует все идентификаторы групп вместе и подсчитывает, сколько пользователей в каждой группе. Если в груп-

пе достаточно пользователей, все данные передаются анализатору. Затем анализатор может расшифровать второй уровень данных и вычислить соответствующий результат.

Наличие двух слоев позволяет разделить данные и их метаданные. Перемешиватель может оперировать идентификаторами групп, но не фактическими данными. Между тем, хранилище видит данные только партиями и не может знать, кто что отправил. Основной интерес заключается в определении того, что делает перемешиватель. Добавляется некоторая случайность в определенные места процесса, и это гарантирует, что конечный результат будет дифференциально приватным.

Гибридная модель является компромиссом между центральной и локальной моделями. Она позволяет добавить меньше шума, чем локальная модель дифференциальной приватности, но больше, чем центральная.

Рассмотрев архитектурные решения систем, удовлетворяющих свойству дифференциальной приватности, осветим основные протоколы.

4.2 Ключевые протоколы

В этом разделе рассматриваются два ключевых механизма, которые гарантируют дифференциальную приватность. Приведенные механизмы важны, так как они лежат в основе многих более продвинутых дифференциально приватных протоколов. Первый предполагает непрерывное выходное пространство, а суть механизма - добавление шума с распределением Лапласа. Изначально механизм был введен с понятием дифференциальной приватности [18]. Второй механизм включает в себя функцию полезности и применим, когда добавление шума не имеет смысла или уничтожает полезность данных.

4.2.1 Механизм Лапласа

Предполагается, что рассматривается совокупность U строк базы данных. Пусть U^n - элемент базы данных для некоторого $n \in N$. Агрегатор ответит на данный запрос, применив рандомизированный алгоритм A, аргументом которого является любая база данных, а диапазоном - вероят-

ностное пространство Y. То есть для любой базы данных D, результат алгоритма $A(D)$ является случайной величиной над Y. Чтобы зафиксировать идею, стоит подумать о детерминированном алгоритме с двумя входными аргументами, один из которых является случайной величиной.

Мы уже определяли, что дифференциальная приватность - это свойство распределения ответов агрегатора на статистические запросы. Поскольку агрегатор отвечает на статистические запросы, вызывая рандомизированные алгоритмы (рассматриваемые как функция), это выражается как свойство соответствующего случайного отображения.

Случайное отображение $A : U^n \to Y$ называется (ϵ, δ)-дифференциально приватной, если для любого измеримого подмножества $K \subset Y$ и любых смежных баз данных D_1, D_2 выполняется:

$$Pr[A(D_1) \in K] \leq exp(\epsilon) \cdot Pr[A(D_2) \in K] + \delta.$$

В рамках работы рассматривается только $(\epsilon, 0)$-дифференциальная приватность, которую также называют ϵ-дифференциальная приватность. В этом случае дифференциальная приватность сводится к контролю отношений правдоподобия.

Распределение Лапласа с параметром b обозначается через Lap_b и соответствует распределению вероятностей по R, плотность которого равна $x \to \frac{1}{2 \cdot b} \cdot exp(-\frac{|x|}{b})$.

Пусть у нас есть отображение $f : U^n \to R^k$, для некоторого $k \in N$. Теперь посчитаем супремум по двум смежным базам данных D_1 и D_2. Полученное значение имеет название “калибровка шума”.

$$\Delta_f = \sup_{D,D'} ||f(D) - f(D')||$$

Пусть отображение $f : U^n \to R^k$, где $k \in N$. Для некоторого $k \in N$ справедливо, что калибровка шума конечна. Получаем, что для любого $\epsilon > 0$ случайное отображение $A_\epsilon : D \to f(D) + (Y_1, .., Y_k)^T$, где $\{Y_i\}_{i=1}^k$ - независимые одинаково распределённые случайные величины распределения Лапласа $Lap_{\Delta f}/\epsilon$, является ϵ-дифференциально приватным.

Для наглядности представлен график зависимости точности дифференциально приватных данных относительно заданного ϵ (Рис. 14). В качестве исходных данных была использована открытая база данных взрослых людей (Adult Data Set).

Первым делом был обучен обычный классификатор логистической регрессии. С использованием модуля дифференциальной приватности с открытым исходным кодом от Google был обучен созданный ранее классификатор логистической регрессии, который, в свою очередь, удовлетворяет свойству дифференциальной приватности. Для максимальной $I2$ нормы было установлено значение 100. Это значение определяет распределение данных, которые будут защищены в результате применения дифференциальной приватности.

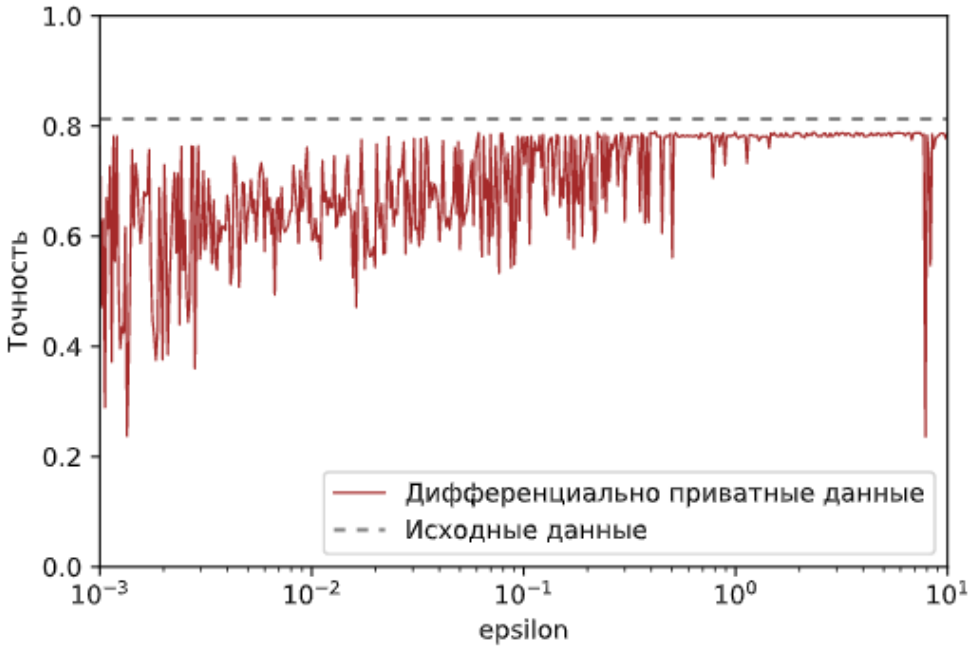

Рис. 14: Точность дифференциально приватных данных

В силу своей простоты именно этот метод может быть использован при реализации центральной модели дифференциальной приватности. Однако, также стоит рассмотреть экспоненциальный механизм, чему и посвящён следующий подраздел.

4.2.2 Экспоненциальный механизм

Как уже говорилось ранее, экспоненциальный метод был разработан для ситуаций, когда добавление шума не имеет смысла или разрушает структуру проблемы. Изначально механизм был представлен в [32].

Ключевой смысл заключается в том, что мы фиксируем дискретный диапазон Y и предполагаем, что у нас есть функция полезности, которая связывает с базой данных D и элементом $y \in Y$ значение $u(D, y) \in R$. Для базы данных D необходимо найти такое значение $y \in Y$, что $u(D, y)$ достаточно близок к $max_{y \in Y} u(D, y)$. Для выявления такого значения необходимо ввести понятие чувствительности.

Пусть u - функция полезности. Тогда следующий супремум по смежным базам данных D_1 и D_2 является чувствительностью функции полезности:

$$\Delta_u = \sup_{y \in Y} \sup_{D, D'} |u(D, y) - u(D', y)|$$

Теперь рассмотрим конечный диапазон Y и функцию полезности u. Для любого $\epsilon > 0$ случайное отображение A_ϵ, которое для данной базы данных D выводит $y \in Y$ с вероятностью, пропорциональной $exp(\epsilon \cdot \frac{u(D,y)}{2\Delta_u})$, является ϵ-дифференциально приватной. Последнее и является экспоненциальным механизмом.

Рассмотрев основные архитектурные решения и ключевые протоколы, хочется отметить, что дифференциальная приватность имеет ряд преимуществ над ранее использовавшимися техниками. Во-первых, это устойчивость к атакам типа связывания. Во-вторых, отсутствие необходимости выявлять идентифицирующие атрибуты в наборе данных, так как любая информация уже является идентифицирующей. И в-третьих, композиционность, что означает, даже при многократном использовании одних и тех же данных, мы способны давать гарантии приватности.

Глава 5. Безопасность для вертикально-интегрированных структур

В последнем издании БСЭ, вертикальная интеграция — это интеграция, объединение в единый технологический процесс всех или основных звеньев производства и обращения, например, от выращивания сельскохозяйственных продуктов до реализации готовой продукции под контролем одного центра — промышленной, банковской или торговой компании. При этом центральная компания контролирует все стадии производства и сбыта.

В современной Британнике, вертикальная интеграция — это форма организации бизнеса, в которой все этапы товарного производства, от приобретения сырья до розничной продажи конечного продукта, контролируются одной компанией. При этом происходит вертикальное слияние бизнеса, то есть компания приобретает либо поставщика, либо клиента.

Главная идея вертикальной интеграции заключается в получении максимального контроля над процессом создания какой-то ценности. При этом отчетливо видны различные сегменты “value chain”, которые расположены на разном расстоянии от клиента. Все наиболее успешные стартапы современной эпохи интернета начинали с самого близкого к клиенту уровня, последовательность развития подобных бизнесов часто похожа:

- стартапы начинают работу в своей небольшой нише, в случае успеха начинают в ней доминировать;
- далее происходит горизонтальная интеграция, когда компании добавляют новые сервисы и продукты (кластеризация);
- с этого момента возникает необходимость вертикальной интеграции, когда уже бывшие стартапы идут “вперед” или “назад” по цепочке создания ценности.

Многие современные ИТ гиганты (Amazon, Facebook, Uber, у нас - Яndex) двигались в этом направлении многими годами. Компания Apple

построила успешный бизнес, используя исключительно удобную интеграцию своих продуктов.

Но это все экономическая теория, и определения вертикальной интеграции говорят в большей степени о слиянии каких-то бизнес-структур в единое целое; для нас же более важным является признак использования единых бизнес-процессов, а в основе - использование единых математических инструментов.

Система BGX представляет собой хороший пример основы для распределенных вычислений, когда заявлены перспективные возможности интеграции в разных направлениях:

- главной целью сети BGX является поддержка транзакций в распределенной среде, под транзакциями понимаются операции с цифровыми активами;
- сеть строится на основе потенциально полнофункциональных объектов (узлов), которые могут действовать как в составе сети, так и независимо;
- каждый узел состоит из набора виртуальных серверов (сервисов), предлагающих необходимую функциональность;
- узлы общаются посредством сообщений, содержащих или транзакции или служебные сообщения;
- узлы используют для расчетов внутренние токены стандарта ERC20;
- для расчетов вне системы (цифровые или фиатные валюты) используется платежный шлюз;
- узлы относительно независимы, однако организованы в структуру на основе дерева Меркла;
- узлы связаны в отдельные кластеры, которые поддерживают внутреннюю адресацию, определяют количество участников кластера;

- консенсус достигается внутри кластера, за исключением случаев отсутствия кворума.

Внутри сети информация организована как набор отдельно сгруппированных данных, разделенных по типам, понятно, что для этого нужно организовать виртуальные подсети однотипных виртуальных серверов. В BGX такие виртуальные подсети называются виртуальными связями (Virtual Links), можно выделить несколько их типов.

Верификация транзакций - транзакции попадают в сеть через единую входную точку API, помещаются в список необработанных транзакций, серверы в несколько заходов верифицируют и подписывают каждую транзакцию. Если в кластере отсутствует кворум, транзакция отправляется в соседний кластер через родительский узел.

Запись транзакций в реестр - если транзакция одобрена, она помещается в реестр и распространяется между нодами по следующему сценарию - когда последний (по порядку одобрения и необходимому количеству согласий) узел успешно верифицировал транзакцию, он пытается записать ее в реестр (DAG) и в случае успеха отправляет измененную запись как параметр функции в родительский узел, а тот в свою очередь рассылает изменения дочерним узлам и своему родительскому узлу.

KYC/AML - система использует модель дифференциальной приватности, это означает, что используются отдельные KYC-серверы, которые поддерживают несколько моделей работы с персональными данными.

На рисунке (Рис. 15) обозначены главные компоненты модели BGX. Чтобы подчеркнуть сложную структуру каждого узла, используется термин Substrate Node. На рисунке отмечены пути прохождения транзакций в соответствии с типом Virtual Link.

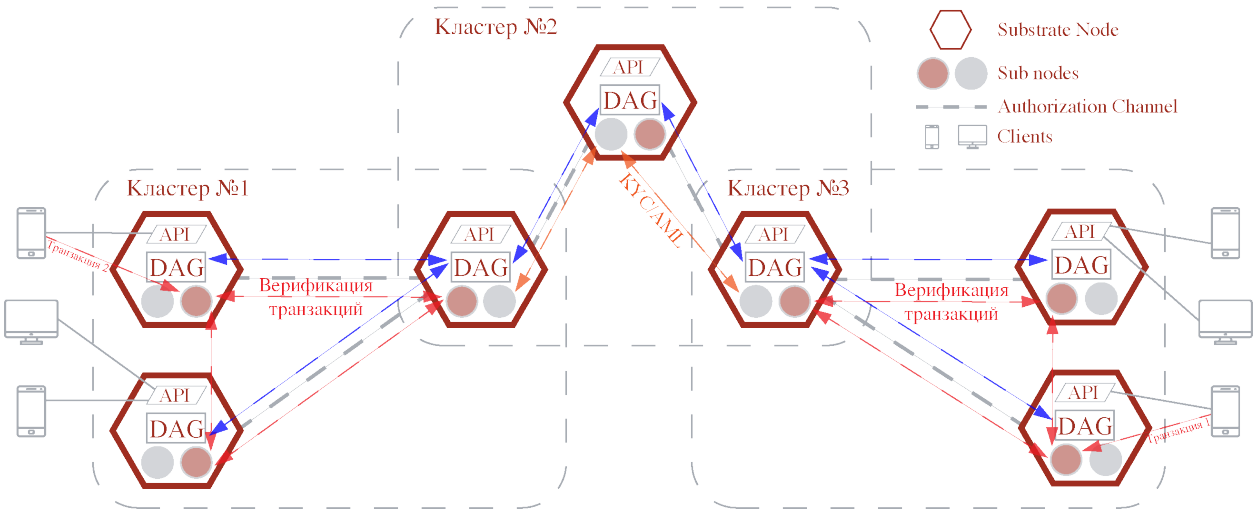

Рис. 15: Модель системы BGX/DGT

Мы можем рассматривать выход за пределы кластера как выход за границы горизонтальной интеграции, и это является большим архитектурным вопросом.

- С точки зрения организации взаимодействия системы с пользовательскими устройствами (смартфонами, компьютерами), решением выглядит создание универсального пользовательского API с использованием подхода RESTful API или через SOA, для идентификации использовать систему публичных ключей.

- Клиентские программы должны поддерживать общие для всех функции API, даже в случае сторонних программ.

- Для соблюдения приватности используется отдельный сервис KYC/AML со сквозным шифрованием.

Как мы видим, с основными сложностями в части конфиденциальности и приватности можно столкнуться при выходе за пределы нашей системы, даже покидая пределы кластера. Сервис KYC/AML со сквозным шифрованием, безусловно, позволяет решать вопросы приватности. При расширении его действия вертикально, на несколько кластеров, и соблюдении общих правил доступа к нашему общему реестру из разных кластеров, не возникает принципиального усложнения концепции защиты приватности.

Защиту конфиденциальности транзакций можно проводить с использованием принципов, уже описанных в данной работе.

Для платежей во “внешнем мире” используется платежный шлюз. В настоящее время существует большое количество шлюзов-посредников для осуществления платежей из разных систем, это направление достаточно развито. Выбор подходящего общего шлюза определяется удобствами сервиса и стоимостью обслуживания, что немаловажно.

Глава 6. Эксперимент

Данный раздел посвящён разработке, тестированию и исследованию системы, которая описана в разделе “Постановка задачи”. Ключевые шаги можно представить следующим образом (Рис. 16).

Рис. 16: Практическая часть

6.1 Организация сети

Создание своей приватной сети Ethereum аналогично созданию альтернативной реальности. В данном случае мы получаем совершенно новый блокчейн, а именно транзакции и блоки (включая первый блок); стоит учитывать, что он не обменивается данными с другими сетями или блокчейнами. Все, что нам необходимо для определения новой сети, – идентификатор сети и файл генезиса.

В действительности никакой узел не сможет подключиться к созданному новому блокчейну, если идентификатор сети, в которой он находится, и генезис не совпадают. Тут хочется остановиться и сделать два важных замечания. Во-первых, корректность идентификатора сети не является достаточным условием (при отличии генезисного блока) для подключения к сети. Во-вторых, не стоит воспринимать созданную сеть как приватную только из-за того, что в ней используется протокол Ethereum; по умолчанию кто угодно с подходящими параметрами может присоединиться к сети. То есть мы никак не защищены от случайных подключений, но, несмотря

на это, специальные инструкции клиента Geth позволяют программно блокировать подобное.

Заслуживает внимания генезисный блок (первый блок в системе) единственный блок, который создается и утверждается человеком, а не машиной. Когда мы создаем или пересоздаем свою цепочку блоков, мы должны создать этот генезисный блок единожды. Для клиента Geth, это будет классический JSON (distr/genesis.json).

Стоит обратить внимание на атрибут генезисного блока «difficulty» (сложность вычислений), значение которого экстремально мало и применимо только для тестовых сетей, в данном случае “0x1” (значение в основной сети - “0x400000000”). Исходя из этого можно сказать, что первый блок будет создан крайне быстро, после чего процесс майнинга стабилизируется в 5-10 секунд для каждого блока. Это позволяет Geth без затруднений работать и проводить расчеты в рамках наших виртуальных машин (ВМ). К слову о минимальных необходимых требованиях, компьютер или ВМ должны иметь не менее 3 ГБ оперативной памяти и 128 МБ видеопамяти для способности майнить.

Итак, с помощью программного обеспечения Docker была создана приватная сеть. После чего был описан Dockerfile. Были созданы и запущены 4 контейнера, основанные на полученном образе. С помощью описанной программы в рамках каждого контейнера были запущены узлы Ethereum, созданы соответствующие аккаунты и запущены узлы Geth. По завершении генерации DAG (5-7 минут для каждого узла), можно приступить к объединению узлов в сеть.

На данный момент узлы Ethereum не знают друг о друге, то есть они работают независимо. Для объединения в сеть, необходимо знать адреса контейнеров в рамках приватной сети. Данные адреса можно получить по идентификатору контейнера. Также написан небольшой скрипт, который делает это в автоматическом порядке (смотри distr/connect.sh).

Сеть готова, перейдём к рассмотрению реализации приватного протокола.

6.2 Приватный протокол

Основная идея, которая лежит в основе приватного протокола или смарт-контракта Ethereum, - поддержка зашифрованных балансов счетов. Перед интеграцией смарт-контракта в сеть необходимо развернуть токен стандарта ERC-20, который условно “присоединен” к смарт-контракту. Это нужно для предоставления пользователю возможности вносить и выводить активы в стандарте ERC-20.

После зачисления средств на собственный счет пользователь может использовать эти средства по назначению, а именно - отправлять на другие аналогичные счета. Преимущество данного контракта в том, что транзакции выполняются конфиденциально (переводимые суммы являются приватными) и анонимно (личности участников транзакций являются конфиденциальными). В данном случае только владелец приватного ключа в праве распоряжаться своими активами. Из вышесказанного следует, что для повышения приватности и анонимности в рамках сети пользователям необходимо совершать больше транзакций, используя приватный протокол.

Учетные записи приватного протокола идентифицируются с помощью открытых ключей Эль-Гамаля, которые хранятся во внутреннем состоянии протокола. Чтобы пополнить счет с открытым ключом Y на X внутренних токенов, необходимо отправить на адрес смарт-контракта X ETH. Внутри протокола X шифруется со случайностью 0, так как X в любом случае является частью транзакции, и добавляется к внутреннему зашифрованному балансу аккаунта, связанного с Y.

Также предлагается конвертировать внутренний токен обрано в ETH, сделав баланс X^* публичным и предоставив доказательство с нулевым разглашением, что ключ Y действительно шифрует X^*.

Для перевода X внутренних токенов с публичного адреса Y на Y', не раскрывая значения X, можно зашифровать X с помощью двух публичных ключей Y и Y'. В тот же момент необходимо предоставить доказательство с нулевым разглашением, что шифрование было выполнено корректно (ключи шифруют одно и то же число), а оставшийся баланс является положительным. В качестве ZK-доказательства используется ги-

бридный алгоритм «Σ-Bullets», который упоминался ранее.

Шифрование Эль-Гамаля - схема шифрования с открытым ключом. Пусть достаточно большое случайное число $x \in Z_p^*$ является приватным ключом, а соответствующее ему значение $y = g^x$ - публичный ключ. Чтобы зашифровать целое число B, его необходимо сопоставить с одним или несколькими элементами группы. Если $b \in Z_p^*$, то простым решением является возведение g в степень b. Теперь шифровка для B имеет вид $(g^b y^r, g^r)$, где $r \in Z_p^*$. Зная x, можно разделить $g^b y^r$ на $(g^r)^x$, чтобы восстановить g^b. Однако для вычисления B необходимо выполнить перебор g^b.

Основное преимущество балансировки экспоненты состоит в том, что шифрование Эль-Гамаля является аддитивно гомоморфным. Если b и b' зашифрованы с использованием одного и того же открытого ключа y для получения зашифрованных текстов $(C_L = g^b y^r, C_R = g^r)$ и $(C'_L = g^{b'} y^{r'}, C'_R = g_{r'})$ соответственно, то $(C_L C'_L = g^{b+b'} y^{r+r'}, C_R C'_R = g^{r+r'})$ - это шифрование $b + b'$ с y.

6.3 Измерения

Реализованный приватный протокол был протестирован в приватной сети Ethereum. В процессе тестирования был произведен ряд измерений. Во-первых, самым важным является исследование стоимости транзакций (в gas) для различных операций протокола, а именно: регистрация пользователя, размещение средств (депозит) и перевод средств от одного аккаунта другому в рамках данного протокола. С результатами можно ознакомиться на гистограмме (Рис. 17). Для сравнения в гистограмму намеренно было включено два дополнительных значения: ETH и ERC-20. Результатом для ETH (стоимость перевода ETH между аккаунтами сети) является значение 21000 gas. На основе стандарта ERC-20 написан простой токен, для которого была переопределена функция перевода средств. Стоимость данной операции соответствует значению 58743 gas.

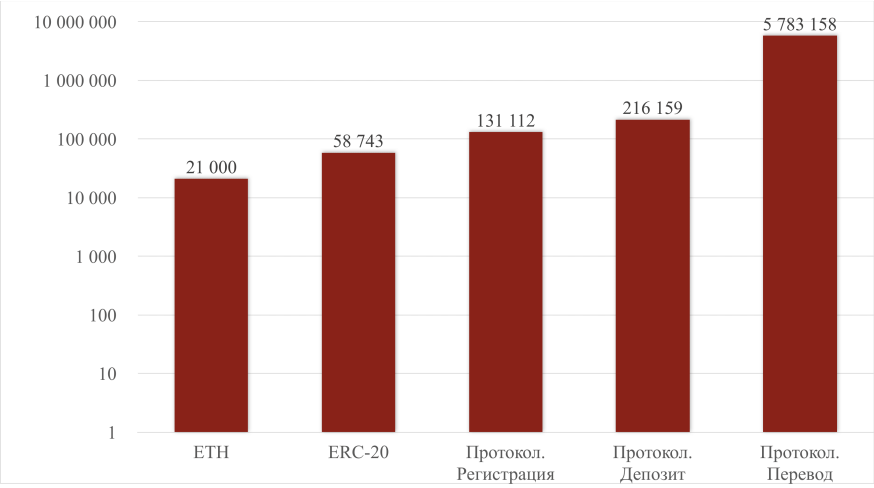

Рис. 17: Стоимость транзакции в gas

Кроме того, на гистограмме (Рис. 18) представлена доля каждой транзакции в одном блоке. Данные измерения актуальны на май 2021 года, когда лимит газа в реальной сети Ethereum составляет 14'990'000 gas.

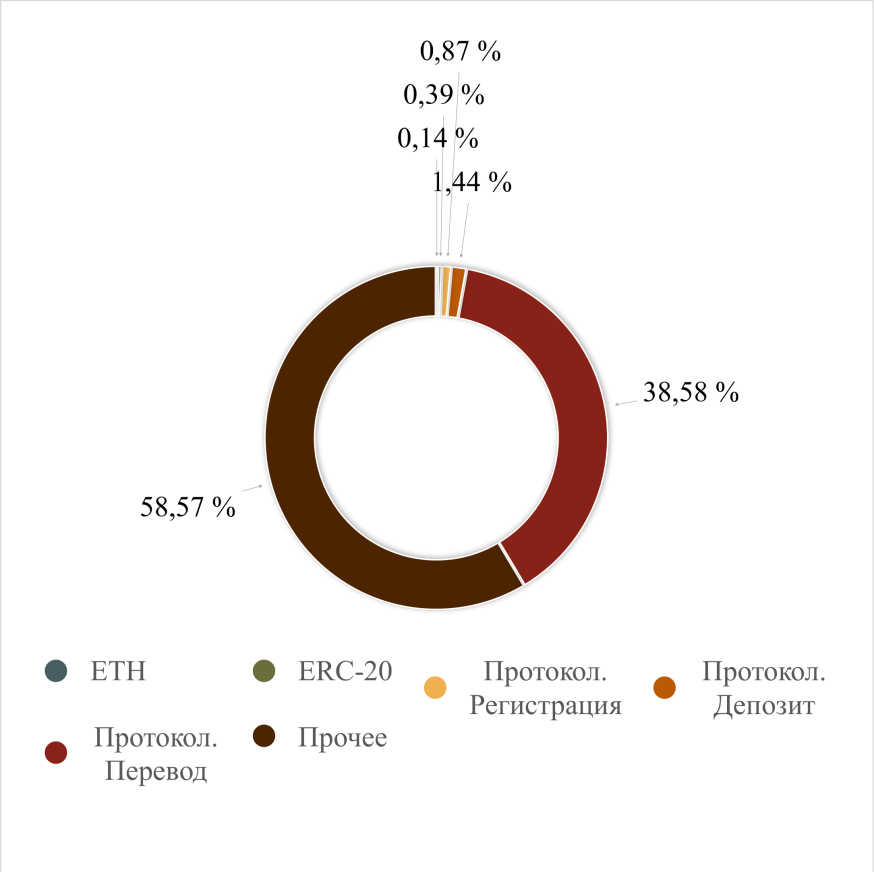

Рис. 18: Доля в блоке на май 2021

Помимо оценки операций в gas и доли транзакции в блоке, интерес-

ным также является исследование скорости операций без накладных расходов сети Ethereum (Рис. 19). На данной гистограмме приведены результаты измерения скорости транзакций в миллисекундах.

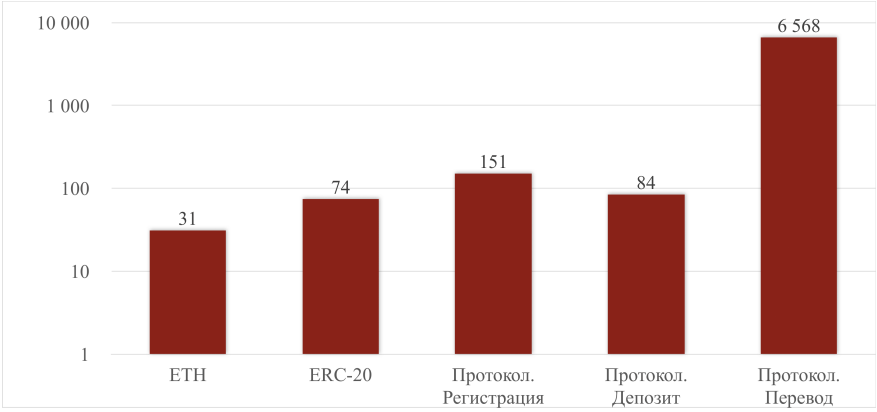

Рис. 19: Скорость без накладных расходов

Заключительное измерение, которое посвящено приватному протоколу, - исследование зависимости скорости исполнения и принятия транзакции перевода в блок от комиссии за выполнение работы. Результаты приведены на графике (Рис. 20).

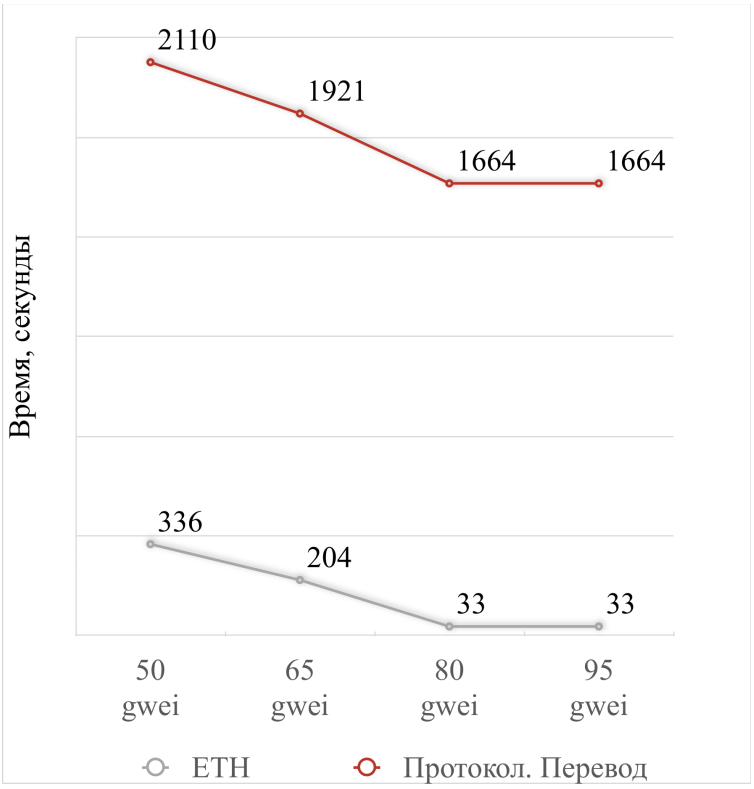

Рис. 20: Время работы в основной сети

Глава 7. Выводы

В ходе работы были рассмотрены ключевые вопросы, касающиеся безопасности, конфиденциальности, приватности и дифференциальной приватности, а также основные механизмы и архитектурные решения, существующие в этой области на настоящий момент. Для реализованного приватного протокола были проведены измерения. Стоит отметить, что стоимость выполнения операций имеет крайне высокую стоимость по отношению к стандартным не конфиденциальным операциям. Дело в том, что приблизительно 90% стоимости операций приватного протокола приходится на операции с эллиптической кривой. Тем не менее, подход имеет неплохие асимптотические показатели и, скорее всего, может найти применение в будущем. В настоящее же время использование такой технологии оправдано только в случае необходимости гарантированной анонимности транзакции. Данный функционал может быть предложен для выполнения по запросу клиента.

Подобное решение может быть расширено путем реализации дополнительного уровня зашумления конфиденциальных данных с использованием центральной модели дифференциальной приватности. Подобная синергия положительно скажется на гибридных федеративных приложениях, которые хранят метаданные клиентов или не бизнес-важные метрики вне блокчейна. Такие данные могут быть представлены третьим лицам для анализа.

Говоря о перспективах использования этого механизма на практике, следует упомянуть такие направления, как платежные шлюзы, электронное голосование, электронные аукционы, а также обеспечивающий конфиденциальность балансов участников механизм консенсуса посредством PoS. Стоит отметить, что решение будет крайне полезно той же сети Ethereum, когда случится переход от алгоритма консенсуса PoW к алгоритму PoS [33].

Глава 8. Заключение

Современные блокчейн системы обещают и стремятся повысить прозрачность распределенных приложений для решения проблемы доверия. В настоящее время прозрачность зачастую достигается за счет значительного снижения уровня конфиденциальности. Результаты, полученные в процессе выполнения выпускной квалификационной работы показывают, что блокчейн системы могут добиться требуемого уровня конфиденциальности и приватности, не теряя при этом функциональности; предложенное решение ограничено лишь накладными расходами.

В ходе практической реализации системы была организована частная приватная сеть узлов, каждый из которых является участником сети Ethereum и имеет открытый API для взаимодействия извне. Для организованной сети Ethereum был представлен приватный и конфиденциальный протокол, который может быть использован в реальной сети в том числе. Было проведено тестирование и произведены измерения.

В результате работы были достигнуты поставленные цели и задачи. В полной мере изложены ключевые и актуальные практики для достижения приватности, конфиденциальности и дифференциальной приватности системы. В рамках работы была предложена гибридная модель системы, которая соответствует описанным выше характеристикам. Программно реализован [34] приватный протокол для частной сети Ethereum, соответствующие измерения представлены.

Список литературы

[1] Fernando Belezas, Manuel Au-Yong-Oliveira, Frederico Branco, Ramiro Gonçalves. "Blockchain in Collaborative Economy business models". IEEE, 14th Iberian Conference on Information Systems and Technologies (CISTI), 2019, pp. 1-7.

[2] The Dai Stablecoin System. Whitepaper, 2017. - URL: https://makerdao.com/whitepaper/DaiDec17WP.pdf

[3] Compound: The Money Market Protocol. Whitepaper, 2019. - URL: https://compound.finance/documents/Compound.Whitepaper.pdf

[4] Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. "Hawk: The blockchain model of cryptography and privacy preserving smart contracts". IEEE, Symposium on Security and Privacy, 2016, pp. 839–858.

[5] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah M. Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden. "A platform for confidentiality-preserving, trustworthy, and performant smart contract execution". ACM, SysTEX '18, Challenges for Combining Smart Contracts with Trusted Computing. 2018, pp. 20-41.

[6] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. "Zerocoin: Anonymous distributed E-cash from Bitcoin". IEEE Symposium on Security and Privacy, pp. 397–411.

[7] George Danezis, Cedric Fournet, Markulf Kohlweiss, and Bryan Parno. "Pinocchio coin: Building zerocoin from a succinct pairing-based proof system". In Proceedings of the First ACM Workshop on Language Support for Privacy-enhancing Technologies, PETShop '13, 2013.

[8] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. "Zerocash: Decentralized anonymous payments from bitcoin". IEEE Symposium on Security and Privacy, 2014, pp. 459–474.

[9] Vitalik Buterin. Thoughts on UTXOs, 2016. - URL: https://medium.com/@ConsenSys/thoughts-on-utxo-by-vitalik-buterin-2bb782c67e53, 2016. Дата обращения: 08.04.2021.

[10] Прасти Нараян. "Блокчейн. Разработка приложений". БХВ-Петербург, 2018.

[11] Wei Cai, Zehua Wang; Jason B. Ernst, Zhen Hong, Chen Feng, Victor C. M. Leung. "Decentralized Applications: The Blockchain-Empowered Software System". IEEE, IEEE Access, 2018, pp. 53019 - 53033.

[12] Равал С. "Децентрализованные приложения. Технология Blockchain в действии". Бестселлеры O'Reilly, 2017.

[13] V. Buterin. On public and private blockchains, 2015. - URL: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/. Дата обращения: 08.04.2021.

[14] EIP-777: ERC777 Token Standard/Ethereum Improvement Proposals, 2021. - URL: https://eips.ethereum.org/EIPS/eip-777. Дата обращения: 08.04.2021.

[15] L. Lamport, R. Shostak, and M. Pease. "The Byzantine generals problem ACM Trans. Program. Lang. Syst., 1982, vol. 4, no. 3, pp. 382401.

[16] IPFS, Protocol Labs, 2021. - URL: https://ipfs.io. Дата обращения: 08.04.2021.

[17] Merkle Distributed Acyclic Graphs (DAGs)/IPFS, Protocol Labs, 2021. - URL: https://docs.ipfs.io/concepts/merkle-dag/#further-resources. Дата обращения: 08.04.2021.

[18] C. Dwork and F. McSherry and K. Nissim, K and 1. Smith. "Calibrating noise to sensitivity in private data analysis". Springer Berlin Heidelberg, Proceedings of Theory of Cryptography Conference, 2006, pp. 265-284.

[19] Garfinkel, Simson, John M. Abowd, and Christian Martindale. "Understanding database reconstruction attacks on public data". Communications of the ACM. 2019, pp. 46-53.

[20] Andrea Gadotti, Florimond Houssiau, Luc Rocher, Benjamin Livshits, Yves-Alexandre de Montjoye. "When the signal is in the noise: exploiting diffix's sticky noise". 28th USENIX Security Symposium (USENIX Security 19). 2019.

[21] Dinur, Irit, and Kobbi Nissim. "Revealing information while preserving privacy". Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. 2003, pp. 202-210.

[22] Sweeney, Latanya. "Simple demographics often identify people uniquely". Health (San Francisco). 2000, pp. 1-34.

[23] L. Sweeney. "k-anonymity: A model for protecting privacy". International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2002, pp. 557-570.

[24] H. Zang and J. Bolot. "Anonymization of location data does not work: A large- scale measurement study". ACM, c, 2011, pp. 145-156.

[25] de Montjoye, Yves-Alexandre, César A. Hidalgo, Michel Verleysen, Vincent D. Blondel. "Unique in the Crowd: The privacy bounds of human mobility". Scientific Reports, 2013, no.1376, p. 1-5.

[26] M. Gymrek, A.L. McGuire, D. Golan, E. Halperin and Y. Erlich. "Identifying personal genomes by surname inference". Science, 339(6117), 2013, pp. 321-324.

[27] A. Narayanan and S. Vitaly Shmatikov. "Robust de-anonymization of large sparse datasets". IEEE, Symposium on Security and Privacy. 2008, pp 111-125.

[28] Dwork, Cynthia, and Aaron Roth. "The algorithmic foundations of differential privacy". Foundations and Trends in Theoretical Computer Science 9(3). 2014, pp. 211-407.

[29] Erlingsson, Úlfar, Vasyl Pihur, and Aleksandra Korolova. "Rappor: Randomized aggregatable privacy-preserving ordinal response". In Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, 2014, pp. 1054-1067.

[30] Apple Inc. "Apple Differential Privacy Technical Overview 2020. - URL: https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf. Дата обращения: 08.04.2021.

[31] Bittau, Andrea, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. "Prochlo: Strong privacy for analytics in the crowd". In Proceedings of the 26th Symposium on Operating Systems Principles. 2017 pp. 441-459.

[32] F. McSherry and K. Talwar. "Mechanism design via differential privacy". Foundations of Computer Science. 2007, pp.94-103.

[33] Vitalik Buterin and Virgil Griffith. “Casper the friendly finality gadget”. Cryptography and Security, 2017.

[34] Ivan Podsevalov. GitHub, 2021. - URL: https://github.com/IvPod/docker-eth