МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

УДК 004.75

РАЗРАБОТКА МЕТОДА СОКРЫТИЯ ПРИВАТНЫХ ДАННЫХ ДЛЯ СИСТЕМЫ ТЕНДЕРОВ НА ОСНОВЕ ТЕХНОЛОГИИ БЛОКЧЕЙН[1]

Д. О. Кондырев

Институт математики им. С. Л. Соболева СО РАН, г. Новосибирск, Россия
Новосибирский государственный университет, г. Новосибирск, Россия
Лаборатория криптографии JetBrains Research, г. Новосибирск, Россия

На основе открытой блокчейн-платформы Ethereum разработана система тендеров, которая позволяет скрывать информацию о заявках на этапе запроса предложений. Создан новый метод, позволяющий решить проблему приватности информации в открытых блокчейн-системах с использованием криптографического протокола доказательства с нулевым разглашением zk-SNARK. Предложенный метод реализован в виде криптографической схемы на основе библиотеки libsnark. Для интеграции криптографической схемы в систему модифицирован Ethereum C++ клиент, куда добавлены новые функции и интерфейс для работы с ними в виде предкомпилированных контрактов.

Ключевые слова: *тендеры, распределенные системы, блокчейн, доказательство с нулевым разглашением, zk-SNARK, платформа Ethereum.*

DOI 10.17223/20710410/48/6

DEVELOPMENT OF A METHOD FOR HIDING PRIVATE DATA FOR A BLOCKCHAIN-BASED TENDER SYSTEM

D. O. Kondyrev

Sobolev Institute of Mathematics, Novosibirsk, Russia
Novosibirsk State University, Novosibirsk, Russia
Laboratory of Cryptography JetBrains Research, Novosibirsk, Russia

E-mail: dkondyrev@gmail.com

A tender system has been developed based on the Ethereum open blockchain platform that allows to hide the information about applications at the request for proposals stage. A new method has been created to solve the problem of information privacy in open blockchain systems using the zk-SNARK, cryptographic zero-knowledge proof protocol. The proposed method has been implemented as a cryptographic scheme based on the libsnark library. To integrate the cryptographic scheme into the system, the Ethereum C++ client has been modified — a new tenderzkp module has been

[1]Работа выполнена при поддержке Математического Центра в Академгородке, соглашение с Министерством науки и высшего образования Российской Федерации № 075-15-2019-1613, и лаборатории криптографии JetBrains Research.

added. It implements functions for creating and verifying zk-SNARK proofs. Interaction with the implemented cryptographic scheme from the smart contract codes is carried out through the new added precompiled contracts. A Solidity library has been created to work with these contracts. The JSON-RPC API of the Ethereum C++ client has been expanded to enable to call methods of the cryptographic scheme from third-party applications.

Keywords: *tenders, distributed systems, blockchain, zero-knowledge proof, zk-SNARK, Ethereum platform.*

Введение

На сегодняшний день большинство конкурсных закупок и электронных торгов проводится через специализированные информационные системы. Для таких систем критичным является вопрос доверия оператору торговой площадки. Участники должны быть уверены в том, что никто не имеет возможности нарушить правила проведения тендера или получить доступ к конфиденциальной информации. В рассмотренных системах вероятность нарушения этих правил не может быть полностью исключена.

Решить проблему доверия при проведении тендеров позволяет блокчейн-технология надёжного распределённого хранения записей о транзакциях. Преимущество этой технологии в том, что она позволяет взаимодействовать участникам напрямую без посредника — оператора площадки. При этом данные хранятся распределённо на узлах блокчейн-сети, история транзакций не может быть изменена или удалена [1 – 3].

Однако при использовании этой технологии данные сохраняются в открытом виде и доступны всем участникам, что не всегда приемлемо при создании промышленных программных систем. В случае с тендерами открытость информации нарушает тайну заявок, которая должна быть сохранена до окончания этапа запроса предложений. Это не позволяет проводить конкурсные закупки в существующих открытых блокчейн-системах.

Целью данной работы является разработка открытой блокчейн-системы для проведения тендеров, которая решила бы проблему приватности информации.

В работе проведён анализ предметной области, представлен краткий обзор технологий, рассмотрены существующие проблемы электронных торговых площадок и предложен новый метод сокрытия приватной информации в открытых блокчейн-системах для реализации конкурсных закупок. Разработанный метод основан на протоколе доказательства с нулевым разглашением zk-SNARK (zero-knowledge Succinct Non-Interactive Argument of Knowledge) и позволяет скрывать конфиденциальную информацию на этапе подачи заявок.

Для реализации предложенного метода модифицирован Ethereum C++ клиент, в который интегрирована разработанная криптографическая схема на основе библиотеки libsnark. Добавлены новые предкомпилированные контракты для работы с криптографической схемой и реализована Solidity-библиотека для работы с ними.

1. Основные требования к системе

Основными принципами процедуры проведения тендеров являются открытость, прозрачность, конкурентность, равенство участников и справедливость [4]. Исходя из этого, можно сформулировать требования, которым должна удовлетворять информационная система тендеров:

T1 **Невозможность изменения информации.** Вся история транзакций в системе должна быть неизменяемой. Участники не должны иметь возможности исправлять данные зарегистрированных заявок, а организатор — изменять правила или результаты после окончания конкурса. Однако после того, как тендер опубликован, часто возникают уточнения или изменения. Все такие правки должны оформляться и регистрироваться в системе в виде отдельных документов.

T2 **Невозможность раннего вскрытия заявок.** Ни у кого из пользователей системы (в том числе и у организатора тендера) не должно быть возможности просматривать данные предложений участников конкурса до завершения периода приема заявок.

T3 **Анонимность заявок.** Участники не должны знать, кто подал заявки на тендер, до его завершения.

T4 **Сокрытие факта подачи заявки.** Факт подачи конкретным пользователем заявки на какой-либо тендер должен быть скрыт от других пользователей системы, поскольку знание этого факта раскрывает информацию о деятельности пользователя и может нарушать принцип конкурентности.

T5 **Запрет подмены пользователя.** Осуществление действий в системе (объявление тендера, подача заявки и др.) от имени другого пользователя должно быть запрещено.

T6 **Открытость информации.** Вся информация должна быть в открытом доступе. Во-первых, это касается информации об объявленных тендерах, все потенциальные участники должны иметь к ней доступ, причём получать его одновременно, чтобы не нарушить принцип честности. Во-вторых, после завершения тендера сторонние наблюдатели должны иметь возможность проверить честность проведения конкурса, поэтому им необходим доступ к результатам, заявкам и всей истории операций.

T7 **Невозможность нарушения сроков.** Заявки не могут быть поданы до начала процедуры подачи заявок и после её окончания.

T8 **Гарантия выполнения правил тендера.** Все правила должны быть чётко зафиксированы и обязательны к исполнению всеми участниками процесса. Недопустимо изменение правил тендера после его объявления.

T9 **Доказательство подачи заявки.** Участники конкурса должны иметь возможность доказать факт подачи своей заявки. При этом ни один пользователь системы не может подделать такое доказательство.

2. Обзор предлагаемого решения

Большинство существующих систем для проведения тендеров имеют общую схему функционирования. Какая-либо организация, выступающая в роли оператора, предоставляет площадку, которой могут пользоваться заинтересованные компании. В этом случае всё взаимодействие участников закупок с площадкой основано на доверии организации-оператору. Подобные платформы не удовлетворяют требованиям открытости и прозрачности и не всегда могут считаться надёжными системами.

В настоящее время ведутся исследования в области децентрализованных систем проведения тендеров. Технология блокчейн позволяет создать площадку, с помощью которой пользователи могут проводить тендеры и заключать договоры напрямую без участия посредника. Вся информация о тендерах хранится на всех узлах блокчейн-сети, что делает систему более отказоустойчивой. Кроме того, любая транзакция, ко-

торая записана в блокчейне, не может быть изменена или удалена. Корректность выполнения правил участниками контролируется смарт-контрактами, что позволяет избавиться от влияния человеческого фактора на результаты тендера (никто не может нарушить процедуру, поскольку все ограничения реализованы в виде программного кода, который не может быть изменён) [5].

Одним из достоинств технологии блокчейн является открытость информации. Любой пользователь всегда имеет возможность просмотреть информацию, хранящуюся в блокчейне, а также проследить всю историю транзакций. Но полная открытость всей информации нарушает требования к организации конкурсных закупок. Процедура проведения тендеров предполагает, что участники не имеют возможности ознакомиться с заявками других претендентов на стадии приёма заявок. Организатор тендера тоже должен получать доступ к заявкам только после того, как завершён их прием. Поэтому открытые блокчейн-системы для проведения тендеров не обеспечивают необходимый уровень приватности информации.

Решением проблемы приватности информации в блокчейн-системе может быть защита данных посредством шифрования. Согласно такому подходу, при подаче заявки участник генерирует симметричный ключ, шифрует информацию о заявке этим ключом и отправляет шифртекст в качестве своей заявки в блокчейн. После окончания срока приёма заявок все участники, отправившие заявки на тендер, должны отправить ключи, которыми эти заявки шифровались. Имея ключ и зашифрованную заявку, любой желающий может проверить корректность данных. Такая система предложена в [6]. Однако такой подход не позволяет проверить корректность зашифрованной заявки в момент её подачи. Ещё одним недостатком является то, что все участники могут наблюдать факт подачи заявки пользователем.

Систему тендеров предлагается реализовать на основе технологии блокчейн, потому что она позволяет обеспечить прозрачность и открытость процедуры проведения конкурса; проверка действий участников может быть реализована в виде смарт-контрактов, которые выступают гарантом выполнения правил.

Необходимо реализовать возможность анонимной подачи заявок на тендеры и добавить сокрытие информации о заявках, при этом обеспечив проверку её корректности. По истечении сроков подачи заявок информация должна раскрываться и сохраняться в открытом виде. Таким образом, вся история будет открытой, что позволит обеспечить прозрачность процедуры тендеров, при этом не будут нарушаться правила конкурсных закупок.

3. Проблема сокрытия информации

На сегодняшний день можно выделить два основных подхода к сокрытию информации о транзакциях в блокчейн-сети [7]:

— механизм смешивания (mixing);
— подход на основе доказательства с нулевым разглашением.

3.1. Механизм смешивания

Протоколы, основанные на данном подходе, принимают разные формы, но все реализуют одну идею. Базовая перемешивающая сеть, также известная как mixnet, является протоколом маршрутизации, в котором сервер принимает в качестве входных сообщения от нескольких отправителей, перемешивает их и отправляет в случайном порядке получателям. Цель такой сети — исключить возможность отследить соответствия между отправителями и получателями транзакций [7].

Такой подход реализуется в CoinShuffle [8], XIM [9], Mixcoin [10] и многих других системах и протоколах. Данный подход обладает рядом недостатков, которые не позволяют применять его в тендерных системах:

1) лишь частично решается проблема анонимности, поскольку не полностью скрывается информация о транзакциях;
2) смешивание применимо только для задачи анонимизации транзакций (позволяет скрыть отправителя); нет возможности расширить алгоритм для более общего случая сокрытия произвольных данных в блокчейн-транзакциях.

3.2. Доказательство с нулевым разглашением

Доказательство с нулевым разглашением — криптографический протокол, в котором принимают участие две стороны — доказывающая и проверяющая (верификатор). Цель протокола заключается в том, чтобы верификатор мог убедиться, что доказывающая сторона обладает знанием секретного параметра. При этом сам секретный параметр не должен раскрываться верификатору или кому-либо ещё [11].

Это может быть представлено в виде программы с двумя входами $C(x, a)$. Вход x является открытым, a — секретный параметр (witness). Выход программы бинарный (`true` либо `false`). Задаётся конкретный общедоступный x. Задача состоит в том, чтобы доказать, что доказывающая сторона знает секретный параметр a, такой, что $C(x, a) = \texttt{true}$.

Доказательство с нулевым разглашением по определению должно удовлетворять следующим трём свойствам:

1) Полнота: если утверждение верно и обе стороны следуют одному и тому же протоколу, то верификатор может убедиться в истинности утверждения.
2) Устойчивость: если утверждение ложно, верификатор с большой вероятностью не будет убеждён в его истинности.
3) Нулевое разглашение: верификатор не получает дополнительной информации.

Концепция интерактивных систем доказательства с нулевым разглашением впервые введена в работе [12]. За годы исследований в области доказательства с нулевым разглашением системы, основанные на этом методе, постепенно улучшались с упором на оптимизацию их эффективности для конкретных приложений. Это привело к появлению алгоритмов, которые существенно сократили количество раундов взаимодействия участников протокола.

Особенности технологии блокчейн, которая взята за основу построения системы тендеров, накладывают ряд ограничений на используемые криптографические протоколы, в частности на доказательство с нулевым разглашением. Поскольку блокчейн является распределённой системой, пользователи могут не быть в сети одновременно. При этом доказательство должно быть доступно всем участникам. После того как доказательство предоставлено, любой пользователь должен иметь возможность проверить его корректность в любой момент времени. Это делает применение интерактивных протоколов доказательства с нулевым разглашением в блокчейн-системах труднореализуемым.

В работе [13] впервые предложен неинтерактивный протокол доказательства с нулевым разглашением. Неинтерактивная система содержит только одно сообщение (доказательство), которое доказывающая сторона отправляет верификатору, т. е. взаимодействие между участниками протокола сводится к одному раунду. Дальнейшие исследования в области неинтерактивных протоколов были направлены на оптимизацию вычислительной эффективности и сокращение размера доказательства.

Существенным прорывом в этом направлении можно считать появление zk-SNARK [14], который сделал возможным эффективное использование неинтерактивных протоколов доказательства с нулевым разглашением в блокчейн-системах.

3.3. Криптографический протокол zk-SNARK

zk-SNARK — это криптографический протокол неинтерактивного доказательства знания с нулевым разглашением [15]. Он позволяет доказывать, что некоторые приватные данные удовлетворяют системе ограничений, выраженной в виде арифметической схемы C, не раскрывая эти данные.

zk-SNARK представляет собой тройку алгоритмов полиномиального времени выполнения (Gen, P, V):

- $Gen(\lambda, C) \to (pk, vk)$. Этот алгоритм принимает в качестве входных данных параметр безопасности λ и арифметическую схему C. На их основе генератор Gen создаёт пару ключей — ключ доказательства (pk, proving key) и ключ верификации (vk, verification key). Оба ключа публикуются как открытые параметры и могут использоваться любое количество раз для создания доказательства и проверки его корректности.
- $P(pk, x, a) \to \pi$. Принимая на вход ключ доказательства pk и любые (x, a), где x — публичные данные, a — секретный параметр, алгоритм P выводит неинтерактивное доказательство π.
- $V(vk, x, \pi) \to b$. Принимая на вход ключ верификации vk, публичные данные x и доказательство π, верификатор V выдаёт $b = 1$, если доказательство является корректным, и 0 иначе.

Данная конструкция удовлетворяет всем требованиям, предъявляемым к алгоритмам доказательства с нулевым разглашением [15].

Преимущество zk-SNARK над другими протоколами доказательства с нулевым разглашением заключается в гарантиях эффективности: длина доказательства зависит только от параметра безопасности, а время проверки не зависит от размера схемы и секретного параметра. Таким образом, zk-SNARK можно рассматривать как неинтерактивный протокол с коротким доказательством и быстрым временем верификации, что делает его наиболее подходящим для использования в блокчейн-системах [16].

3.4. Сокрытие информации в платформе Ethereum

Вся информация в Ethereum-блокчейне хранится в открытом виде, а транзакции не скрывают своих значений. Каждая транзакция содержит адреса аккаунтов отправителя и получателя и передаваемые данные [17]. При этом нет возможности средствами Ethereum скрыть часть полей транзакции (например, нельзя скрыть адрес аккаунта отправителя транзакции).

В zk-SNARK процедура проверки доказательства состоит из операций на эллиптических кривых. В частности, верификатор требует скалярного умножения и сложения на группе точек эллиптических кривых, а также вычислительно более сложной операции — билинейного спаривания.

Ethereum предоставляет реализацию этих операций в виде предварительно скомпилированных контрактов. С их помощью есть возможность реализовать схемы на основе доказательства с нулевым разглашением в коде смарт-контрактов [18, 19].

Сами алгоритмы генерации и верификации доказательства zk-SNARK не реализованы в платформе. В связи с этим возникает ряд проблем при использовании схем на основе zk-SNARK в Ethereum:

1) Нет возможности создавать сложные схемы. Все алгоритмы должны быть реализованы в смарт-контрактах, на размер кода которых накладываются жёсткие ограничения, а криптографические схемы, как правило, требуют большого количества операций.
2) Все алгоритмы приходится реализовывать вручную.
3) Для каждого нового контракта необходимо генерировать отдельные параметры.

3.5. Библиотека libsnark

Libsnark — криптографическая библиотека с открытым исходным кодом, написанная на языке C++, которая обеспечивает эффективные реализации конструкций zk-SNARK [20]. Библиотека является самым быстрым и полным набором доказательств с нулевым разглашением, доступных на данный момент [16].

Для создания криптографических схем zk-SNARK библиотека представляет набор высокоуровневых интерфейсов (gadgetlib1, gadgetlib2 и др.). Эти интерфейсы осуществляют преобразования высокоуровневых спецификаций в арифметические схемы, реализованные в ядре библиотеки. С их помощью можно строить новые криптографические схемы на основе реализованных низкоуровневых примитивов.

4. Архитектура системы проведения тендеров

В данной работе создана система проведения тендеров на основе платформы Ethereum. Архитектура системы состоит из следующих модулей, которые изображены на рис. 1:

— смарт-контракты в Ethereum, обеспечивающие работу с блокчейном;
— модифицированный Ethereum-клиент;
— Java–приложение, предоставляющее высокоуровневый интерфейс для работы с системой.

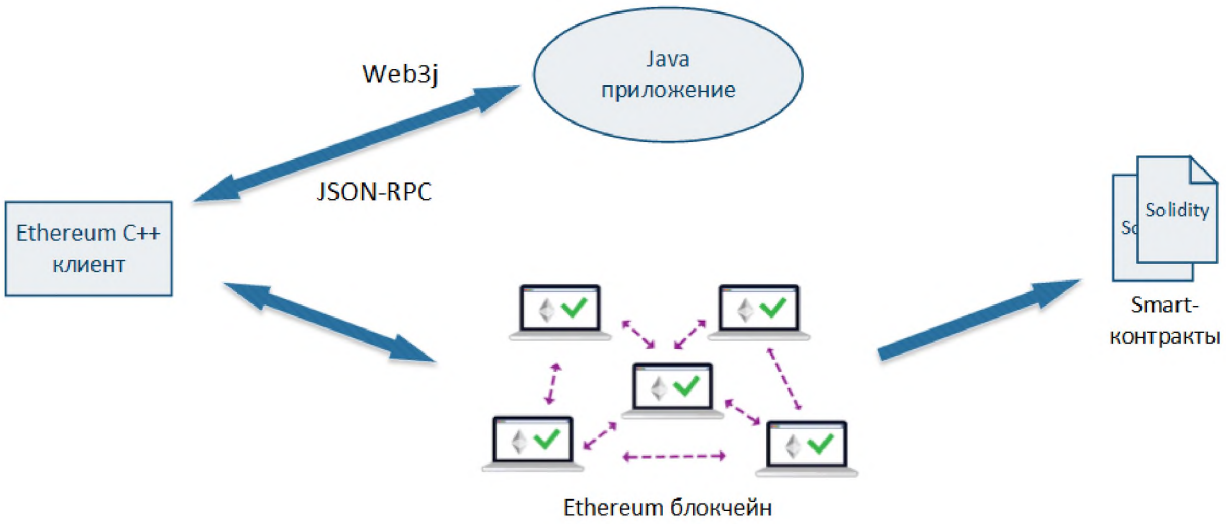

Рис. 1. Схема архитектуры системы

Ethereum-клиент представляет собой реализацию протокола Ethereum. Это программа, которая поддерживает состояние цепочки блоков транзакций и предоставляет API для проведения транзакций и запроса информации о текущем состоянии цепочки блоков. Через него происходит всё взаимодействие с блокчейном, в том числе со смарт-контрактами. Клиент включает в себя реализацию виртуальной машины

Ethereum, которая запускается при выполнении транзакций, взаимодействующих со смарт-контрактами [21]. Ещё одной его задачей является осуществление сетевого взаимодействия с другими клиентами, вместе они образуют единую блокчейн-сеть.

Существует несколько реализаций Ethereum-клиента на различных языках программирования. В разработанной системе используется C++ Ethereum-клиент (aleth), поскольку интегрировать библиотеку libsnark в код этого клиента оказалось наиболее просто и эффективно. При этом система не полагается на особенности реализации клиента aleth. Все необходимые модификации можно внести и в другие Ethereum-клиенты, для чего не потребуется менять архитектуру системы.

Всё взаимодействие между Java-приложением и Ethereum-клиентом (вызов методов смарт-контрактов, прослушивание событий, запрос информации) осуществляется с помощью JSON RPC. JSON-RPC — это легковесный протокол удалённого вызова процедур (RPC) без сохранения состояния [22]. Он использует JSON в качестве формата данных, а в качестве протокола передачи сообщений в системе используется HTTP.

Протокол JSON RPC, реализуемый клиентом Ethereum, является довольно низкоуровневым, и работать с ним напрямую неэффективно. Поэтому для взаимодействия со смарт-контрактами используется Java-библиотека web3j, которая работает поверх JSON-RPC API клиента Ethereum. Web3j позволяет работать с блокчейном без дополнительных накладных расходов на написание собственного интеграционного кода. Библиотека поддерживает все методы JSON-RPC API и может работать с любым клиентом Ethereum, который его реализует [23].

Для более удобного взаимодействия со смарт-контрактами web3j позволяет создать Java-оболочки. На основе кода смарт-контрактов порождаются классы, которые предоставляют функции создания и развертывания смарт-контракта, вызова его функций и выполнения транзакций из Java-кода [24].

Рассмотрим подробно, как устроены отдельные модули.

4.1. Модифицированный Ethereum-клиент

Реализация криптографических протоколов на основе zk-SNARK предполагает генерацию и проверку ограничений, а также выполнение операций над эллиптическими кривыми.

Изначально реализация этих операций была выполнена в виде смарт-контрактов, однако оказалась неэффективной. Код смарт-контрактов хранится в блокчейне, а его выполнение происходит при процессе проверки корректности транзакций в виртуальной машине Ethereum на каждом узле сети, поддерживающем цепочку блоков транзакций. Поэтому к коду смарт-контрактов предъявляются жёсткие требования по эффективности вычисления и размеру. А криптографические операции, необходимые для работы системы, являются вычислительно затратными и при реализации непосредственно в смарт-контрактах сильно увеличивают размер кода. Поэтому было решено реализовать этот протокол на стороне Ethereum-клиента.

4.2. Криптографическая схема

Для реализации алгоритма сокрытия информации о заявках на основе доказательства с нулевым разглашением в Ethereum C++ клиент добавлен отдельный модуль *tenderzkp*. Он построен на базе протокола zk-SNARK с предобработкой (preprocessing zk-SNARK) для NP-полного языка системы ограничений ранга 1 (R1CS — rank-1 constraint systems). Протокол использует эллиптическую кривую Баррето — Наерига. Реализация криптографической схемы предоставлена библиотекой libsnark [20].

Основной интерфейс этого модуля составляют две функции:

— $generate\_proof(proving\_key,\ public\_input,\ private\_input) \to proof$;
— $verify\_proof(verification\_key,\ public\_input,\ proof) \to \{true, false\}$.

Функция генерации доказательства $generate\_proof$ принимает на вход открытые ($public\_input$) и приватные ($private\_input$) данные, а также ключ доказательства ($proving\_key$). Приватными данными для заявки на тендер являются:

— ID участника, подающего заявку;
— ID тендера, на который подается заявка;
— время подачи заявки;
— сумма предложения.

К открытым данным относится информация о тендерах (ID тендеров, время окончания приёма заявок, максимально допустимые суммы предложений), которые проводятся на данный момент, и ID пользователей системы.

Для приватных данных заявки необходимо проверить, что выполнены следующие условия:

— сумма предложения не превышает максимально допустимой для тендера, на который подаётся заявка;
— время подачи заявки не превышает времени окончания приёма заявок на данный тендер;
— участник с таким ID действительно зарегистрирован в системе.

Чтобы создать доказательство с нулевым разглашением, необходимо выразить эти условия в виде ограничений на приватные и открытые входные данные. Для этого создан класс $TenderGadget$, в котором реализована криптографическая схема.

$TenderGadget$ выражает условия корректности заявки с помощью базовых схем библиотеки gadgetlib1. Для этого используются функции $comparison\_gadget$, $conjunction\_gadget$ и $disjunction\_gadget$, реализующие сравнение целочисленных значений (в данном случае это ID тендеров, ID пользователей, суммы предложений и время) и логические операции конъюнкции и дизъюнкции соответственно.

Построенная арифметическая схема преобразуется в более низкоуровневый вид — систему ограничений ранга 1. Полученное R1CS-представление используется в дальнейшем алгоритмами генерации и верификации доказательства libsnark.

На основе R1CS-представления генерируется доказательство утверждения, что входные данные удовлетворяют системе ограничений. Это доказательство является возвращаемым значением функции $generate\_proof$.

Функция проверки доказательства $verify\_proof$ принимает открытые данные ($public\_input$), доказательство, сгенерированное функцией $generate\_proof$ ($proof$), и ключ верификации ($verification\_key$). Она возвращает `true`, если доказательство корректно, и `false` иначе.

Пара ключей (доказательства и верификации), необходимая для работы алгоритмов zk-SNARK, является общей для всей схемы, т. е. для всех контрактов тендеров используются одни и те же ключи. Благодаря этому, есть возможность не хранить пару ключей в смарт-контрактах, а переложить функцию управления ими на Ethereum-клиент, что более эффективно как с точки зрения используемой памяти, так и с точки зрения скорости загрузки. В разработанной системе ключи передаются в Ethereum-клиент в виде конфигурационных файлов и загружаются модулем $tenderzkp$ при вызовах функций $generate\_proof$ и $verify\_proof$.

В разработанной схеме на тестовых данных параметры имеют следующие размеры:

— параметр безопасности λ — 192 байта;
— ключ доказательства pk — 231 кбайт;
— ключ верификации vk — 1 884 байта;
— публичные данные x — 480 байт;
— секретный параметр a — 192 байта;
— доказательство π — 576 байт;
— арифметическая схема C задаётся в коде (класс $TenderGadget$). Её размер в преобразованном для алгоритмов доказательства и верификации виде составляет около 77 кбайт.

4.3. Взаимодействие с криптографической схемой

У смарт-контрактов должна быть возможность взаимодействовать с реализованной криптографической схемой — вызывать функции генерации и проверки доказательства и получать возвращаемые значения. Выполнение кода смарт-контрактов происходит в виртуальной машине Ethereum, поэтому одним из возможных вариантов реализации взаимодействия было бы добавление новых операций в EVM. Но при таком решении необходимо вносить большое количество изменений в платформу Ethereum — не только дополнить набор команд EVM, но и внести соответствующие доработки в компиляторы высокоуровневых языков написания смарт-контрактов (таких как Solidity).

Альтернативным подходом является создание предкомпилированных контрактов. Предкомпилированный контракт — это смарт-контракт, который имеет фиксированный адрес и код которого реализован непосредственно в Ethereum-клиентах. Большая часть криптографических операций в Ethereum (восстановление адреса аккаунта из ECDSA подписи, хеш-функции SHA-256 и RIPEMD-160 и др.) реализована именно в виде предкомпилированных контрактов [25]. Такие контракты являются тестовыми изменениями архитектуры, которые впоследствии могут стать частью протокола Ethereum [26].

В разработанной системе решено использовать второй подход, чтобы минимизировать количество изменений относительно существующих реализаций протокола Ethereum. В Ethereum C++ клиент добавлены новые предкомпилированные контракты с адресами `0x00...09` и `0x00...0a`. При обращении к ним из кода смарт-контрактов вызываются функции $generate\_proof$ и $verify\_proof$ добавленного модуля $tenderzkp$.

Генерация доказательства должна происходить вне блокчейна, так как приватная информация заявки не должна попасть в открытый доступ на данном этапе. Всё взаимодействие с клиентом происходит через JSON-RPC API, поэтому чтобы добавить возможность вызывать методы криптографической схемы из сторонних приложений, добавлены соответствующие интерфейсы в модуль $web3jsonrpc$ Ethereum C++ клиента. На рис. 2 представлена схема всех модификаций, внесённых в Ethereum C++ клиент.

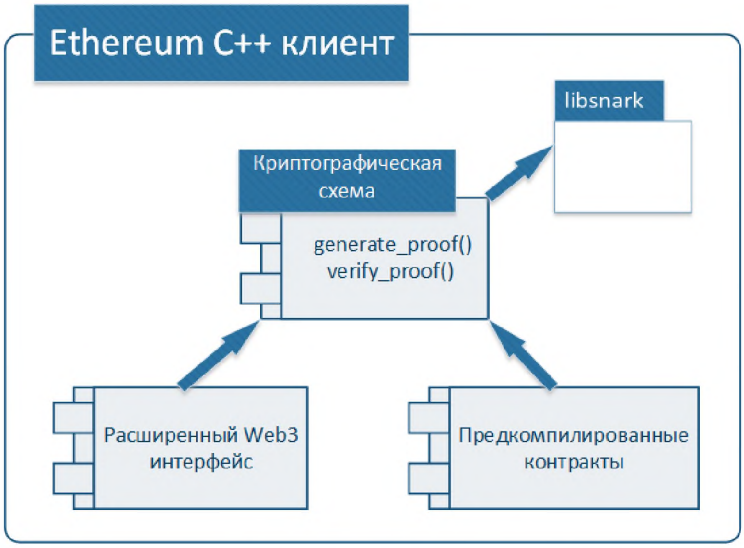

Рис. 2. Модификации Ethereum-клиента, реализованные в системе

4.4. Смарт-контракты

Смарт-контракты — это объекты в блокчейне, которые содержат своё состояние и код функций. Они написаны на статически типизированном высокоуровневом языке программирования Solidity, предоставляемом платформой Ethereum. Общая схема модуля смарт-контрактов изображена на рис. 3.

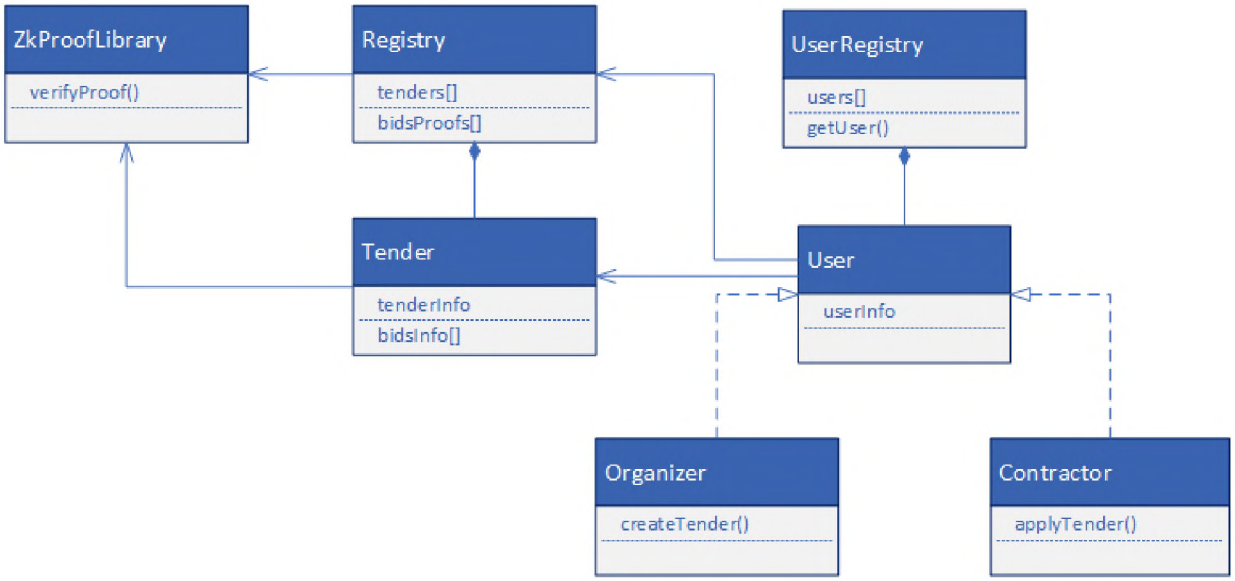

Рис. 3. Схема модуля смарт-контрактов

Основным смарт-контрактом является реестр тендеров (Registry). В нём хранится информация о зарегистрированных тендерах и ссылки на их контракты, а также все закрытые заявки на тендеры (доказательства, полученные алгоритмом криптографической схемы). Через этот контракт проходит регистрация всех тендеров в системе, а также подача закрытых заявок. Закрытая заявка может быть зарегистрирована в реестре только в том случае, если проходит проверка доказательства, реализованная в предкомпилированном контракте.

Для каждого тендера создается отдельный контракт (Tender), в котором сохраняется информация о нём, текущая стадия конкурса, поданные открытые заявки и информация о результатах. Подача открытых заявок происходит через контракт тендера. При этом для каждой открытой заявки происходит проверка корректности — вызывается предкомпилированный контракт, который генерирует доказательство на основе приватных данных, и полученное доказательство сравнивается с ранее зарегистрированным в реестре.

Пользователи работают с контрактами реестра и тендеров не напрямую, вызывая их методы, а через специальные смарт-контракты пользователей (User, Organizer, Contractor). Это позволяет реализовать разграничение прав пользователей (заказчик может объявлять новые тендеры, а участник конкурса может только подавать заявки на существующие). Контракт хранит информацию о пользователе и адрес Ethereum-аккаунта, к которому он привязан. Только транзакции, отправленные от имени этого аккаунта, считаются корректными, поэтому никто, кроме владельца аккаунта, не может совершать действия от имени этого контракта. Кроме того, для каждого заказчика в контракте сохраняется список объявленных им тендеров, а для каждого участника конкурса — все поданные заявки.

Информация обо всех пользователях хранится в реестре пользователей (UserRegistry). Этот контракт регулирует добавление новых пользователей в систему, а также смену Ethereum-аккаунтов, от имени которых работают пользователи.

Стоит отметить, что реализация всех действий пользователей в системе через специальные контракты User вместе с возможностью смены аккаунта позволяют решить одну из фундаментальных проблем блокчейн-систем — потерю приватного ключа аккаунта. Она заключается в том, что отсутствует возможность восстановления приватного ключа аккаунта, и при его потере пользователь не сможет использовать этот аккаунт для дальнейшей работы в блокчейн-системе. В разработанной системе в случае потери ключа есть возможность сменить аккаунт, не изменяя контракт пользователя. При этом информация о пользователе и история его действий в системе (создании тендеров и подаче заявок) сохраняются.

Использование проверок в коде функций смарт-контрактов исключает возможность нарушения участниками правил проведения конкурсных закупок, таких, как:

— объявление тендеров от имени другого пользователя;
— объявление победителем участника, заявка которого не была зарегистрирована, и т.д. [27].

Все проверки, которые осуществляются в смарт-контрактах, гарантированно выполняются, так как блокчейн-транзакции, которые не удовлетворяют условиям в коде проверок, откатываются. Для реализации проверок используется стандартная функция языка Solidity *require*().

Функция *require*() компилируется в набор инструкций, которые осуществляют проверку условия, и инструкцию REVERT (`0xfd`) виртуальной машины Ethereum, к которой переходит управление в случае, когда условие не выполняется. Если во время выполнения кода смарт-контракта в виртуальной машине встречается инструкция REVERT, выполнение кода останавливается, а все изменения, произведенные транзакцией, отменяются. Такой откат всех изменений позволяет сохранить атомарность транзакции. При этом сама транзакция сохраняется в блокчейне.

Контракты являются частью протокола и утверждаются участниками до старта работы системы.

Во время работы системы могут создаваться новые объекты контрактов двух типов — User и Tender. Однако их создание производится не напрямую пользователями, а через контракты UserRegistry и Registry соответственно, т. е. сам код контрактов User и Tender предварительно скомпилирован и интегрирован в код контрактов UserRegistry и Registry. У пользователей отсутствует возможность добавлять свои собственные реализации каких-либо контрактов.

Если в процессе работы обнаруживаются ошибки в смарт-контрактах, они могут быть исправлены в новых версиях этих же контрактов. Переход на новую версию осуществляется только в случае одобрения изменений участниками системы.

Взаимодействие с предкомпилированными смарт-контрактами не может быть реализовано средствами языка Solidity, для вызова кода функций таких контрактов используются ассемблерные вставки. Для удобства работы создана Solidity-библиотека. Она инкапсулирует низкоуровневое взаимодействие с предкомпилированными контрактами и предоставляет интерфейс для работы с ними в виде Solidity-функций.

5. Алгоритм работы системы

Рассмотрим более подробно алгоритм работы системы. Можно выделить следующие этапы проведения тендера:

1) Заказчик создаёт контракт Tender, в котором размещает всю необходимую информацию о проводимом конкурсе.
2) Пользователи подают скрытые заявки в общий реестр тендеров.
3) После окончания срока приёма заявок пользователи вскрывают заявки (отправляют открытую информацию в контракт Tender со ссылкой на скрытую заявку). Все заявки, которые не были вскрыты, аннулируются.
4) После окончания срока предоставления открытой информации заказчик оценивает заявки и определяет победителя.

Отдельные крупные этапы данного алгоритма — подача скрытой заявки и раскрытие информации.

5.1. Процесс подачи скрытой заявки

Процесс подачи скрытой заявки, схематично представленный на рис. 4, проходит следующим образом:

1) Пользователь создаёт новый анонимный Ethereum-аккаунт.
2) После этого пользователь формирует заявку на выбранный тендер, которая содержит необходимые приватные данные (ID участника, ID тендера, текущее время, сумму предложения).
3) На основе приватных данных заявки создаётся публичное доказательство. Для этого через JSON-RPC API модифицированного Ethereum-клиента вызывается функция генерации доказательства криптографической схемы доказательства с нулевым разглашением.
4) Далее от имени анонимного аккаунта пользователь отправляет доказательство в контракт Registry.
5) Смарт-контракт Registry осуществляет проверку корректности доказательства, вызывая код предкомпилированного контракта верификации. Если проверка пройдена успешно, заявка записывается в хранилище контракта Registry. В противном случае заявка считается некорректной и отклоняется.

Благодаря тому, что для каждой подачи заявки генерируется новый аккаунт, нельзя отследить, кто именно записывает публичное доказательство. Это обеспечивает со-

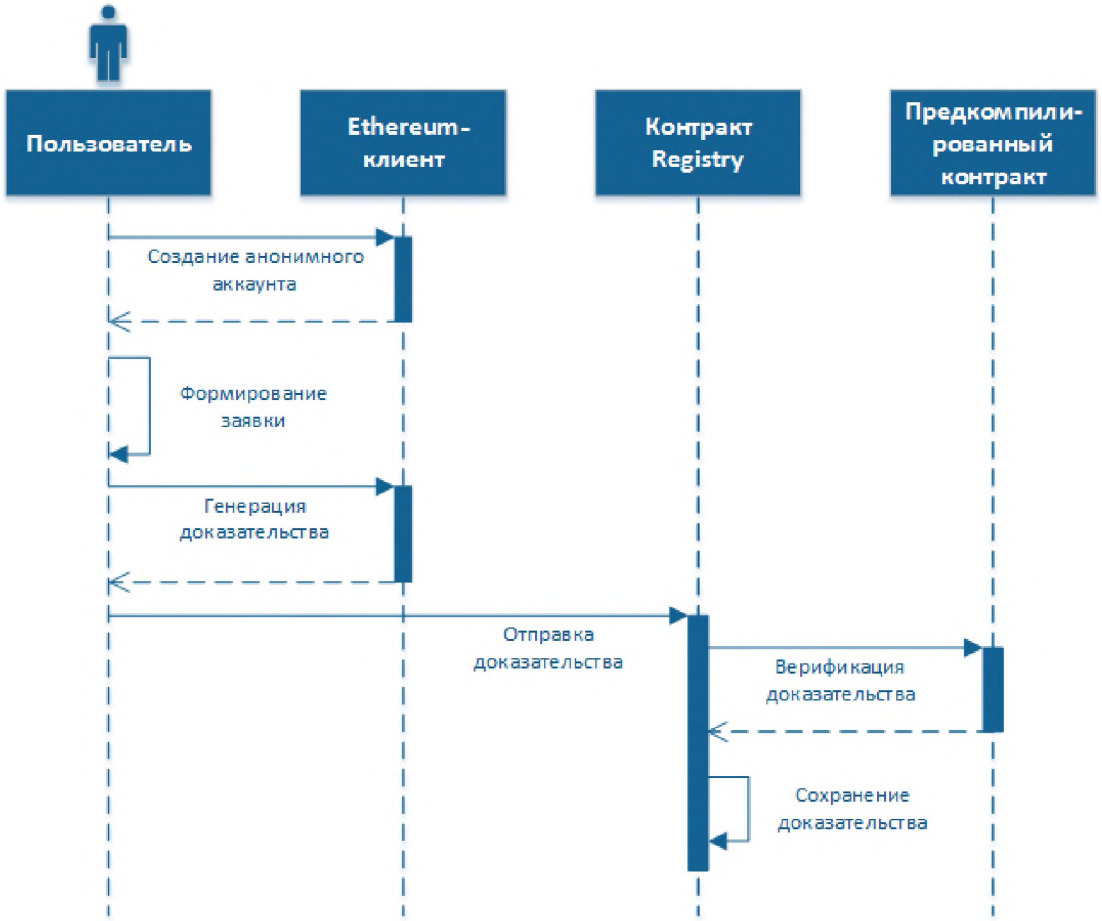

Рис. 4. UML-диаграмма последовательности процесса подачи скрытой заявки

крытие не только приватных данных заявки, но и самого факта подачи пользователем заявки на конкретный тендер.

Стоит отметить, что первые три шага описанного алгоритма выполняются вне блокчейна, поэтому секретная информация не видна никому, кроме самого пользователя.

5.2. Процесс раскрытия заявки

После окончания срока приёма заявок начинается этап раскрытия информации (рис. 5). На этом этапе:

1) Пользователь отправляет раскрытые данные в контракт Tender, указывая ссылку на закрытую заявку.
2) Контракт тендера запрашивает из контракта Registry зарегистрированное доказательство.
3) Далее вызывается предкомпилированный контракт, который на основе раскрытых приватных данных генерирует доказательство.
4) После этого контракт Tender производит ряд проверок:
   — Предоставленная информация соответствует ранее зарегистрированной в реестре закрытой заявке. Для этого производится сравнение сгенерированного доказательство с сохранённым в реестре. Их совпадение свидетельствует о том, что доказательства сгенерированы на основе одних и тех же приватных данных и заявка является подлинной. Если они не сов-

падают, это означает, что доказательство, соответствующее подаваемой открытой заявке, не было зарегистрировано на этапе запроса предложений, и такая заявка, согласно правилам проведения тендеров, не может принимать участие в конкурсе.
- ID пользователя в заявке совпадает с ID пользователя, который отправляет данные.
- ID тендера в заявке совпадает с ID тендера контракта.

5) При успешном прохождении проверок предоставленная информация сохраняется в контракте.

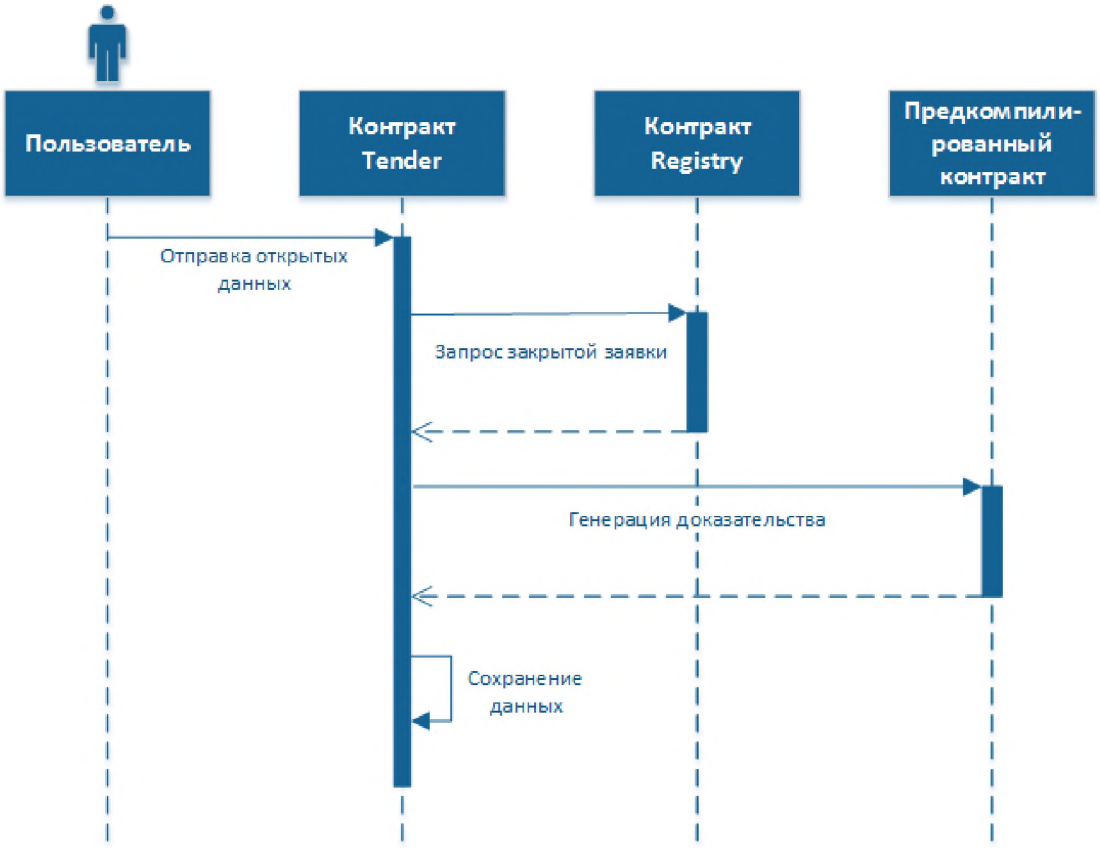

Рис. 5. UML-диаграмма последовательности процесса раскрытия заявки

Стоит отметить, что при раскрытии заявок пользователь производит действия в блокчейн-сети от имени своего аккаунта. Это позволяет удостовериться в личности пользователя и исключить возможность подачи заявки от лица другого участника.

Алгоритм работы системы, в основе которого лежит протокол доказательства с нулевым разглашением, кроме возможности проверять корректность информации при подаче заявки, даёт ещё несколько преимуществ.

Во-первых, уже на этапе подачи заявок можно собирать статистику, которая может быть использована для различных целей.

Во-вторых, при раскрытии заявок можно предоставлять в открытом виде только некоторые данные, гарантируя корректность всей остальной информации.

В-третьих, есть возможность реализовать алгоритм предоставления информации только отдельным участникам на этапе вскрытия заявок. Такой алгоритм может быть применён в системах закрытых тендеров, где информация о раскрытых заявках долж-

на быть доступна только организатору закупки. При этом за счёт использования алгоритмов доказательства с нулевым разглашением все остальные участники смогут проверить корректность поданных заявок и удостовериться, что победитель тендера выбран согласно правилам, не получая доступ к самой информации заявок.

Таким образом, реализованный алгоритм даёт более богатые возможности для расширения функциональности системы.

6. Развёртывание системы

Развёртывание любой программной системы, созданной на базе платформы Ethereum, может быть осуществлено двумя основными способами:

— в основной блокчейн-сети Ethereum (Ethereum Mainnet);
— в собственной блокчейн-сети.

Разработанный метод сокрытия приватной информации полагается на модификации, внесенные в Ethereum-клиент, а внедрение модифицированных Ethereum-клиентов в основную сеть невозможно из-за несовместимости протокола с обычными узлами. Вызов предкомпилированных контрактов будет невозможен на обычных узлах, а механизм консенсуса полагается на то, что все операции должны одинаково выполняться на всех узлах блокчейн-сети. Поэтому для системы проведения тендеров необходимо развернуть собственную блокчейн-сеть, состоящую из узлов, поддерживающих внесённые модификации. Это могут быть модифицированные Ethereum C++ клиенты, которые описаны ранее, либо любые другие клиенты, в которых добавлены предкомпилированные контракты с теми же адресами и реализующие описанную криптографическую схему доказательства с нулевым разглашением.

Поскольку предполагается использовать отдельную блокчейн-сеть, механизм оплаты действий в системе может регулироваться сообществом участников путём определения своих правил поверх существующей стандартной схемы оплаты транзакций в Ethereum. Например, взимание комиссии можно полностью отменить. Кроме того, в отдельной блокчейн-сети количество транзакций должно быть значительно меньше, чем в основной сети Ethereum, что увеличивает пропускную способность системы.

Для работы криптографической схемы необходима пара ключей (доказательства и верификации). Их генерация должна выполняться доверенной стороной. Получившиеся в результате открытые параметры публикуются и становятся доступными для всех сторон. В разработанной системе ключи передаются в качестве конфигурационных файлов при старте Ethereum-клиента. Процесс генерации ключей выполняется только один раз, после этого доверенная сторона не требуется.

Эта фаза является критической с точки зрения безопасности системы. Любой, кто обладает параметром безопасности, на основе которого сгенерированы ключи, получит возможность генерировать ложные доказательства, которые будут приняты алгоритмом верификации как корректные. Поэтому при внедрении системы процедуре генерации стоит уделить особое внимание. Как правило, используются многосторонние протоколы для безопасной генерации параметров, которые дают возможность не полагаться на честность единственного участника. В процессе инициализации параметров участвует ряд сторон, использующих многосторонний протокол для генерации ключей доказательства и верификации. Для обеспечения надёжности созданной криптографической схемы достаточно, чтобы хотя бы одна из сторон была честна. Распределённый протокол генерации параметров для zk-SNARK приведён в [16].

Заключение

Предложена и реализована система тендеров, которая удовлетворяет критериям безопасности, открытости и конфиденциальности. Вопрос доверия решён с помощью технологии блокчейн, а сокрытие приватной информации — с помощью алгоритмов доказательства с нулевым разглашением.

Разработан принципиально новый метод, позволяющий решить проблему приватности информации в блокчейн-системах с использованием алгоритмов доказательства с нулевым разглашением. Метод позволяет участникам зафиксировать факт подачи заявки на тендер, не раскрывая её содержания.

Предложенный и реализованный метод может быть использован не только для тендеров, но и в других системах, где есть необходимость скрывать часть информации в открытой блокчейн-сети. Он расширяет область применения технологии блокчейн в промышленных программных комплексах.

ЛИТЕРАТУРА

1. *Wattenhofer R.* The Science of the Blockchain. 1st ed. Inverted Forest Publishing, 2016. 115 p.
2. *Nakamoto S.* Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`
3. *Равал С.* Децентрализованные приложения. Технология Blockchain в действии. СПб.: Питер, 2017. 192 с.
4. *Кузнецов К. В.* Конкурентные закупки: торги, тендеры, конкурсы. СПб.: Питер, 2005. 368 с.
5. *Кондырев Д. О., Бобров В. С., Ефремов И. Е., Власов В. Н.* Система проведения тендеров на основе платформы Ethereum // Вестник НГУ. Сер. Информационные технологии. 2017. Т. 15. № 3. С. 31–39.
6. *Hardwick F. S., Akram R. N., and Markantonakis K.* Fair and transparent blockchain based tendering framework — A step towards open governance // IEEE Intern. Conf. TrustCom/BigDataSE, New York, USA, 2018. P. 1342–1347.
7. *Heilman E., Baldimtsi F., and Goldberg S.* Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions // Intern. Conf. Financial Cryptography and Data Security. Springer, 2016. P. 43–60.
8. *Ruffing T., Moreno-Sanchez P., and Kate A.* Coinshuffle: practical decentralized coin mixing for bitcoin // ESORICS 2014. LNCS. 2014. V. 8713. P. 345–364.
9. *Bissias G., Ozisik A. P., Levine B. N., and Liberatore M.* Sybil-resistant mixingfor bitcoin // Proc. WPES'14. Scottsdale, Arizona, USA, November 2014. P. 149–158.
10. *Bonneau J., Narayanan A., Miller A., et al.* Mixcoin: anonymity for bitcoin with accountable mixes // Intern. Conf. Financial Cryptography and Data Security. Springer, 2014. P. 486–504.
11. *Шнайер Б.* Прикладная криптография: протоколы, алгоритмы и исходные коды на языке С. 2-е изд. СПб.: ООО «Альфа-книга», 2018. 1040 с.
12. *Goldwasser S., Micali S., and Rackof C.* The knowledge complexity of interactive proof systems // STOC'85. Proc. 17th Ann. ACM Symp. Theory of Computing. Providence, Rhode Island, USA, 1985. P. 291–304.
13. *Blum M., Feldman P., and Micali S.* Non-interactive zero-knowledge proof systems and applications // STOC'88. Proc. 20th Ann. ACM Symp. Theory of Computing. Chicago, USA, 1988. P. 103–112.
14. *Ben-Sasson E., Chiesa A., Genkin D., et al.* SNARKs for C: Verifying program executions succinctly and in zero knowledge // CRYPTO'2013. LNCS. 2013. V. 8043. P. 90–108.

15. *Ben-Sasson E., Chiesa A., Garman C., et al.* Zerocash: Decentralized anonymous payments from bitcoin // IEEE Symp. Security and Privacy. San Jose, USA, 2014. P. 459–474.
16. *Virza M.* On Deploying Succinct Zero-Knowledge Proofs. PhD Thesis. Massachusetts Institute of Technology, 2017. 131 p.
17. `http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf` — Ethereum White Paper.
18. *Galal H. S. and Youssef A. M.* Verifable sealed-bid auction on the Ethereum blockchain // Intern. Conf. Financial Cryptography and Data Security. Springer, 2018. P. 265–278.
19. *Eberhardt J. and Tai S.* ZoKrates — scalable privacy-preserving off-chain computations // IEEE Intern. Conf. Blockchain. Halifax, Canada, 2018. P. 1084–1091.
20. `https://github.com/scipr-lab/libsnark` — libsnark: a C++ library for zkSNARK proofs.
21. `http://ethdocs.org/en/latest` — Ethereum Homestead Documentation.
22. `https://www.jsonrpc.org/specification` — JSON-RPC 2.0 Specification.
23. *Svensson C.* Blockchain: Using cryptocurrency with Java // Java Magazine. 2017. January/February. P. 36–46.
24. `https://docs.web3j.io/index.html` — Web3j documentation.
25. `https://solidity.readthedocs.io/en/v0.4.24` — Solidity documentation.
26. *Wood G.* Ethereum: A Secure Decentralised Generalised Transaction Ledger. `http://gavwood.com/Paper.pdf`
27. *Кондырев Д. О.* Разработка системы проведения тендеров на основе платформы Ethereum // Материалы 55-й Междунар. научн. студ. конф. МНСК-2017. Информационные технологии. Новосибирск, Новосибирский государственный университет, 2017. С. 53.

REFERENCES

1. *Wattenhofer R.* The Science of the Blockchain. 1st ed. Inverted Forest Publishing, 2016. 115 p.
2. *Nakamoto S.* Bitcoin: A Peer-to-Peer Electronic Cash System. `https://bitcoin.org/bitcoin.pdf`
3. *Raval S.* Decentralized Applications. Harnessing Bitcoin's Blockchain Technology. O'Reilly, 2016. 118 p.
4. *Kuznetsov K. V.* Konkurentnye zakupki: torgi, tendery, konkursy [Competitive Procurement: Bidding, Tendering, Contests]. St. Petersburg, Piter Publ., 2005. 368 p. (in Russian)
5. *Kondyrev D. O., Bobrov V. S., Efremov I. E., and Vlasov V. N.* Sistema provedeniya tenderov na osnove platformy Ethereum [Ethereum-Based Tender System]. Vestnik NSU, Ser. Information Technologies, 2017, vol. 15, no. 3, pp. 31–39. (in Russian)
6. *Hardwick F. S., Akram R. N., and Markantonakis K.* Fair and transparent blockchain based tendering framework — A step towards open governance. IEEE Intern. Conf. TrustCom/BigDataSE, New York, USA, 2018, pp. 1342–1347.
7. *Heilman E., Baldimtsi F., and Goldberg S.* Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. Intern. Conf. Financial Cryptography and Data Security, Springer, 2016, pp. 43–60.
8. *Ruffing T., Moreno-Sanchez P., and Kate A.* Coinshuffle: practical decentralized coin mixing for bitcoin. ESORICS 2014, LNCS, 2014, vol. 8713, pp. 345–364.
9. *Bissias G., Ozisik A. P., Levine B. N., and Liberatore M.* Sybil-resistant mixingfor bitcoin. Proc. WPES'14, Scottsdale, Arizona, USA, November 2014, pp. 149–158.
10. *Bonneau J., Narayanan A., Miller A., et al.* Mixcoin: anonymity for bitcoin with accountable mixes. Intern. Conf. Financial Cryptography and Data Security, Springer, 2014, pp. 486–504.

11. *Schneier B.* Applied Cryptography. Protocols, Algorithms, and Source Code in C. John Wiley & Sons, 1996. 784 p.
12. *Goldwasser S., Micali S., and Rackof C.* The knowledge complexity of interactive proof systems. STOC'85. Proc. 17th Ann. ACM Symp. Theory of Computing, Providence, Rhode Island, USA, 1985, pp. 291–304.
13. *Blum M., Feldman P., and Micali S.* Non-interactive zero-knowledge proof systems and applications. STOC'88. Proc. 20th Ann. ACM Symp. Theory of Computing, Chicago, USA, 1988, pp. 103–112.
14. *Ben-Sasson E., Chiesa A., Genkin D., et al.* SNARKs for C: Verifying program executions succinctly and in zero knowledge. CRYPTO'2013, LNCS, 2013, vol. 8043, pp. 90–108.
15. *Ben-Sasson E., Chiesa A., Garman C., et al.* Zerocash: Decentralized anonymous payments from bitcoin. IEEE Symp. Security and Privacy, San Jose, USA, 2014, pp. 459–474.
16. *Virza M.* On Deploying Succinct Zero-Knowledge Proofs. PhD Thesis. Massachusetts Institute of Technology, 2017. 131 p.
17. `http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf` — Ethereum White Paper.
18. *Galal H. S. and Youssef A. M.* Verifable sealed-bid auction on the Ethereum blockchain. Intern. Conf. Financial Cryptography and Data Security, Springer, 2018, pp. 265–278.
19. *Eberhardt J. and Tai S.* ZoKrates — scalable privacy-preserving off-chain computations. IEEE Intern. Conf. Blockchain, Halifax, Canada, 2018, pp. 1084–1091.
20. `https://github.com/scipr-lab/libsnark` — libsnark: a C++ library for zkSNARK proofs.
21. `http://ethdocs.org/en/latest` — Ethereum Homestead Documentation.
22. `https://www.jsonrpc.org/specification` — JSON-RPC 2.0 Specification.
23. *Svensson C.* Blockchain: Using cryptocurrency with Java. Java Magazine, 2017, January/February, pp. 36–46.
24. `https://docs.web3j.io/index.html` — Web3j documentation.
25. `https://solidity.readthedocs.io/en/v0.4.24` — Solidity documentation.
26. *Wood G.* Ethereum: A Secure Decentralised Generalised Transaction Ledger. `http://gavwood.com/Paper.pdf`
27. *Kondyrev D. O.* Razrabotka sistemy provedeniya tenderov na osnove platformy Ethereum [Development of a tender system based on the Ethereum platform]. Proc. MNSK-2017, Information Technology, Novosibirsk, NSU Publ., 2017, p. 53. (in Russian)