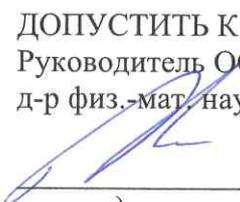


Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Научно-образовательный центр «Высшая ИТ школа»

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
д-р физ.-мат. наук, профессор


О.А. Змеев

подпись

« 11 » июня 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РЕАЛИЗАЦИЯ CROSS-CHAIN МОСТА МЕЖДУ СЕТЯМИ ETHEREUM И BNB CHAIN

по направлению подготовки 09.03.04 Программная инженерия
направленность (профиль) «Программная инженерия»

Хрущев Павел Евгеньевич

Руководитель ВКР
канд. физ.-мат. наук,
ассистент


К. С. Ким

подпись

« 10 » июня 2022 г.

Консультант ВКР
ассистент НОЦ "Высшая ИТ школа"

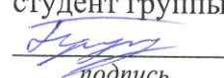

Д. С. Красавин

подпись

« 09 » июня 2022 г.

Автор работы

студент группы № 971704


П. Е. Хрущев

подпись

« 08 » июня 2022 г.

УТВЕРЖДАЮ
руководитель ООП
д-р. физ.-мат. наук, профессор
О.А.Змеев
« 20 » февраль 2022 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы бакалавра обучающегося
Хрущев Павел Евгеньевич

(Ф.И.О. обучающегося)

по направлению подготовки Программная инженерия, направленность «Программная инженерия»

1. Тема выпускной квалификационной работы бакалавра
Реализация Cross-Chain моста между сетями Ethereum и BNB Chain

2. Срок сдачи обучающимся выполненной выпускной квалификационной работы:
а) в учебный офис – « 09 » июня 2022 г.
б) в ГЭК – « 14 » июня 2022 г.

3. Исходные данные к работе:

Цель: Разработка контрактов токенов и Cross-Chain моста между ними в сетях Ethereum

и BNB Chain. Задачи: Исследование стандартов токенов, разработка токенов согласно

установленным стандартам, рассмотрение преимуществ целевых блокчейн сетей,
разработка контракта моста, тестирование в среде разработки, развертывание

контрактов в тестовых сетях целевых блокчейн сетей и проведение тестирования.

Организация, по тематике которой выполняется работа

Общество с ограниченной ответственностью «КРИПТОН СТУДИО»

Руководитель выпускной квалификационной работы

канд. физ.-мат. наук, ассистент ТГУ
(должность, место работы)

 / К.С. Ким
(подпись) (И.О. Фамилия)

Консультант выпускной квалификационной работы

ассистент НОЦ «Высшая ИТ Школа»
(должность, место работы)

 / Д.С. Красавин
(подпись) (И.О. Фамилия)

Задание принял к исполнению

« 18 » 02 20 22 г
(дата)

 / П.Е. Хрущев
(подпись) (И.О. Фамилия)

АННОТАЦИЯ

Выпускная квалификационная работа 59 стр., 21 рис., 11 лист., 1 таблица, 17 источников.

СМАРТ-КОНТРАКТ, CROSS-CHAIN МОСТ, ETHEREUM, BNB CHAIN, PROOF-OF-WORK, PROOF-OF-STAKE, ERC-20, ERC-721, SOLIDITY

Цель работы — Разработка смарт-контрактов токенов и Cross-Chain моста между ними в сетях Ethereum и BNB Chain.

Результат работы — Смарт-контракты токенов и моста разработаны и развернуты в тестовых сетях целевых блокчейн сетей.

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ	3
ВВЕДЕНИЕ	5
1 ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ	7
2 ЦЕЛИ И ЗАДАЧИ	9
3 ОПИСАНИЕ ЦЕЛЕВЫХ БЛОКЧЕЙН СЕТЕЙ И ИХ СРАВНЕНИЕ	11
3.1 Описание сети Ethereum.....	11
3.2 Разбор сети BNB Chain (Binance Smart Chain).....	13
3.3 Сравнение сетей Ethereum и BNB Chain.....	15
4 ИЗУЧЕНИЕ АЛГОРИТМОВ КОНСЕНСУСА И ИХ СРАВНЕНИЕ	19
4.1 Разбор алгоритма консенсуса PoW(Proof-of-Work)	20
4.2 Разбор алгоритма консенсуса PoS(Proof-of-Stake)	23
4.3 Сравнение алгоритмов консенсусов PoW и PoS.....	26
5 ОПИСАНИЕ СТАНДАРТОВ ТОКЕНОВ ERC	28
5.1 Описание стандарта токена ERC-20	29
5.2 Описание стандарта токена ERC-721	33
6 РАЗРАБОТКА СМАРТ-КОНТРАКТА МОСТА.....	38
6.1 Проектирование смарт-контрактов.....	38
6.2 Разработка смарт-контрактов токенов	43
6.3 Разработка смарт-контрактов моста	47
7 АНАЛИЗ ИНСТРУМЕНТОВ И ТЕСТИРОВАНИЕ	51
8 РЕЗУЛЬТАТЫ.....	53
ЗАКЛЮЧЕНИЕ	57
СПИСОК ЛИТЕРАТУРЫ.....	58
ПРИЛОЖЕНИЕ А	60

ГЛОССАРИЙ

1. Блокчейн (Blockchain) — растущий список записей, содержащих информацию и называемых блоками, которые связаны друг с другом при помощи криптографии. Каждый последующий блок в данном списке содержит криптографический хэш предыдущего блока, метку времени и данные транзакции
2. Смарт-контракт (Smart Contract) – компьютерный алгоритм или протокол транзакции, предназначенный для автоматического выполнения, контроля и/или документирования юридически значимых событий и действий в соответствии с условиями контракта или соглашения.
3. Транзакция – это подписанный пакет данных, предназначенный для передачи криптоактивов между двумя адресами, либо любой другой полезной информации.
4. Децентрализованное приложение (Decentralized application, dApp) – приложение, которое имеет возможность работать автономно, зачастую с использованием смарт-контрактов, размещенных в блокчейн сети.
5. Деплой смарт-контракта (deploy, также развертывание) – Отправка в блокчейн сеть скомпилированного кода смарт-контракта без указания получателя.
6. EIP (Ethereum Improvement Proposal) – стандарты, определяющие потенциально новые функции или процессы для Ethereum. EIP содержат технические спецификации для предлагаемых изменений и служат «источником правды» (“source of truth”) для сообщества. Обновления сети и стандарты приложений для Ethereum обсуждаются и разрабатываются в процессе EIP. Как только этот EIP будет одобрен комитетом и доработан, он станет ERC
7. ERCs (Ethereum Request for Comments) – технические документы, написанные разработчиками Ethereum для сообщества Ethereum.

Каждый такой документ содержит набор правил, необходимых для внедрения токенов в экосистему Ethereum.

8. NFT (non-fungible token, также невзаимозаменяемый токен) – вид криптографических токенов, каждый экземпляр которых является уникальным, не может быть замещен или обменян подобным или аналогичным экземпляром.
9. Виртуальная машина Ethereum (Ethereum Virtual Machine, EVM) – это программная среда для выполнения смарт-контрактов в системе Ethereum.
10. Нативная валюта – цифровой актив, неразрывно связанный с блокчейном и являющийся его частью.
11. Алгоритм консенсуса блокчейна – метод защиты, при котором узлы сети блокчейна достигают согласия в вопросах добавления новых блоков.
12. Токен – единица учета, используемая как показатель цифрового баланса в некотором активе.
13. Форк – изменение протокола работы блокчейна, по которому происходит признание подлинности блока.

ВВЕДЕНИЕ

В данной выпускной квалификационной работе (ВКР) рассматривается решение задач безопасной передачи данных между двумя блокчейн сетями Ethereum и BNB Chain посредством создания децентрализованного приложения (dApp) моста при помощи криптографической подписи с открытым и закрытым ключами.

Cross-Chain мост представляет из себя в данном случае смарт-контракты мостов, расположенные в целевых блокчейн сетях и связанные между собой посредством бэкенд сервиса.

Актуальность данной работы вытекает из двух факторов. Во-первых, в текущее время возрастает количество блокчейн сетей, которые, в следствии своего устройства, не имеют прямого доступа друг к другу, таким образом создавая необходимость в посреднике, частью которого и является продукт данной работы. Во-вторых, при достаточном росте токена возникает необходимость расширения на другую блокчейн сеть. Второй фактор также является наиболее важным, так как количество токенов уверенно возрастает с интересом обывателей, что можно наблюдать в последнее время, а также ввиду коммерческой цели создания большинства существующих токенов.

Немаловажную роль в существовании моста играет фактор доверия. Смарт-контракты в сети блокчейн не подвергаются вмешательству со стороны их владельцев, если на подобные действия нет прописанных заранее условий в самом смарт-контракте. Таким образом, любой пользователь может быть уверен в том или ином развитии событий, согласно содержанию смарт-контракта.

В итоге, смарт-контракты, в рамках одной блокчейн сети, могут взаимодействовать друг с другом, образуя сложные системы, в которых устройство функций всегда доступно пользователю. Пользователь может найти целевые контракты и, при минимальном ознакомлении с кодом, вынести для себя окончательное решение о необходимости вызова той или иной функции.

В случае с бэкенд сервисом, который также является частью моста, пользователь не может быть уверен в его механизмах работы, таким образом разработчик не может использовать чужой сервис для налаживания работы своих смарт-контрактов в разных сетях. Возникает необходимость создания собственного моста, включая бэкенд сервис и смарт-контракты моста для двух блокчейн сетей. Таким образом, актуальность разработки смарт-контрактов данного типа весьма велика.

В дипломной работе подробно описываются стандарты токенов, наиболее широко распространенных в сети, а также архитектура моста, созданного для взаимодействия между токенами смарт-контрактов, расположенных на двух блокчейн сетях.

1 ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ

Для того чтобы сформулировать проблему и задачу, необходимо иметь представление о предметной области.

Блокчейн в более простой интерпретации – это децентрализованный цифровой реестр транзакций. Его принципиальное отличие от базы данных в том, что возможно только внесение в него информации, изменение или удаление которой невозможно. Таким образом, информация, попавшая в блокчейн, остается в нем в течении всего его жизненного цикла.

В большинстве случаев, в EVM совместимых сетях транзакции имеют три типа, отвечающих целям:

- 1) Деплой смарт-контракта в сети. Такие контракты имеют нулевой целевой адрес и полезная информация, отправленная в транзакции, воспринимается как байт-код EVM.
- 2) Обращение к уже существующему в сети смарт-контракту. В таких случаях полезная информация в транзакции воспринимается как входные данные для целевого смарт-контракта.
- 3) Отправка нативной валюты. Именно эта цель является наиболее значимой в виду принадлежности нативной валюты блокчейн сети.

После создания транзакции пользователем, она отправляется к ближайшему узлу сети, который рассылает ее по всем узлам сети, где транзакция проверяется на корректность, а также проверяется наличие у инициировавшего транзакцию пользователя достаточных средств для ее отправки. После этого происходит верификация транзакции, где несколько проверенных транзакций собираются в блок, который впоследствии присоединяется к блокчейну.

Посредством включения новых транзакций, список блоков постоянно растет, таким образом накапливая и информацию в нем. Любой включенный в сеть блок можно всегда просмотреть, сверить информацию, но изменить, либо удалить ее нельзя.

Следует учитывать, что для изменения какой-либо информации в блокчейне, необходимо признание больше половины владельцев узлов данного блокчейна. Таким образом, хотя изменение и возможно, но крайне маловероятно для малозначительного вмешательства, когда события не затрагивают весь блокчейн в целом.

Стоит также объяснить, что такое блокчейн с физической точки зрения. Как было упомянуто ранее, блокчейн состоит из множества узлов сети, также называемых нодами, которые являются цифровыми устройствами, имеющими операционную систему, подключение к сети, а также вычислительную мощность для взаимодействия с ней. И каждый из узлов блокчейна хранит в себе копию истории транзакций, по которой сверяются другие узлы сети.

Возможны события, когда необходимо изменить правила работы сети. В таких случаях происходит форк сети, который подразделяется на софтфорк - в случае, когда узлы старой версии сети могут взаимодействовать с узлами новой, а также хардфорк – когда нововведения настолько серьезны, что узлы новой версии более не способны взаимодействовать с узлами старой версии.

Узлы в сети блокчейн также могут иметь разную роль, в первую очередь в зависимости от типа алгоритма консенсуса сети. И особенно это касается узлов сети, отвечающих за добавление новых блоков в блокчейн. В случае, когда сеть имеет тип алгоритма PoW (Proof of Work) – данные узлы являются майнерами, а в случае алгоритма PoS (Proof of Stake) – валидаторами, но в обоих случаях узлы берут определенную «плату» за работу, которая измеряется в газе – единице измерения работы транзакций, которая имеет определенный курс относительно нативной валюты в тот или иной момент времени.

2 ЦЕЛИ И ЗАДАЧИ

Исходя из введения, были сформированы цели и задачи.

Цель: разработка смарт-контрактов токенов и Cross-Chain моста между ними в сетях Ethereum и BNB Chain.

Задачи:

1. Исследование стандартов токенов.
2. Рассмотрение преимуществ целевых блокчейн сетей, а также различий в их работе.
3. Проектирование архитектуры смарт-контрактов.
4. Разработка токенов согласно установленным стандартам.
5. Разработка смарт-контрактов моста согласно требованиям
6. Тестирование смарт-контрактов в среде разработки
7. Развертывание смарт-контрактов в тестовых сетях целевых блокчейн сетей и проведение тестирования.

Функциональное требование:

1. Смарт-контракт моста, как и смарт-контракты токенов должны реализовывать ограничение в доступе к функциям относительно согласованных ролей.

Нефункциональные требования:

1. Смарт-контракты должны быть разработаны на языке Solidity
2. Разрабатываемые токены должны придерживаться стандартов ERC-20 [1] и ERC-721 [2], включая оригинальные сигнатуры функций.
3. Тестирование смарт-контрактов должно быть произведено на базе среды разработки hardhat с применением необходимых библиотек и расширений для среды разработки
4. Файлы скриптов развертывания смарт-контрактов должны также содержать команды верификации смарт-контрактов спустя

определенное количество времени после успешного развертывания смарт-контрактов в блокчейн сети.

5. Развертывание контрактов окончательно должно быть произведено в двух тестовых сетях BSC testnet и Rinkeby testnet, используемых для проверки смарт-контрактов перед их деплоем в реальную сеть.

3 ОПИСАНИЕ ЦЕЛЕВЫХ БЛОКЧЕЙН СЕТЕЙ И ИХ СРАВНЕНИЕ

3.1 Описание сети Ethereum

Блокчейн сеть Ethereum является первой сетью, являющейся платформой для создания децентрализованных онлайн сервисов, которые работают на базе смарт-контрактов. [3]

С момента предложения, Ethereum позиционировался не столько как платежная система, опыт которой можно четко наблюдать в лице блокчейн сети Bitcoin, сколько как база для доступного внедрения самой технологии блокчейн в сторонние проекты.

Концепт Ethereum был предложен Виталием Бутеркиным в конце 2013 года в одной из его публикаций. После этого, в начале 2014 года в Швейцарии была основана компания Ethereum Switzerland GmbH, и в апреле этого же года Ethereum был формально описан Гэвином Вудом в так называемой «Желтой книге».

После проведения краунфандингового сбора средств на разработку в виде первого публичного предложения во второй половине 2014 года, генезис блок Ethereum, также являющийся первым в цепочке блоков, был сформирован 20 июля 2015 года.

По заверениям разработчиков Ethereum, алгоритм консенсуса PoW, используемый с самого начала разработки, планировался к использованию лишь на начальном этапе, с дальнейшим переходом на алгоритм консенсуса PoS, что было подтверждено последующими обновлениями сети для подготовки к этому переходу.

Нативной валютой Ethereum является эфир (Ether), обозначаемый греческой буквой Кси (Ξ). Эфир имеет дробные части, которые имеют названия finney ($1/1000$ Eth), szabo ($1/10^6$ Eth), gwei ($1/10^9$ Eth) и wei ($1/10^{18}$ Eth). Wei также является минимальной единицей при расчете эфира.

С точки зрения экономического смысла, Ethereum позволяет регистрировать любые сделки, не прибегая к традиционным юридическим

процедурам. Для получения юридической значимости таких сделок требуется соответствия смарт-контрактов законам государства, посредством добавления условий и ограничений, которые установлены законодательством.

Ethereum является вторым популярным блокчейном (после Bitcoin), что стал известен благодаря своему подходу «Всемирный программируемый блокчейн», который позиционирует его как электронную программируемую сеть со множеством приложений. Именно развитие Ethereum подтолкнуло криптовалютную индустрию к новому витку развития.

3.2 Разбор сети BNB Chain (Binance Smart Chain)

Для начала стоит отметить, что помимо Binance Smart Chain, второй целевой блокчейн сети, существует Binance Chain. Данная блокчейн-платформа была разработана централизованной блокчейн биржей Binance и запущена в 2019 году для поддержки децентрализованной биржи «Binance DEX». Платформа хорошо оптимизирована для мгновенной торговли, но не имеет достаточной функциональности для использования широкого спектра децентрализованных приложений. [4][5]

Ввиду этого, в сентябре 2020 года был запущен Binance Smart Chain – блокчейн-сервис, позволяющий реализацию децентрализованных приложений при помощи смарт-контрактов. Данный блокчейн совместим с EVM, что позволяет использовать смарт-контракты из Ethereum, обеспечивая при этом сравнимую с первой блокчейн-платформой скорость транзакций.

Кроме того, Binance Smart Chain является форком Go Ethereum – одной из трех реализаций протокола Ethereum.

Стоит отметить, что данный блокчейн-сервис не является расширенной версией Binance Chain, его боковой веткой или форком. Binance Smart Chain – это полностью автономный и отдельный блокчейн, разработанный для параллельной работы с Binance Chain, но может существовать и отдельно от него в случае неполадок оригинального блокчейна.

Чжао Чанпэн, основатель Binance, объяснял в октябре 2010 года, что компания рассматривает Binance Smart Chain как возможный способ для преодоления различий между централизованными и децентрализованными финансами.

Нативная валюта в данных сетях называется BNB (Binance Coin). Ввиду того, что Binance Chain и Binance Smart Chain разработаны для параллельной работы и имеют кросс-чейн совместимость относительно Binance, это позволяет быстро перемещать средства между этими блокчейнами. Таким образом, данная разработка позволяет использовать преимущества быстрой

торговли одного блокчейна с функциональностью смарт-контрактов другого блокчейна.

В феврале 2022 года стало известно о ребрендинге децентрализованной экосистемы блокчейна Binance. И Binance Chain, и Binance Smart Chain стали BNB Chain. Также количество набора валидаторов было расширено с 21 до 41 узла.

С точки зрения защиты своих блоков BNB Chain использует алгоритм консенсуса PoSA (Proof-of-Staked Authority) – гибридный алгоритм консенсусов PoS и PoA (Proof-of-Authority). Данный алгоритм в BNB Chain ограничивает количество узлов, которые отвечают за составление и проверку блоков до 41 узла, в зависимости от количества поставленной суммы нативной валюты BNB, поочередно давая этим узлам создавать блоки и получать вознаграждение в виде комиссии. Список из 41 узла меняется каждый день.

Обобщая вышеизложенное, можно сказать, что BNB Chain ориентирована на разработчиков децентрализованных приложений. Включая тех, кто только рассматривает преимущества существующих блокчейн сетей для развития проекта, так и тех, кто ищет возможность переместить проект на менее затратную блокчейн сеть.

3.3 Сравнение сетей Ethereum и BNB Chain

После рассмотрения обоих блокчейнов, можно сказать, что они во многом похожи, так как оба поддерживают работу децентрализованных приложений на EVM. Кроме того, адреса Ethereum и BNB Chain кошельков пользователей являются идентичными.

Если начать рассматривать преимущества и недостатки сетей можно сразу отметить, что BNB Chain выигрывает относительно Ethereum за счет своих дешевых операций, что привлекает большой процент разработчиков проектов в данную сеть. Но это также означает, что, в сравнении с Ethereum, операции в данной сети рассматриваются гораздо менее серьезно, ввиду доступности.

Время подтверждения также является преимуществом BNB Chain, так как его среднее значение, исходя из данных bscscan составляет 3 секунды (Рис. 1), в то время как в Ethereum это время составляет 13 секунд, что можно увидеть исходя из данных etherscan (Рис. 2).

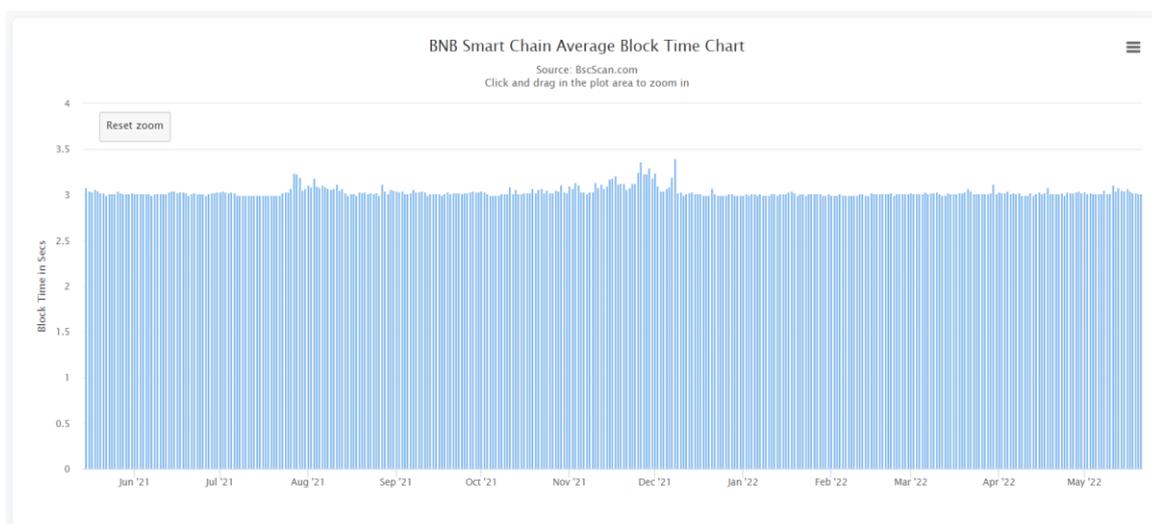


Рисунок 1. Диаграмма среднего количества времени подтверждения блоков в сети BSC (BscScan).

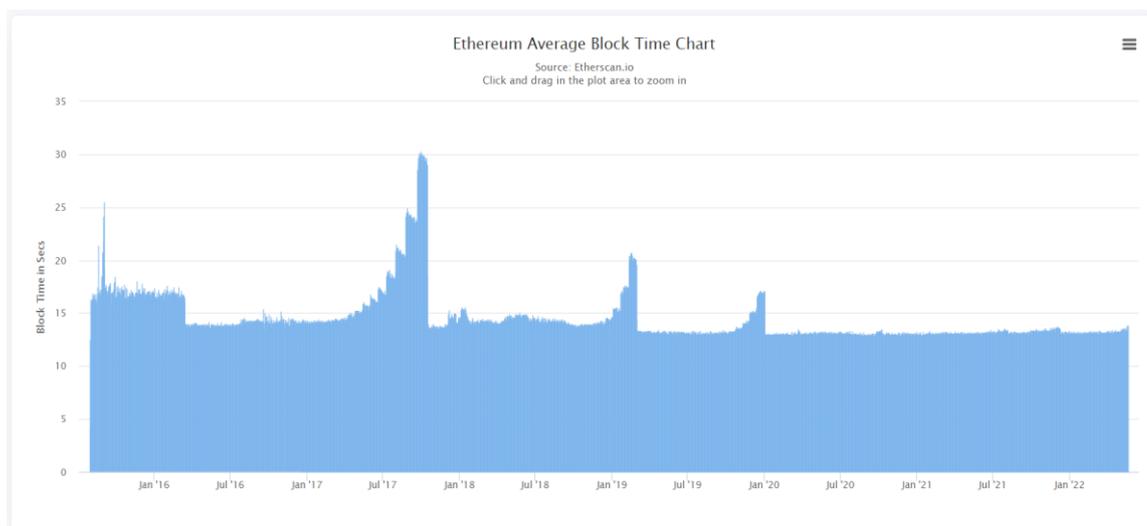


Рисунок 2. Диаграмма среднего количества времени подтверждения блоков в сети Ethereum (Etherscan).

Так как операции в BNB Chain гораздо более дешевы, и при этом имеют большую скорость, пропускная способность сети также является выше, как и количество транзакций, обработанных за определенный промежуток времени. По данным на май 2022 года для BNB Chain данная характеристика является равной 32.4 TPS(Transactions-per-seconds) [6], в то время как для Ethereum эта же характеристика является равной 12.6 TPS [7](По данным bscscan.com и etherscan.com). Но ввиду того, что относительно популярные сервисы и приложения имеют чрезвычайно большое количество транзакций за день, искусственно увеличивая объем транзакций, сложно точно проанализировать, сколько из данных транзакций являются реально значимыми.

Следующим стоит рассмотреть общее количество децентрализованных приложений на обоих блокчейнах. В данном вопросе Ethereum выигрывает ввиду роли первопроходца блокчейна, позволяющего использовать смарт-контракты. Если посмотреть на статистику июня 2021 года, на блокчейне Ethereum размещается более 2800 децентрализованных приложений, тогда как в BSC только в районе 810 [8]. Таким образом, несмотря на рост BNB Chain, лидерство в количестве децентрализованных приложений остается за

Ethereum. Следовательно, разработчики, ориентирующиеся на опыт предыдущих приложений, посмотрят в первую очередь на Ethereum.

Немаловажную роль в сравнении блокчейн сетей играет общий объем торгов. Согласно данным от февраля 2021 года криптоаналитической платформы DappRadar за один месяц до этого Ethereum обработал транзакций в общей сумме более 112 миллиардов долларов, в то время как за этот же промежуток времени BNB Chain обработал транзакций только на 15 миллиардов долларов (Рис. 3).[9]

Ethereum Vs. Binance Smart Chain						
	Addresses (M)	Jan Tx Vol (\$B)	TVL (\$B)	Daily Tx (M)	Gas Price (Gwei)	Nodes
ETH	139.7	112	54	1.3	272	10,000
BSC	5.2	15	10	2.6	16	21
	ETH=30x	ETH=7x	ETH=5x	BSC=2x	ETH=17x	ETH=500x

Sources: The Defiant with data from Etherscan, BscScan, Dapper Labs, The Block, Binance Chain Docs, Ethernodes.org

Рисунок 3. Сравнение характеристик блокчейн сетей Ethereum и BNB Chain на февраль 2021 года (DappRadar).

Последней, но не менее важной характеристикой является децентрализованность. Несмотря на то, что BNB Chain был запущен гораздо позднее, чем Ethereum, количество уникальных адресов на май 2022 года по данным etherscan и bscscan лишь с небольшим перевесом склоняется к Ethereum (198,6 миллионов уникальных адресов против 163,5 миллионов) (Рис. 4 и Рис. 5). Но это не отменяет того факта, что большая часть нативной валюты сконцентрирована у адресов, которые являются валидаторами в блокчейн сети из-за устройства алгоритма консенсуса. И даже эти адреса, по большей части принадлежат самой бирже Binance, значительно пошатывая устойчивость к цензуре в глазах пользователей.

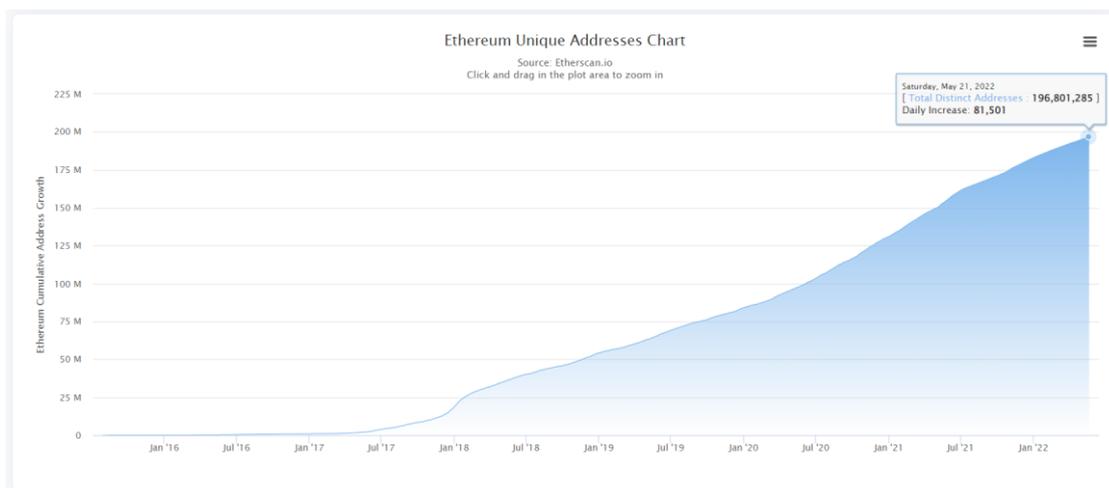


Рисунок 4. Диаграмма количества уникальных адресов в сети Ethereum (Etherscan).

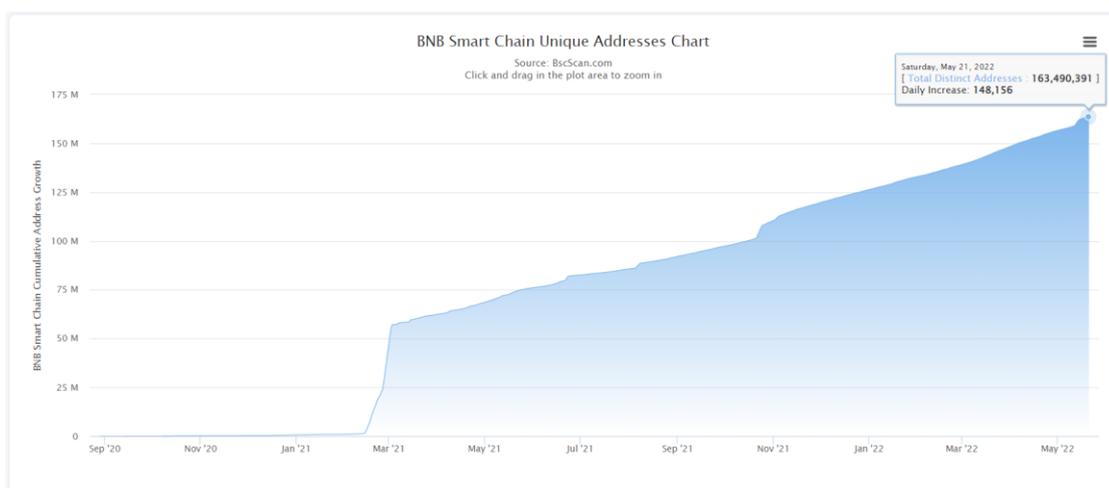


Рисунок 5. Диаграмма количества уникальных адресов в сети BNB Chain (BscScan).

Исходя из всего этого можно сказать, что обе блокчейн сети имеют свои преимущества и недостатки, не позволяя однозначно выбрать лучшую сеть. Таким образом крупные децентрализованные приложения предпочитают развиваться сразу на двух платформах, грамотно используя их преимущества и недостатки.

4 ИЗУЧЕНИЕ АЛГОРИТМОВ КОНСЕНСУСА И ИХ СРАВНЕНИЕ

Каждая из блокчейн сетей содержит свой протокол, то есть набор правил, благодаря которым обеспечивается безопасность транзакций в рамках данной сети. И, так как блокчейн сети являются по определению децентрализованными, каждый из узлов сети в блокчейне должен проверять входящие транзакции. Алгоритмы консенсуса в данном случае отвечают за проверку верности правил, позволяя узлам сети приходиться к общему решению в вопросах добавления новых блоков.

Таким образом, алгоритмы консенсуса в блокчейн сетях являются одними из основополагающих моментов, на которых строится механизм работы блокчейн сетей. В описании сетей Ethereum и BNB Chain происходит частое упоминание алгоритмов консенсуса, без четкого анализа их устройства, хотя различные алгоритмы консенсуса сильно влияют на конечную цель работы блокчейна. Ввиду этого требуется более подробное описание алгоритмов консенсуса целевых блокчейн сетей.

4.1 Разбор алгоритма консенсуса PoW(Proof-of-Work)

Алгоритм консенсуса PoW также называется доказательством работы, так как предусматривает при создании нового блока в блокчейне решение сложной вычислительной задачи, посредством подбора криптографического хэша создаваемого блока. [10]

Процесс создания блока при таком подходе называется майнингом, а узлы сети, которые подбирают хэш называются майнерами. Требования к результату генерации хэша определяются правилами протокола. [11]

Принцип Proof-of-Work был впервые предложен в 1993 году в статье «Pricing via Processing or Combatting Junk Mail» [12] и формулировался с той трактовкой, что для получения доступа к общему ресурсу, пользователь должен вычислить определенную функцию, одновременно ресурсоемкую, сложную, но решаемую за допустимое для целей время. Но при этом данный термин появился только в 1999 году в статье «Proofs of Work and Bread Pudding Protocols» авторства Маркуса Якобссона и Ари Джуелс[13].

В последующем PoW был популяризирован в 2008 году Сатоши Накамото для создания и проверки создаваемых блоков в блокчейн сети Bitcoin. При создании нового блока майнеры бесчисленное количество раз подбирают варианты хэша нового блока, пока он не будет соответствовать поставленной задаче. Зачастую, подобной задачей является подбор хэша с определенным количеством нулей подряд в начале хэша. Подсчет подобного хэша идет таким образом, что для его поиска затрачивается гораздо большее время, чем на проверку его правильности.

Таким образом, когда собрано определенное количество транзакций с наибольшим газом, блок формируется майнерами, и они на скорость пытаются подобрать нужный хэш. Тот майнер, что подобрал хэш законного блока первым, получает награду в виде комиссии за транзакции, а также некоторое количество нативной валюты, что было сгенерировано сетью. Все транзакции в реестре четко организованы, что позволяет предотвратить случаи, когда одну

и ту же транзакцию пользователя зафиксировали дважды, внося элемент доверия в данную схему.

После того, как узел подобрал хэш блока, он транслирует его остальным узлам сети для его проверки. Все остальные узлы сети проверяют подлинность данного хэша и голосуют за включение данного блока в блокчейн. При 51% голосов сети блок добавляется в цепь, таким образом задача для майнера решается, и каждый майнер берется за подбор хэша для следующего блока. Процесс добавления транзакции в блокчейн также продемонстрирован на блок-схеме Рисунка 6.

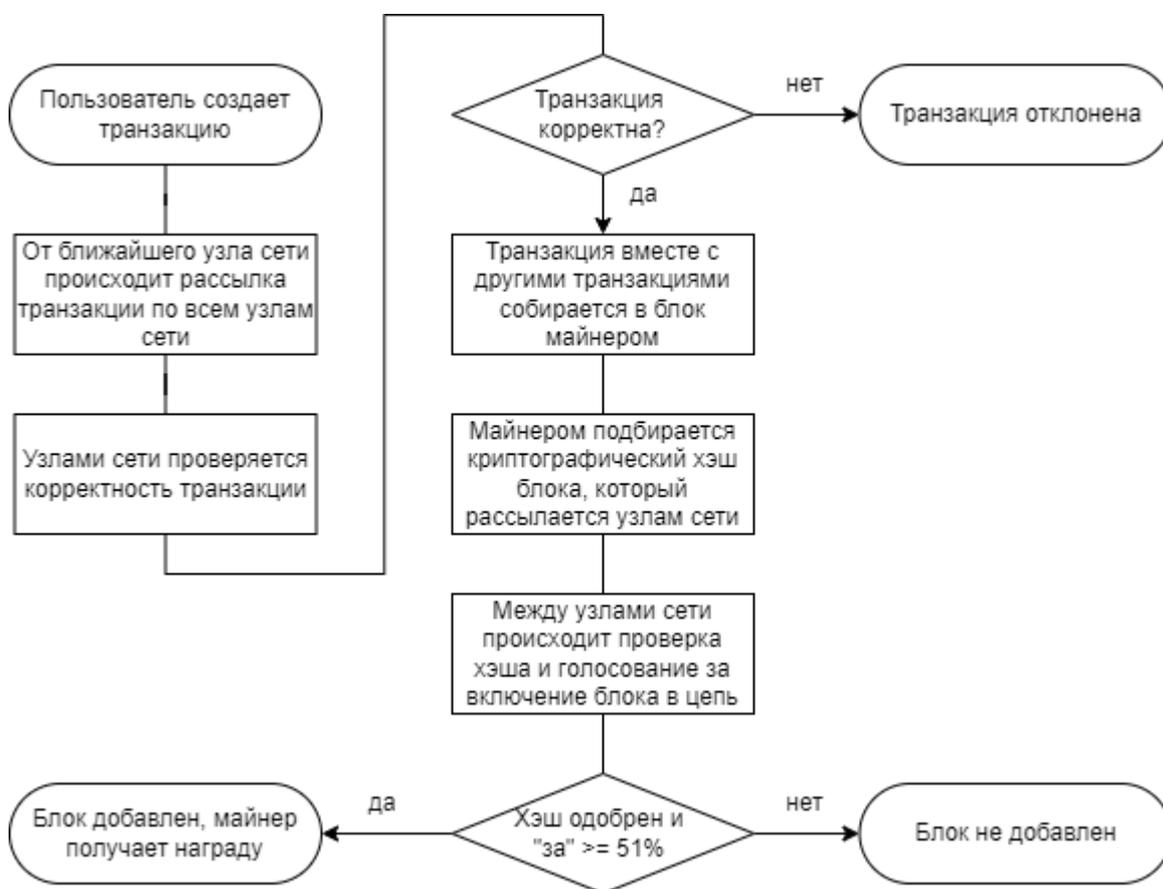


Рисунок 6. Блок-схема добавления транзакции в блокчейн при алгоритме консенсуса PoW.

Данный подход часто критикуется за излишнее энергопотребление и гонку в вычислительной мощности, что является не лучшим решением задачи защиты сети с точки зрения энергоэффективности. Хотя именно количество

вычислительной мощности играет первостепенную роль в безопасности блокчейн сети с этим алгоритмом консенсуса.

Данный алгоритм имеет также и свои уязвимости, например, одна из популярных в приведении в качестве аргумента «атака в 51%». В случае, если злоумышленник получит контроль над более, чем половиной вычислительных мощностей сети, он получает возможность игнорировать чужие блоки, подтверждая только свои, а также блокировать выбранные им транзакции, что понижает доверие к сети в целом.

У данного алгоритма консенсуса также существуют и гибридные варианты, совмещенные с другими алгоритмами консенсуса, или преследующие другие цели в ходе защиты сети. Но это не отменяет того, что данный алгоритм консенсуса является одной из важнейших причин развития блокчейн индустрии в текущее время.

4.2 Разбор алгоритма консенсуса PoS(Proof-of-Stake)

Доказательство доли владения, также известный как PoS, впервые был предложен как новый алгоритм консенсуса в 2011 году на форуме Bitcointalk. [14]

Данный алгоритм преследовал цель решения недостатков высокого энергопотребления PoW. Для этого предлагалось использовать метод, где решающим значением в формировании блока выступает не наиболее мощное оборудование, а количество расчетных единиц криптовалюты, имеющихся у узла сети.

Количество криптовалюты напрямую показывает, какую вероятность имеет узел сети при создании блока, ограничивая процент возможных проверок транзакций. Таким образом, узел сети, владевший $n\%$ криптовалюты от общего числа, в среднем будет создавать $n\%$ новых блоков. Узлы сети, отвечающие за создание блоков в данном случае, имеют название валидаторов.

Процесс добавления транзакции в блокчейн продемонстрирован на блок-схеме Рисунка 7.



Рисунок 7. Блок-схема добавления транзакции в блокчейн при алгоритме консенсуса PoS.

Следует отметить, что данный алгоритм консенсуса, помимо решения проблемы энергопотребления, позволяет увеличить количество узлов сети. А уже их растущее число помогает разработать нормы управления, при которых обеспечивается более сильный иммунитет к централизации. В результате, данный алгоритм консенсуса зачастую рассматривается как алгоритм с наименьшей вероятностью к централизации сети.

При желании узла сети быть рассмотренным для включения в процесс добавления блоков в блокчейн, данный узел должен заблокировать или поставить определенное количество криптовалюты в уникальном контракте. После этого вероятность быть выбранным в роли производителя блоков будет определена количеством заблокированной валюты.

Исходя из этого, многие биржи предлагают свои услуги по размещению ставок от их имени взамен на более стабильные вознаграждения, что дает возможность множеству пользователей, располагающих небольшим

количеством криптовалюты также быть вовлеченными в процесс составления блоков.

Также, как и в случае с текущими реализациями алгоритма PoW, валидаторы в данном алгоритме получают большую часть комиссии за транзакцию, но взамен не получают сгенерированной нативной валюты из-за отказа от высокого энергопотребления.

Данный алгоритм консенсуса также имеет защиту от злонамеренных действий. В ходе обнаружения подобных случаев узел сети может потерять свою долю, что делает возможность использования своего положения непропорциональной возможному наказанию.

Впервые использование этого алгоритма было представлено в криптовалюте PeerCoin. Но на практике в данный момент времени зачастую встречаются либо модифицированные модели, либо гибридные модели в паре с другим алгоритмом консенсуса.

Одной из таких моделей является PoSA (Proof-of-Staked Authority), используемая в блокчейне BNB Chain. Валидаторы сортируются в зависимости от количества заблокированных средств, среди них берется 41 узел, которые в течении следующего дня по очереди будут составлять блоки.

4.3 Сравнение алгоритмов консенсусов PoW и PoS

Сравнение алгоритмов можно разделить по многим аспектам. Одним из которых определено является способ создания блока. В случае с PoS плюсом является отсутствие необходимости в большой вычислительной мощности, но при этом вероятность валидации блока будет настолько же низкой, как и количество имеющейся криптовалюты, что также создает определенный порог для присоединения к текущему количеству узлов сети, отвечающих за создание блоков. Для многих пользователей именно блокчейн с алгоритмом PoW может показаться наиболее заманчивым при выборе. [15]

С другой стороны, количество вычислительной мощности, как и количество заблокированной валюты являются факторами, напрямую влияющими на успех создания блока, но, в отличие от случая с PoW, любой пользователь, имеющий криптовалюту сети с алгоритмом консенсуса PoS, может иметь стабильные награды просто сделав ставку на бирже, что делает порог ниже в случае, если пользователь откажется от прямого участия в составлении блоков.

PoS также выигрывает со стороны децентрализации, так как количество узлов сети возрастает с большей скоростью, позволяя создать иммунитет к централизации. В случае PoW централизация наиболее выражена ввиду того, что решения PoW предназначены для крупномасштабных операций, которые носят централизованный характер.

Уязвимости данных консенсусов являются сравнимыми, ввиду того, что в случае PoW необходимо захватить контроль над 51% узлов сети, в то время как для PoS необходимо иметь контроль над 51% криптовалюты. Оба случая являются равнозначно маловероятными ввиду сложности незаметного исполнения.

Несмотря на плохую энергоэффективность, PoW имеет большую надежность ввиду задействования методов хэширования, в отличие от PoS, где защиту сети подразумевает блокирование криптовалюты в сети, что имеет серьезный вес в вопросах финансовых систем.

Также в PoW разветвление цепочки блоков предотвращается естественным образом благодаря экономическому стимулу, в отличие от PoS, где разветвление автоматически не поощряется PoS системами. Можно сказать, что в данном вопросе оба алгоритма консенсуса имеют свои способы решения проблем.

Исходя из вышеизложенного, можно отметить, что требуемый тип алгоритма консенсуса зависит сугубо от потребностей сети.

PoW может требоваться для обеспечения безопасности и укрепления доверия в сети, а сложность изменений в блоках будет только увеличиваться с течением времени. PoW также поможет определить наиболее законную копию блокчейна в случае, когда в сети появляется множество копий.

PoS, в свою очередь, может использоваться в случаях, когда от сети требуется высокая скорость транзакций и фактических расчетов по сети. Валидаторы, имея большой процент криптовалюты в данном случае также заинтересованы в безопасности сети ввиду финансовой стимуляции.

Хотя оба алгоритма консенсуса имеют свои плюсы и минусы, среди наиболее известных блокчейн сетей происходит отказ от PoW в сторону PoS, ввиду низкой энергоэффективности PoW. Таким образом, возможна ситуация, при которой блокчейны с алгоритмом консенсуса PoW постепенно будут терять своих пользователей, ввиду непопулярности подхода последнего.

5 ОПИСАНИЕ СТАНДАРТОВ ТОКЕНОВ ERC

Для начала стоит отметить понятия EIP и ERC в более широком формате. [16] [17]

EIP (включая ERC) являются проектными документами, содержащими технические характеристики предлагаемого изменения и его обоснование в Ethereum. Являясь частью децентрализованной экосистемы, EIP позволяют любому предлагать, обсуждать и принимать изменения.

Вне зависимости от того, были ли они реализованы в качестве определенного стандарта, либо включены в обновление сети, EIP помогают пользователям понять, как определенные аспекты функционирования Ethereum или токены относятся к действующим смарт-контрактам.

Каждое обновление сети Ethereum сопровождается набором EIP, которые каждый узел сети должен принять для поддержки консенсуса друг с другом.

ERC, в свою очередь, это особый тип EIP, единицы которого являются соглашениями и стандартами на уровне приложений. При этом они могут быть разных типов (токены, схемы, библиотеки и т.д.).

Каждый завершенный EIP меняет само восприятие Ethereum, его возможности и предложения. Вот почему для разработчиков на платформе Ethereum является важным обновлять знания о новых EIP/ERC.

5.1 Описание стандарта токена ERC-20

Стандарт ERC-20 был предложен в ноябре 2015 года Фабианом Вогельстером (Fabian Vogelsteller), как стандарт API для токенов в смарт-контрактах. Данный стандарт предоставляет базовые функции для передачи токенов другим адресам, а также утверждения разрешения для использования токенов третьей стороне. [1]

Технические характеристики к данному стандарту были сформулированы так, что лица, вызывающие функции токена стандарта должны обрабатывать ответ false в случае, если функция возвращает переменную типа bool, так как в некоторых случаях данный вывод возможен.

Листинг 1 – Интерфейс токена стандарта ERC-20

```
interface IERC20 {
    function totalSupply() external view returns (uint256);

    function balanceOf(address who) external view returns (uint256);

    function allowance(address owner, address spender)
        external view returns (uint256);

    function transfer(address to, uint256 value) external returns (bool);

    function approve(address spender, uint256 value)
        external returns (bool);

    function transferFrom(address from, address to, uint256 value)
        external returns (bool);

    event Transfer(
        address indexed from,
        address indexed to,
        uint256 value
    )

    event Approval(
        address indexed owner,
        address indexed spender,
        uint256 value
    )
}
```

В листинге 1 показан интерфейс стандарта токена ERC-20, за исключением функций `name`, `symbol` и `decimals` ввиду того, что они являются стандартными `get` функциями, которые формируются, если параметр является публичным.

Описание функций, входящие в данный стандарт:

- **name** – Функция показывает название токена.
- **symbol** – Функция показывает символ токена, который сокращенно используется для его обозначения
- **decimals** – Функция показывает число, на сколько максимально может быть разделена одна единица токена после запятой.
- **totalSupply** – Функция показывает текущее общее количество данного токена.
- **balanceOf** – Функция показывает баланс токенов аккаунта с адресом, указанным в параметрах функции.
- **transfer** – Функция принимает во входных данных адрес и количество токенов, на которое должен произойти перевод. При успешном выполнении функции, должен произойти перевод токенов с аккаунта пользователя, вызвавшего ее, на аккаунт с указанным адресом, а после этого вывод значения `true`.
- **transferFrom** – Функция принимает во входных данных адрес с которого должно произойти списание токенов, адрес, на который должно произойти начисление токенов, и количество переводимых токенов. Для успешного выполнения данной функции, пользователь, вызывающий ее, должен иметь разрешение на использование данного или большего количества токенов. В ходе выполнения разрешенное количество токенов уменьшается на сумму перевода, после чего происходит вывод значения `true`.

- **approve** – Функция принимает во входных данных адрес пользователя и количество токенов, которых пользователь, вызвавший функцию разрешает использовать пользователю с указанным адресом. При успешном результате, пользователь с адресом во входных данных может использовать функцию **transferFrom** с адресом пользователя, вызвавшего функцию approve, как адрес, с которого происходит списание средств.
- **allowance** – Функция показывает текущее разрешенное количество средств для использования между двумя адресами, указанными во входных данных, как пользователя, который разрешил использование средств и пользователя, которому использование средств было разрешено.

Также, у данного стандарта токена есть два события, которые должны вызываться при успешном выполнении функции, на которые эти события были рассчитаны:

- **Transfer** – Событие показывает, что произошел вызов функции transfer, либо **transferFrom**, то есть перевод определенного количества средств с одного адреса на другой, включая случаи, когда перевод произошел на нулевой адрес.
- **Approval** – Событие показывает, что произошел вызов функции approve, то есть пользователь, вызвавший эту функцию, разрешил использовать определенное количество средств другому пользователю.

Помимо стандартных функций, смарт-контракты могут иметь другие функции и события в зависимости от дальнейшего направления работы токена.

Стоит отметить, что, в зависимости от функционала и направления работы токена, логика стандартных функций может быть изменена

разработчиком, вплоть до полного удаления оригинальной функциональности. Такие модификации могут встречаться в случае токенов, которые могут быть перемещены с одного адреса на другой только адресами, имеющими на то специальные права. Подобные токены зачастую используются не по прямому назначению, а в ходе работы других, более сложных, смарт-контрактов.

Почти все, реализованные на данный момент, токены данного стандарта не используют исключительно стандартные функции, так как это сильно ограничивает возможности используемого токена.

Стандарт ERC-20 имеет цель позволить множеству токенов, которые добавили использование данных функций и событий, быть отмеченными, как соответствующие одобренному разработчиками стандарту, таким образом облегчая работу с данной функцией для смарт-контрактов, уже использующих в своей работе токены данного стандарта.

5.2 Описание стандарта токена ERC-721

Стандарт ERC-721 был предложен в январе 2018 года Уильямом Энтрикеном (William Entriken), Дитером Ширли (Dieter Shirley), Джейкобом Эвансом (Jacob Evans) и Настасьей Сакс (Nastassia Sachs), как стандарт API для NFT в смарт-контрактах. Данный стандарт предоставляет базовые функции для отслеживания и передачи NFT. [2]

Стандартный интерфейс на листингах 2 и 3 позволяет dApp приложениям кошелька/брокера/аукциона работать с любым NFT на Ethereum.

Листинг 2 – Интерфейс токена стандарта ERC-721 (Начало)

```
interface IERC721 is IERC165 {
    event Transfer(
        address indexed _from,
        address indexed _to,
        uint256 indexed _tokenId
    );
    event Approval(
        address indexed _owner,
        address indexed _approved,
        uint256 indexed _tokenId
    );
    event ApprovalForAll(
        address indexed _owner,
        address indexed _operator,
        bool _approved
    );

    function balanceOf(address _owner) external view returns (uint256);

    function ownerOf(uint256 _tokenId) external view returns (address);

    function approve(address _approved, uint256 _tokenId) external;

    function getApproved(uint256 _tokenId) external view returns (address);

    function setApprovalForAll(address _operator, bool _approved) external;

    function isApprovedForAll(address _owner, address _operator)
        external
        view
        returns (bool);
```

Листинг 3 – Интерфейс токена стандарта ERC-721 (Продолжение)

```
function transferFrom(  
    address _from,  
    address _to,  
    uint256 _tokenId  
) external;  
  
function safeTransferFrom(  
    address _from,  
    address _to,  
    uint256 _tokenId,  
    bytes calldata data  
) external;  
  
function safeTransferFrom(  
    address _from,  
    address _to,  
    uint256 _tokenId  
) external;  
}
```

Каждая единица токена стандарта ERC-721 является уникальной и имеет определенный id, который идентифицирует ее в данном смарт-контракте. Ниже представлены описания функций, которые смарт-контракт должен иметь в своей реализации для отметки, как токен стандарта ERC-721:

- **balanceOf** – Функция показывает количество имеющихся уникальных единиц токенов у пользователя с адресом во входных данных.
- **ownerOf** – Функция показывает адрес владельца единицы токена с указанным id единицы во входных данных.
- **approve** – Функция принимает во входных данных адрес и id токена, который пользователь, вызвавший функцию, позволяет использовать другому пользователю с указанным адресом. Для успешного выполнения данной функции пользователь, вызвавший ее, должен обладать единицей токена с указанным id. При выполнении данного условия пользователь с указанным адресом может использовать данную единицу токена, при условии, что ее

владелец не сменился до начала использования функции оперирующей данной единицей токена.

- **getApproved** – Функция показывает адрес пользователя, которому разрешено использование единицы токена с указанным во входных данных id.
- **setApprovalForAll** – Функция принимает во входных данных адрес и переменную типа bool, которая указывает, разрешено ли пользователю с указанным адресом использовать все токены пользователя, вызвавшего данную функцию.
- **isApprovedForAll** – Функция принимает во входных данных два адреса, владельца и оператора, и показывает, имеет ли оператор разрешение на использование всех токенов владельца.
- **transferFrom** – Функция принимает во входных данных два адреса, отправителя и получателя, а также id единицы токена, который в ходе выполнения данной функции переходит во владение от одного адреса к другому. Для выполнения данной функции пользователь либо должен являться владельцем данной единицы токена, либо иметь адрес, который был ранее разрешен для пользования владельцем единицы токена. В случае успешного выполнения функции, при изменении владельца, право пользования единицей токена аннулируется, независимо от адреса, вызвавшего данную функцию.
- **safeTransferFrom** – Функция действует аналогично функции **transferFrom**, с тем исключением, что не позволяют передачу права собственности единицей токена адресам, которые не поддерживают перевод токенов на другие адреса. На деле это означает, что смарт-контракт с адресом получателя должен реализовывать интерфейс кошелька, что был представлен ранее в листинге 5. Помимо этого, в стандарте токена ERC-721 существует аналогичная функция **safeTransferFrom**,

принимающая на вход на один аргумент типа bytes больше. Данный аргумент позволяет отослать сообщение или любую другую полезную информацию, если реализация токена данного стандарта того требует.

Аналогично стандарту ERC-20, стандарт ERC-721 имеет три обязательных к добавлению события, которые демонстрируют успешное завершение функций:

- **Transfer** – Событие показывает, что произошел вызов функции **transferFrom**, либо **safeTransferFrom**, то есть передача права владения единицы токена с определенным id с одного адреса на другой, включая случаи, когда перевод произошел на нулевой адрес
- **Approval** – Событие показывает, что произошел вызов функции **approve**, таким образом пользователь, вызвавший функцию, назначил определенному адресу право владения единицей токена с определенным id.
- **ApprovalForAll** – Событие показывает, что произошел вызов функции **setApprovalForAll**, таким образом пользователь, вызвавший функцию, доверил использование всех своих токенов определенному адресу.

Помимо данных функций и событий смарт-контракт токена стандарта ERC-721, может иметь функции для получения имени токена и его символа, аналогично стандарту ERC-20, но в данном случае эти функции лишь опциональны.

Также, в связи с тем, что в экземплярах токенов стандарта ERC-721 предполагалась реализация более, чем одного интерфейса, данный стандарт также имеет обязательную реализацию интерфейса ERC-165, который представлен на листинге 4 и преследует цель показа сторонним смарт-

контрактам имеет ли существующий контракт реализацию целевого интерфейса.

Листинг 4 – Интерфейс стандарта ERC-165

```
interface IERC165 {  
    function supportsInterface(bytes4 interfaceID) external view returns (bool);  
}
```

Помимо необходимых к реализации интерфейсов у самого смарт-контракта токена, в стандарте ERC-721 также существует интерфейс, который должен реализовывать любой другой смарт-контракт, что должен поддерживать прием и отправку токенов подобного стандарта (Листинг 5).

Листинг 5 – Интерфейс кошелька

```
interface ERC721TokenReceiver {  
    function onERC721Received(  
        address _operator,  
        address _from,  
        uint256 _tokenId,  
        bytes calldata _data  
    ) external returns (bytes4);  
}
```

Как и в случае со стандартом ERC-20, смарт-контракты, реализующие стандарт ERC-721, могут иметь изменения в логике своих функций, в зависимости от направления работы токена и других деталей, характеризующих конечную цель создания смарт-контракта.

Важное отличие, характеризующее стандарт токена ERC-721, в том, что, в отличие от стандарта ERC-20, данный стандарт рассчитан на единицы токенов, которые являются уникальными по своей сути и не могут быть заменены любой другой единицей этого же токена с другим идентификатором. Проводя аналогию, ERC-20 – это любая валюта, имеющая количественный характер, в то время как ERC-721 – это коллекция каких-либо вещей, сгруппированных в одном месте.

6 РАЗРАБОТКА СМАРТ-КОНТРАКТА МОСТА

6.1 Проектирование смарт-контрактов

Как можно понять, исходя из объяснения блокчейн технологии, разные блокчейн сети не имеют прямого доступа друг к другу, создавая необходимость в посреднике при требованиях ко взаимодействию между смарт-контрактами на разных блокчейн сетях.

В самом наглядном виде, что продемонстрирован в данной работе, посредником между смарт-контрактами токенов выступают два смарт-контракта моста, которые имеют доступ к административным функциям для создания/удаления единиц токенов.

Таким образом, при взаимодействии с бэкенд сервисом, который в определенном порядке отслеживает транзакции перевода между блокчейн сетями, происходит требуемое взаимодействие между двумя смарт-контрактами токенов.

При данных условиях, можно судить о вариантах использования комплекса токена и моста, относительно роли обычного пользователя в случаях использования разных стандартов (Рисунок 6), а также о различиях между данными стандартами в вариантах использования (Рисунок 7 и 8). Помимо этого, можно выстроить дополнительные варианты использования для администратора (Рисунок 9), а также бэкенд сервиса.

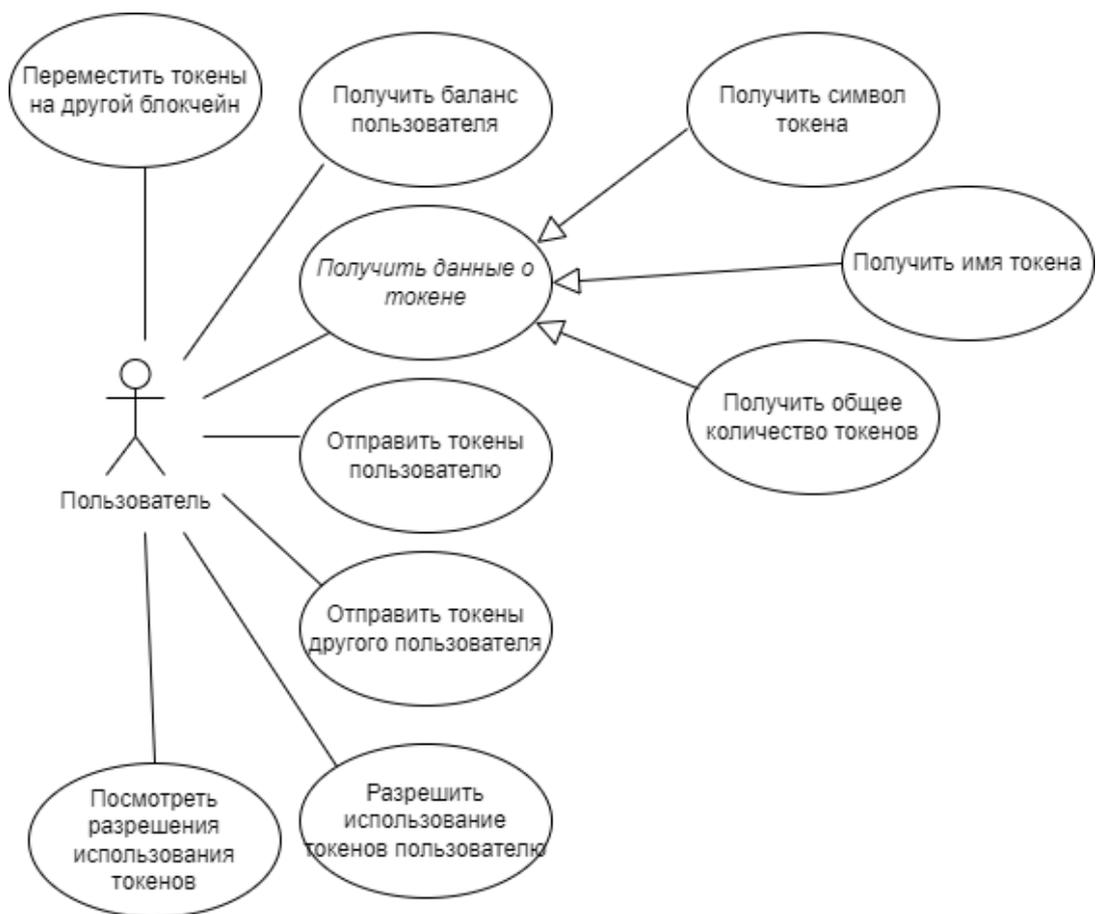


Рисунок 6. Диаграмма общих вариантов использования смарт-контрактов разных стандартов токенов для роли пользователь.



Рисунок 7. Диаграмма дополнительных вариантов использования смарт-контрактов для роли пользователь, когда стандарт токена ERC-20.

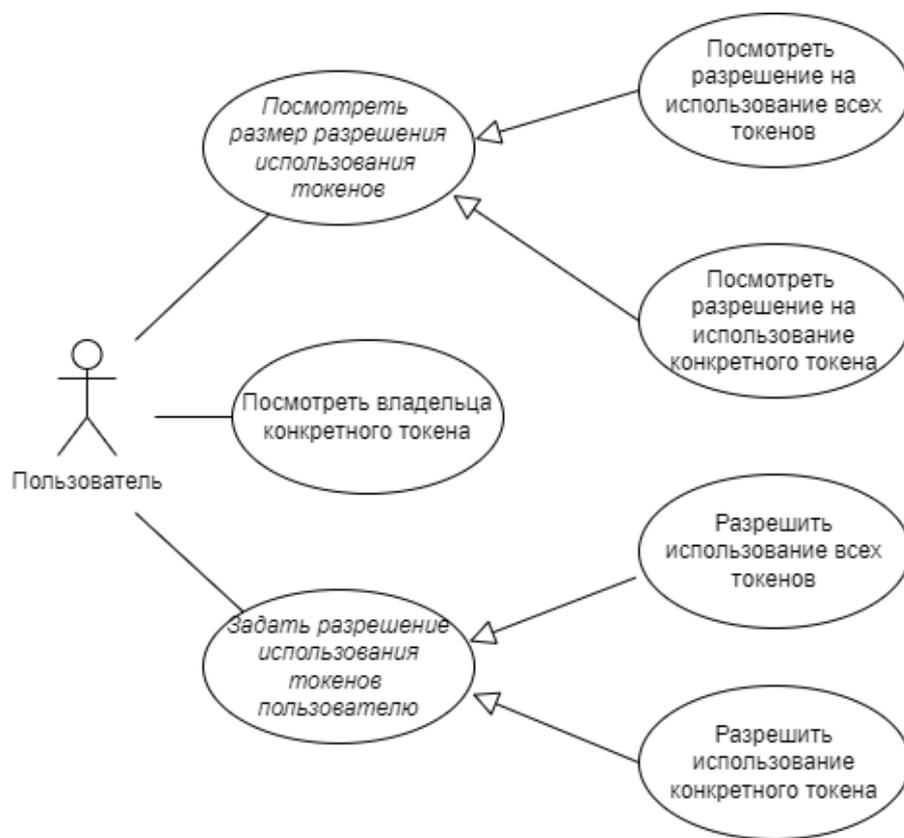


Рисунок 8. Диаграмма дополнительных вариантов использования смарт-контрактов для роли пользователь, когда стандарт токена ERC-721.

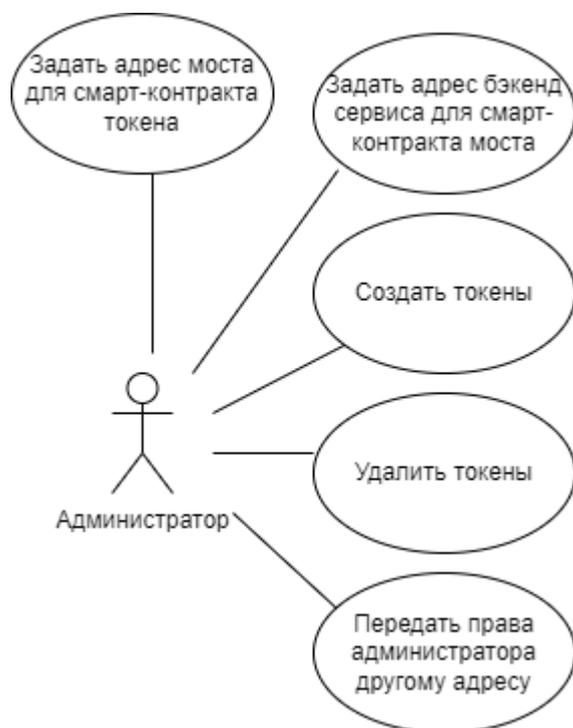


Рисунок 9. Диаграмма вариантов использования смарт-контрактов для роли администратор

При рассмотрении диаграмм вариантов использования на рисунке 6, можно заметить, что много ВИ являются общими для разных стандартов ввиду общих целей, несмотря на различия стандартов токенов в характеристиках. И, несмотря на то, что все варианты использования, кроме ВИ «Переместить токены на другой блокчейн», относятся к смарт-контракту токена, ввиду того, что функциональность моста в данном случае ограничивается лишь пересылкой токенов из одной блокчейн сети в другую, это сильно расширяет возможности для пользователей токенов, ввиду различий характеристик блокчейн сетей.

На рисунке 9 ВИ Администратора ограничены доступными лишь ему функциями для удобства восприятия его возможностей. В остальном, Администратор имеет ровно те же права, что и остальные пользователи блокчейн сети относительно его смарт-контракта.

Для большего понимания работы моста стоит рассмотреть ВИ «Переместить токены на другой блокчейн» с помощью диаграммы последовательностей (Рисунок 10) и сценария данного ВИ. (Таблица 1)

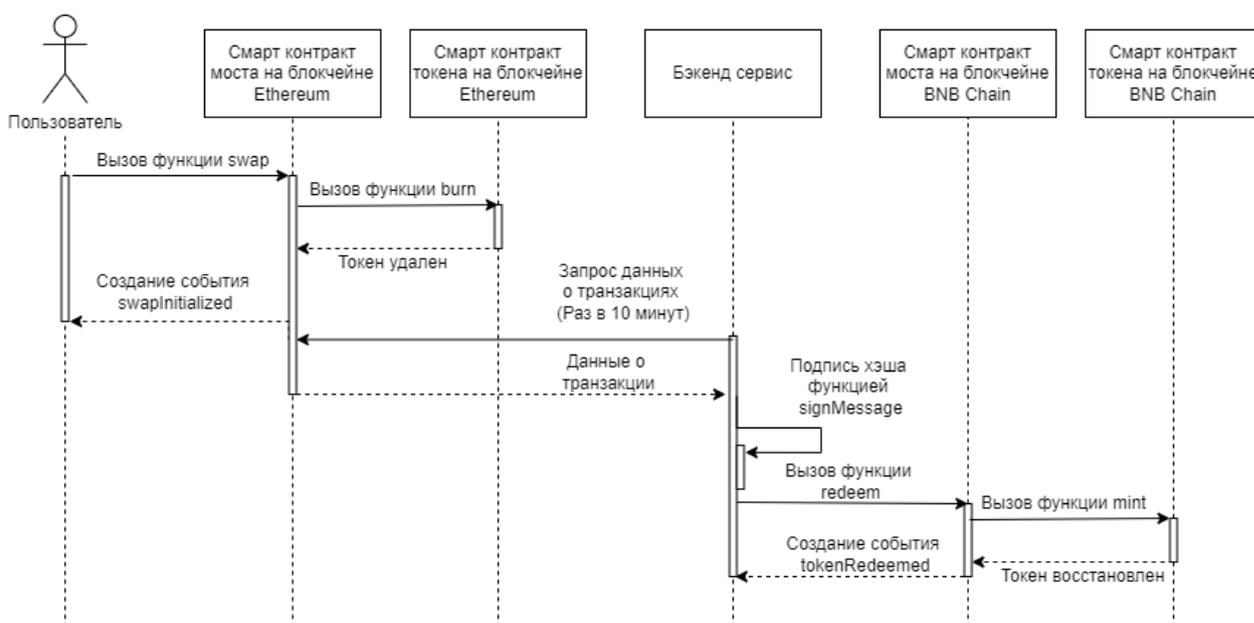


Рисунок 10. Диаграмма последовательности ВИ «Переместить токены на другой блокчейн»

Таблица 1. Сценарий ВИ «Переместить токены на другой блокчейн»

Название ВИ	Переместить токены на другой блокчейн
Цель	По вызову пользователя функции <code>swap</code> переместить определенный токен / некоторое количество токенов с одной блокчейн сети на другую
Актер	Пользователь
Предусловия	Вызов функции « <code>approve</code> » / « <code>setApprovalForAll</code> » для передачи разрешения на использование токена(ов) смарт-контрактом моста
Сценарий	<ol style="list-style-type: none"> 1. Пользователь вводит данные для передачи своего токена(ов) и <code>id</code> блокчейн моста 2. Пользователь подтверждает транзакцию 3. Блокчейн сохраняет транзакцию и создает событие «<code>swapInitialized</code>» 4. Бэкенд сервис, создает и подписывает хэшированные данные 5. Бэкенд сервис создает транзакцию с хэшированными данными для целевого блокчейна 6. Целевой блокчейн сохраняет транзакцию и создает событие «<code>tokenRedeemed</code>»
Альтернативы	<ol style="list-style-type: none"> 2.1. Пользователь отклоняет транзакцию 3.1 Блокчейн отклоняет транзакцию, так как у пользователя недостаточно средств / нет прав на данный токен 6.1 Блокчейн отклоняет транзакцию, так как подобная транзакция выполнялась ранее

6.2 Разработка смарт-контрактов токенов

Исходя из функционала стандарта токена ERC-20, а также функций, необходимых для взаимодействия со смарт-контрактом моста, была разработана структура данных смарт-контракта, что продемонстрировано на диаграмме классов ниже (Рисунок 11).



Рисунок 11. Диаграмма классов смарт-контракта токена стандарта ERC-20

В данном случае, переменные `_name`, `_symbol`, `_decimals` и `_totalSupply` хранят основные данные о токене, тогда как `_balances` и `_allowances` являются

структурами данных с предоставлением данных по ключу, которые хранят значения балансов пользователей и разрешенных сумм для пользования доверенными лицами. Оставшиеся переменные хранят адреса владельца данного контракта, который отвечал за развертывание смарт-контракта токена в блокчейн-сеть, и смарт-контракта моста.

Помимо стандартных функций, реализованных из стандарта ERC-20, были также добавлены функции удаления и создания токенов. Первая из данных функций, `burn`, отвечает за удаление токенов с баланса пользователя и необходима для списания токенов, при желании со стороны пользователя перенести токены на другую блокчейн сеть. Вторая функция, `mint`, позволяет восстановить (создать) токены на балансе определенного адреса. И, так как данные функции предназначены для узкого круга лиц, они разделяют между собой модификаторы, ограничивающие допуск к данным функциям до владельца контракта и смарт-контракта моста. (Листинг А.1, А.2)

Помимо этого, для добавления/смены адреса смарт-контракта моста, была добавлена функция `setBridgeAddress`, к которой имеет доступ только владелец смарт-контракта токена. (Листинг А.3)

Данные функции вместе дают возможность сжигать и добавлять определенное количество токенов на балансы определенных пользователей, что и требуется для корректной работы функции по перенаправлению токенов между смарт-контрактами разных блокчейн сетей.

Аналогичным образом, исходя из функционала стандарта токена стандарта ERC-721, была разработана структура данных смарт-контракта, что продемонстрировано на диаграмме классов ниже (Рисунок 12).

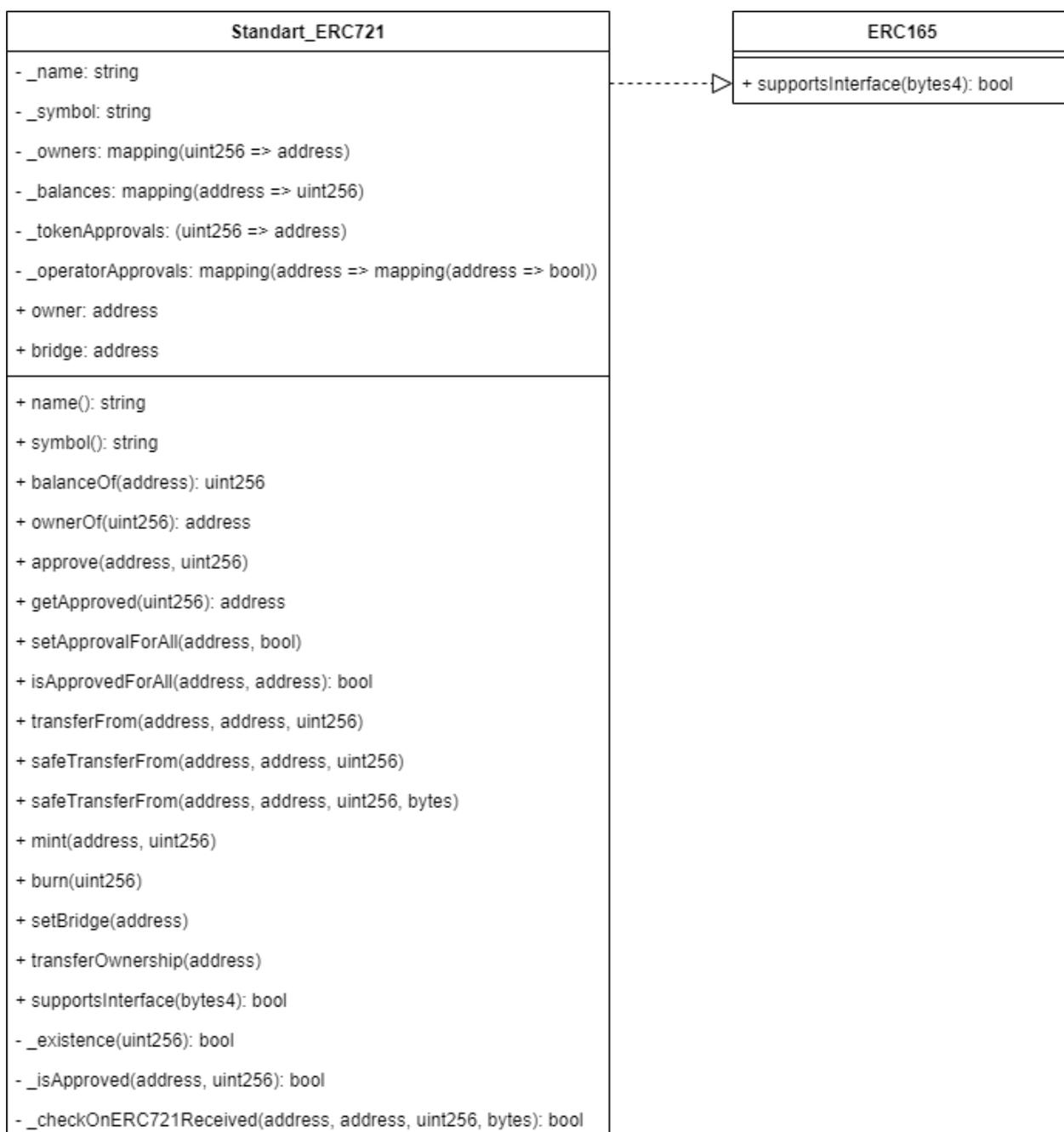


Рисунок 12. Диаграмма классов смарт-контракта токена стандарта
ERC-721

Переменные `_name` и `_symbol` данного смарт-контракта выполняют те же роли, что и в стандарте ERC-20, храня основную информацию о токене. `_owners` и `_balances` являются структурами данных с предоставлением данных по ключу, что хранят в себе текущих владельцев токенов и их балансы. Структуры `_tokenApprovals` и `_operatorApprovals` хранят разрешения на использование токенов другими пользователями.

Помимо функций стандарта ERC-721, аналогично стандарту ERC-20, были добавлены функции сжигания (burn) и создания (mint) токенов, а также функция для смены адреса смарт-контракта моста (setBridgeAddress), при разработке которых происходило ориентирование на структуру данных смарт-контракта токена.

6.3 Разработка смарт-контрактов моста

В начале написания смарт-контракта моста, необходимо понять какие переменные и массивы нужны в структуре данных создаваемого смарт-контракта.

Во-первых, необходимо иметь переменную, которая будет хранить id номер блокчейна в сети. Таким образом, при необходимости расширения на другие сети, не будет необходимости развертывать контракт по новой.

Во-вторых, необходимо хранить адрес смарт-контракта токена, так как текущая реализация моста не предполагает операции с более, чем одним токеном необходимого стандарта.

В-третьих, необходимо хранить адрес бэкенд сервиса, чтобы было возможно создать ограничения для запуска функций с допуском только для этого адреса.

И последнее, необходимо хранить данные о перемещениях токенов между сетями. Таким образом можно предотвратить повторный вызов второй функции (redeem), которая будет отвечать за перевод токенов в блокчейн сети получателя.

Стоит также отметить, что для двух токенов разных стандартов должны быть реализованы два моста, во многом идентичных, кроме главных функций, отвечающих за перемещение токенов между блокчейн сетями. И, так как эти отличия затрагивают лишь названия переменных, структуру обоих мостов можно показать на одной диаграмме классов. (Рисунок 13)

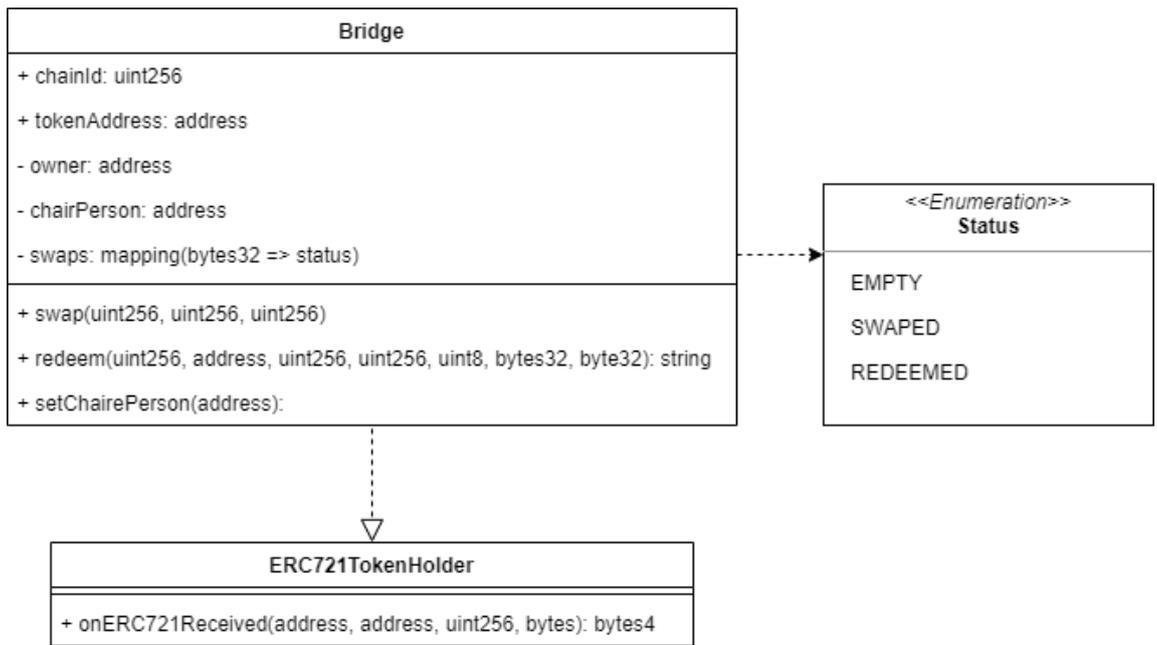


Рисунок 13. Диаграмма классов смарт-контракта моста, относительно стандарта ERC-721

Так как смарт-контракт моста должен реализовывать интерфейс кошелька из Листинга 5, для возможности в дальнейшем использовать токены стандарта ERC-721, данный интерфейс был также отображен на диаграмме классов смарт-контракта моста. Данный интерфейс характерен лишь для хранения токенов стандарта ERC-721, таким образом смарт-контракт моста для токенов стандарта ERC-20 не реализовывает данный интерфейс.

Первая из трех функций смарт-контракта моста, setChairePerson, используется для задания нового адреса бэкенд сервиса. Без использования данной функции, бэкенд сервис не будет иметь доступа к функциям, предназначенным для его пользования.

Следующая по счету, и первая из двух функций, отвечающих за перемещение токенов между сетями – функция swar. Диаграмма активности данной функции находится на рисунке 14. Внутри функции проверяется, имеет ли пользователь права на данную единицу токена, после чего данные об этом перемещении хэшируются и записываются в структуре данных с ключом в виде данного хэша. И в конце токен уничтожается на данной блокчейн сети. Отличии этой же функции для токена ERC-20 в том, что вместо параметра

_itemId посылается параметр _amount, показывающий, сколько токенов нужно списать со счета пользователя. (На Листинге А.4 показана версия данной функции для токена стандарта ERC-721.)

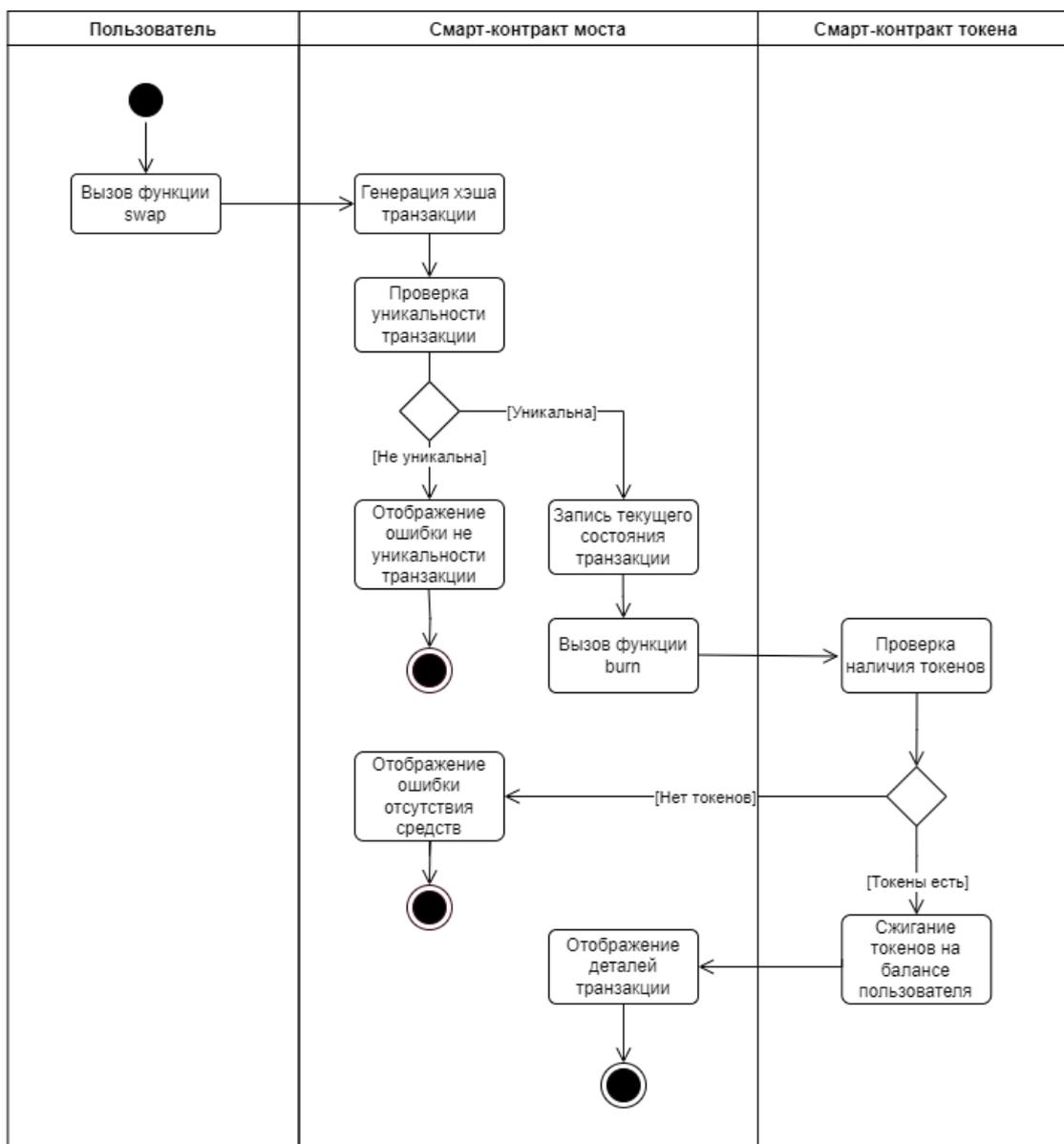


Рисунок 14. Диаграмма активности функции swap

Последняя оставшаяся функция, которая является продолжением функции swap – функция redeem, диаграмма активности которой показана на рисунке 15. В начале функция при помощи модификатора проверяется, является ли вызывающий данную функцию адрес адресом бэкенд сервиса, после чего происходит расшифровка зашифрованных данных. В случае, если

человек, вызвавший данную функцию, является тем же лицом, кто подписал данный хэш, перевод признается успешным, то есть в структуру данных с ключом в виде хэша записывается, что перевод прошел, а также создается токен, права на который принадлежат человеку, вызвавшему функцию swap в прошлой блокчейн сети. Отличие данной функции для токена стандарта ERC-20, аналогично различию в функциях swap: параметр `_itemId` заменяется на параметр `_amount`, который показывает сколько токенов было отослано при функции swap. (На Листинге А.5 показана версия данной функции для токена стандарта ERC-721.)

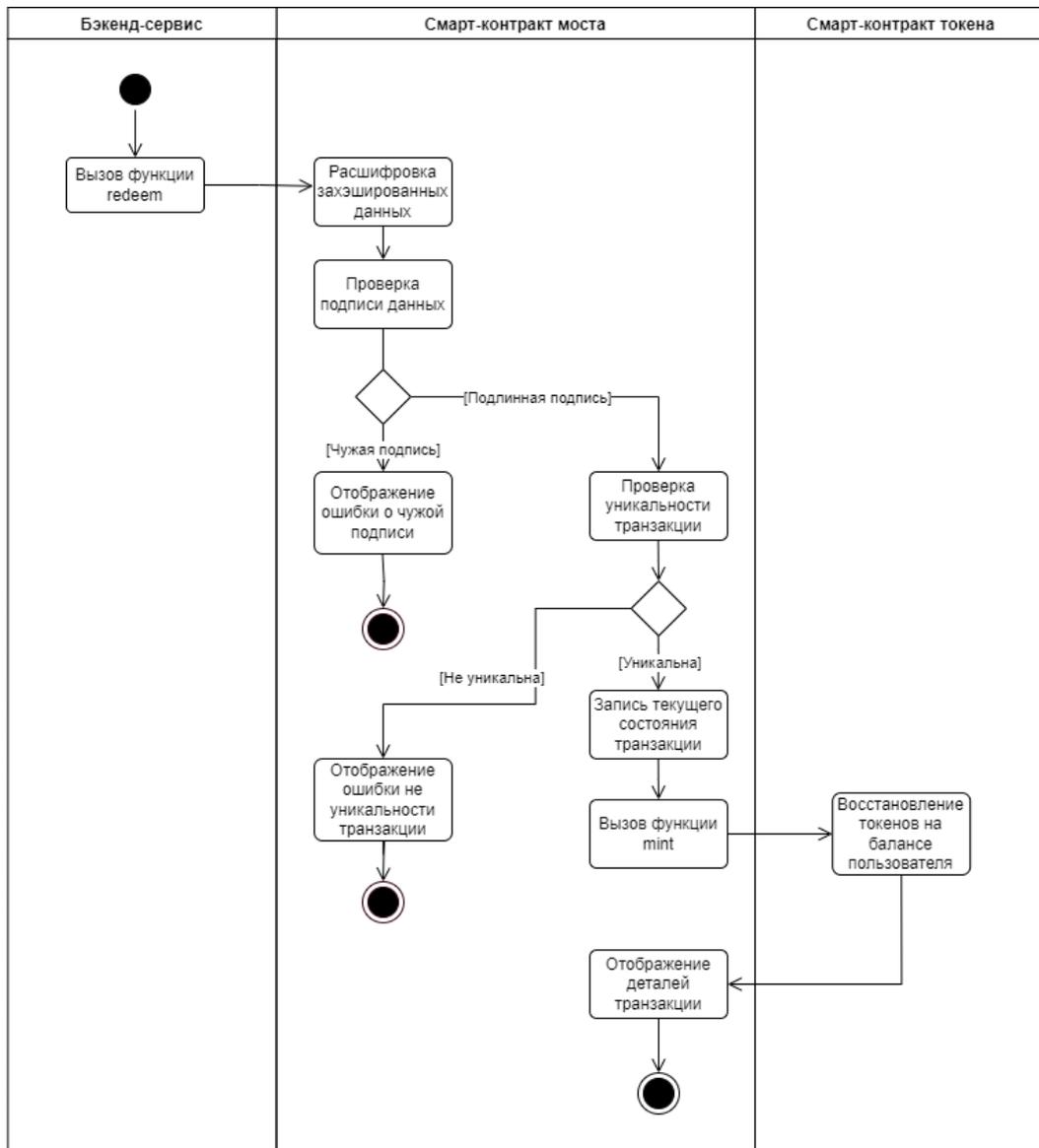


Рисунок 15. Диаграмма активности функции swap

7 АНАЛИЗ ИНСТРУМЕНТОВ И ТЕСТИРОВАНИЕ

Языком разработки смарт-контрактов данной работы был выбран Solidity – предметно-ориентированный язык программирования, широко распространившийся с появлением технологии блокчейн, особенно в стеке технологий на Ethereum. Solidity наиболее популярный и надежный язык программирования смарт-контрактов для работы с EVM.

Для разработки и тестирования смарт-контрактов была выбрана среда разработки Visual Studio Code, ввиду удобства пользования и доступности всех необходимых расширений и библиотек для работы со смарт-контрактами. Помимо этого, для Visual Studio Code была установлена среда разработки Hardhat, специально разработанная для более удобной и функциональной работы со смарт-контрактами на языке Solidity.

Помимо этого, были установлены расширения и библиотеки:

- **hardhat-ethers** – плагин, позволяющий взаимодействовать с блокчейном Ethereum.
- **hardhat-etherscan** – плагин, позволяющий верифицировать контракт на блокчейн сети. Верифицирование подразумевает под собой процедуру, позволяющую прочитать контракт и взаимодействовать с ним посредством транзакций с помощью крипто-кошелька.
- **hardhat-waffle** – плагин, позволяющий писать тесты, используя waffle, на hardhat.
- **hardhat-web3** – плагин, позволяющий hardhat работать с web3 модулем.
- **hardhat-contract-sizer** – плагин для показа размера скомпилированного смарт-контракта.
- **hardhat-gas-reporter** – плагин для показа используемого газа в тестах.

- **mocha** – библиотека для использования тестовой среды mocha, включая отчеты о покрытии тестов, а также использование библиотек утверждений
- **chai** – библиотека утверждений для работы с mocha.
- **node** – библиотека для использования node.js в тестах.

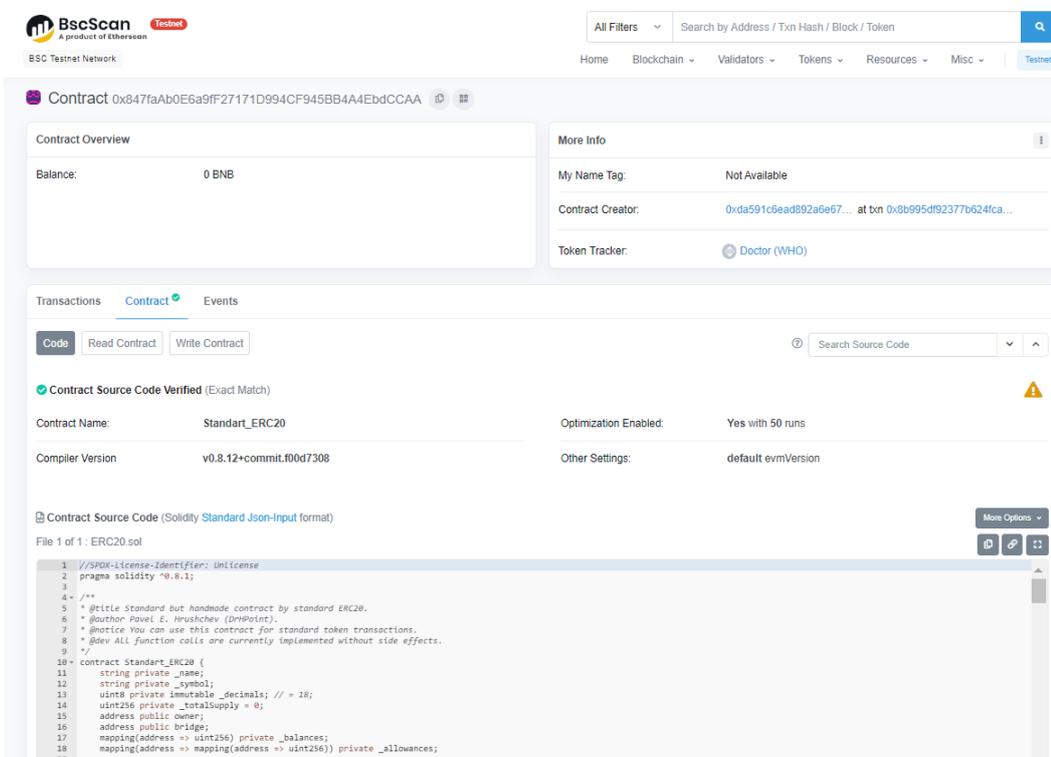
Данная сборка библиотек позволяет провести комплексное тестирование и работу с развернутыми в блокчейн сети смарт-контрактами.

8 РЕЗУЛЬТАТЫ

С помощью сред разработки было проведено комплексное тестирование смарт-контрактов, показывающее их работоспособность и функциональность в рамках поставленной задачи.

Часть теста, имитирующего сценарий работы функций смарт-контракта моста относительно токена стандарта ERC-721 показано на Листинге А. 5

Также, смарт-контракты токенов стандарта ERC-20, ERC-721 и смарт-контракты моста для этих токенов были развернуты в тестовой сети BSC testnet (Рисунок 14-17) и Rinkeby



The screenshot displays the BscScan interface for a smart contract on the BSC Testnet. The contract address is 0x847faAb0E6a9f27171D994CF945BB4A4EbdCCAA. The contract overview shows a balance of 0 BNB. The 'More Info' section indicates that the contract creator is 0xda591c6ead892a6e67... and the token tracker is Doctor (WHO). The 'Contract Source Code Verified' section shows that the source code is verified (Exact Match) and is named Standard_ERC20. The compiler version is v0.8.12+commit.f00d7308, and optimization is enabled with 50 runs. The contract source code is displayed in Solidity format, showing the contract definition for Standard_ERC20.

```
1 //SPDX-License-Identifier: Unlicense
2 pragma solidity ^0.8.1;
3
4 /**
5  * @title Standard but handmade contract by standard ERC20.
6  * @author Pavel E. Hruschev (@npoint).
7  * @notice You can use this contract for standard token transactions.
8  * @dev ALL function calls are currently implemented without side effects.
9  */
10 contract Standard_ERC20 {
11     string private _name;
12     string private _symbol;
13     uint8 private immutable _decimals; // = 18;
14     uint256 private _totalSupply = 0;
15     address public owner;
16     address public bridge;
17     mapping(address => uint256) private _balances;
18     mapping(address => mapping(address => uint256)) private _allowances;
19 }
```

Рисунок 14. Верифицированный в сети BSC testnet смарт-контракт токена стандарта ERC-20 (code)

BscScan A product of Etherscan
BSC Testnet Network

All Filters Search by Address / Txn Hash / Block / Token

Home Blockchain Validators Tokens Resources Misc Testnet

Contract 0xe82e785Fc77c7365B50CfDcbec30d32768B647C2

Contract Overview

Balance: 0 BNB

More Info

My Name Tag: Not Available

Contract Creator: 0xda591c6ead892a6e67... at txn 0xeb724667db696583a...

Transactions **Contract** Events

Code Read Contract Write Contract Search Source Code

Contract Source Code Verified (Exact Match)

Contract Name: Bridge_ERC20 Optimization Enabled: Yes with 50 runs

Compiler Version: v0.8.12+commit.f00d7308 Other Settings: default evmVersion

Contract Source Code (Solidity Standard Json-Input format)

File 1 of 4 : BridgeERC20.sol

```

1 //SPDX-License-Identifier: Unlicense
2 pragma solidity ^0.8.1;
3
4 import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";
5 import "@openzeppelin/contracts/utils/math/SafeMath.sol";
6
7 contract Bridge_ERC20 {
8     using ECDSA for bytes32;
9     address owner;
10    address bridge;
11    uint256 public chainId;
12    address public ERC20Token;
13    mapping (bytes32 => status) swaps;
14    enum status { EMPTY, SWAPED, REDEEMED }
15
16
17    constructor(address _ERC20Address, uint256 _chainId) {
18        owner = msg.sender;
19        ERC20Token = _ERC20Address;
20        chainId = _chainId;
21    }
22

```

Рисунок 15. Верифицированный в сети BSC testnet смарт-контракт моста для токена стандарта ERC-20 (code)

BscScan A product of Etherscan
BSC Testnet Network

All Filters Search by Address / Txn Hash / Block / Token

Home Blockchain Validators Tokens Resources Misc Testnet

Contract 0x0d2c041AcB56A36334fD15727279fbB287de3a14

Contract Overview

Balance: 0 BNB

More Info

My Name Tag: Not Available

Contract Creator: 0xda591c6ead892a6e67... at txn 0xf0bea20a1da35538fa3...

Transactions **Contract** Events

Code Read Contract Write Contract Search Source Code

Contract Source Code Verified (Exact Match)

Contract Name: Standart_ERC721 Optimization Enabled: Yes with 50 runs

Compiler Version: v0.8.12+commit.f00d7308 Other Settings: default evmVersion

Contract Source Code (Solidity Standard Json-Input format)

File 1 of 7 : ERC721.sol

```

1 //SPDX-License-Identifier: Unlicense
2 pragma solidity ^0.8.1;
3
4 import "@openzeppelin/contracts/utils/Address.sol";
5 import "./interfaces/ERC721Metadata.sol";
6 import "./interfaces/ERC721TokenReceiver.sol";
7 import "./interfaces/IERC721.sol";
8 import "./ERC165.sol";
9
10 contract Standart_ERC721 is ERC165, IERC721, ERC721Metadata {
11
12     using Address for address;
13
14     string private _name;
15     string private _symbol;
16     address public owner;
17     address public bridge;
18     mapping(uint256 => address) private _owners;
19     mapping(address => uint256) private _balances;
20     mapping(uint256 => address) private _tokenApprovals;
21     mapping(address => mapping(address => bool)) private _operatorApprovals;
22

```

Рисунок 16. Верифицированный в сети BSC testnet смарт-контракт токена стандарта ERC-721 (code)

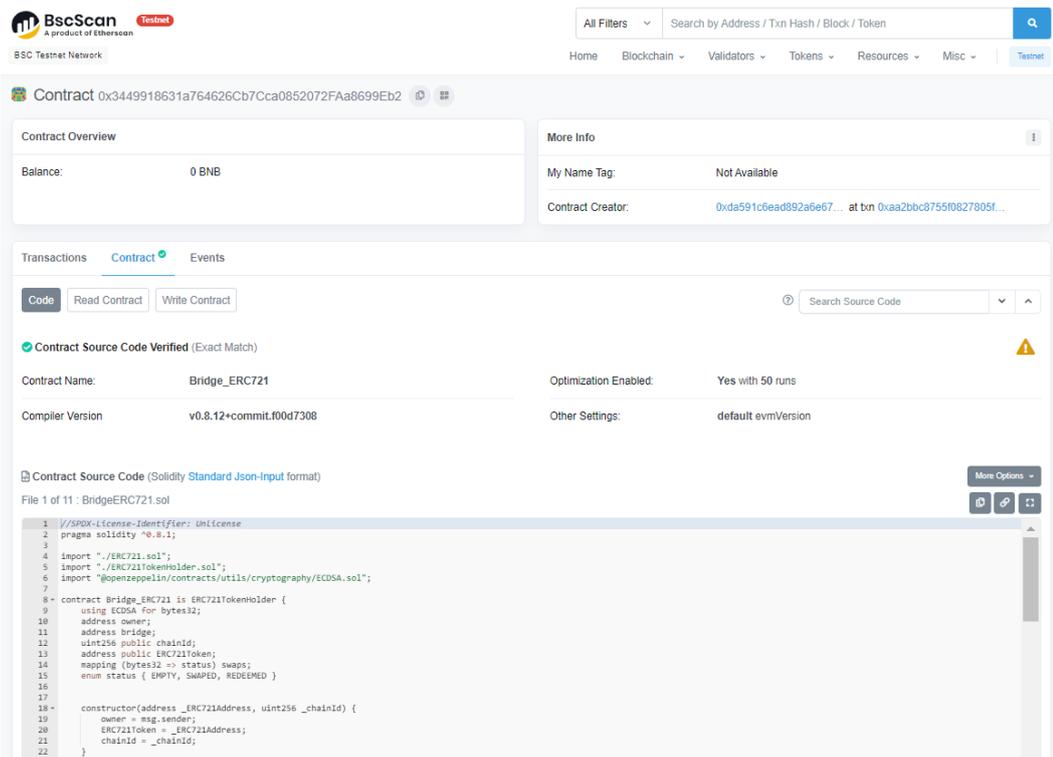


Рисунок 17. Верифицированный в сети BSC testnet смарт-контракт моста для токена стандарта ERC-721 (code)

Для демонстрации работоспособности моста был произведен перевод единицы токена стандарта ERC-721 со смарт-контракта токена, находящегося в тестовой сети Rinkeby, на смарт-контракт токена, находящегося в тестовой сети BSC testnet (Рисунок 18-19)

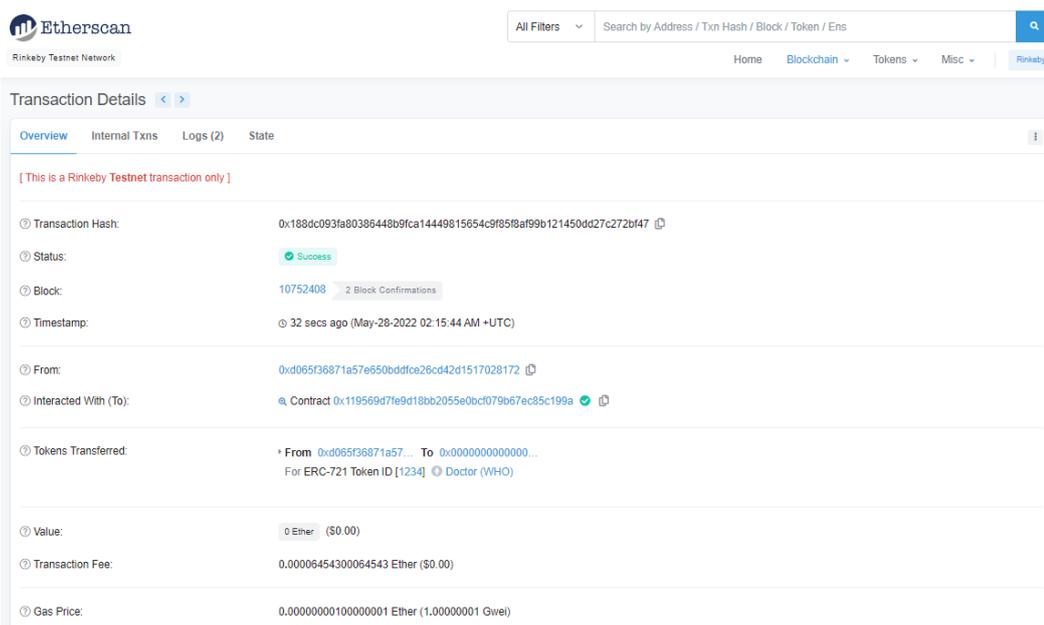


Рисунок 18. Транзакция функции swap в тестовой сети Rinkeby

BscScan
A product of Etherscan

BSC Testnet Network

All Filters Search by Address / Txn Hash / Block / Token

Home Blockchain Validators Tokens Resources Misc Testnet

Transaction Details

Overview Internal Txns Logs (1)

[This is a Bsc Testnet transaction only]

- Transaction Hash: [0xad8bdc1d6401460d7630af327c715b1e2960c915a101c177143978758ff064](#)
- Status: Success
- Block: [19688160](#) 4 Block Confirmations
- Timestamp: 12 secs ago (May-28-2022 02:19:43 AM +UTC)
- From: [0x68ae1a060f8d6783fc6b81127e8ff7d14bea1381](#)
- Interacted With (To): [Contract 0x3449918631a764626cb7cca0852072faa8999eb2](#)
- Tokens Transferred: From [0x0000000000000000000000000000000000000000](#) To [0xd065f36871a57...](#) For ERC-721 TokenID [1234] [Doctor \(WHO\)](#)
- Value: 0 BNB (\$0.00)
- Transaction Fee: 0.00101148 BNB (\$0.30)

[Click to see More](#)

Рисунок 19. Транзакция функции redeem в тестовой сети BSC testnet

ЗАКЛЮЧЕНИЕ

В результате данной квалификационной работы были реализован смарт-контракты токенов стандарта ERC-20 и ERC-721, а также смарт-контракты моста для обоих токенов.

Функциональность смарт-контрактов проверена в среде разработки, а также в тестовых сетях целевых блокчейн сетей. Смарт-контракты обладают всем требуемым функционалам, и могут использоваться как часть для реализации децентрализованных приложений.

СПИСОК ЛИТЕРАТУРЫ

1. EIP-20: Token Standard – URL: <https://eips.ethereum.org/EIPS/eip-20>
2. EIP-721: Non-Fungible Token Standard – URL: <https://eips.ethereum.org/EIPS/eip-721>
3. ИНС: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Ethereum> (дата обращения: 20.05.2022)
4. ИНС: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Binance> (дата обращения: 20.05.2022)
5. Bytwork.com [Электронный ресурс] – URL: <https://bytwork.com/articles/chto-takoe-binance-smart-chain-otlichie-ot-ethereum>
6. bscscan.com [Электронный ресурс] – URL: <https://bscscan.com/>
7. etherscan.io [Электронный ресурс] – URL: <https://etherscan.io/>
8. academy.binance.com [Электронный ресурс] – URL: <https://academy.binance.com/ru/articles/binance-smart-chain-vs-ethereum-what-s-the-difference>
9. block-chain24.com [Электронный ресурс] – URL: <https://www.block-chain24.com/news/novosti-ethereum/tri-prichiny-pochemu-ethereum-poprezhnemu-luchshe-chem-binance-smart-chain>
10. ИНС: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Доказательство_выполнения_работы (дата обращения: 20.05.2022)
11. tadviser.ru [Электронный ресурс] – URL: [https://www.tadviser.ru/index.php/Статья:Алгоритм_консенсуса_Proof-of-Work_\(PoW\)_и_Proof-of-Stake_\(PoS\)](https://www.tadviser.ru/index.php/Статья:Алгоритм_консенсуса_Proof-of-Work_(PoW)_и_Proof-of-Stake_(PoS))
12. Pricing via Processing or Combatting Junk Mail – URL: <https://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.pdf>
13. Proofs of Work and Bread Pudding Protocols – URL: <http://www.arijuels.com/wp-content/uploads/2013/09/JJ99.pdf>

14. ИНС: Википедия. Свободная энциклопедия. – URL:
https://ru.wikipedia.org/wiki/Доказательство_доли_владения (дата обращения: 20.05.2022)
15. cointelegraph.com [Электронный ресурс] – URL:
<https://cointelegraph.com/blockchain-for-beginners/proof-of-stake-vs-proof-of-work:-differences-explained>
16. adrianhetman.com [Электронный ресурс] – URL:
<https://www.adrianhetman.com/what-the-eip-erc/>
17. coindesk.com [Электронный ресурс] – URL:
<https://www.coindesk.com/learn/what-are-eip-and-erc-and-how-are-they-connected/>

ПРИЛОЖЕНИЕ А

Листинг А.1 – Функция burn в смарт-контракте токена стандарта ERC-20

```
function burn(address _from, uint256 _value) public onlyForOwnerAndBridge
returns (bool success){
    require(_from != address(0), "Burn from the zero address");
    require(_balances[_from] >= _value, "Not enough tokens");
    _balances[_from] -= _value;
    _totalSupply -= _value;
    emit Transfer(_from, address(0), _value);
    return true;
}
```

Листинг А.2 – Функция mint в смарт-контракте токена стандарта ERC-20

```
function mint(address _to, uint256 _value) public onlyForOwnerAndBridge
returns (bool success){
    require(_to != address(0), "Mint to the zero address");
    _balances[_to] += _value;
    _totalSupply += _value;
    emit Transfer(address(0), _to, _value);
    return true;
}
```

Листинг А.3 – Функция setBridgeAddress в смарт-контракте токена стандарта ERC-20

```
function setBridgeAddress(address _newBridge) public onlyForOwner {
    require(_newBridge != address(0), "New bridge have the zero address");
    bridge = _newBridge;
}
```

Листинг А.4 – Функция swap для токена стандарта ERC-721

```
function swap(
    uint256 _itemId,
    uint256 _chainIdTo,
    uint256 _nonce
) external {
    require(Standart_ERC721(ERC721Token).ownerOf(_itemId)
        == msg.sender, "User has no rights to this token");
    bytes32 dataHash = keccak256(
        abi.encodePacked(_itemId, msg.sender, _nonce, chainId, _chainIdTo)
    );
    swaps[dataHash] = status.SWAPED;
    Standart_ERC721(ERC721Token).burn(_itemId);
    emit swapInitialized(_itemId, msg.sender, chainId, _chainIdTo);
}
```

Листинг А.5 – Функция redeem для токена стандарта ERC-721

```
function redeem(
    uint256 _itemId,
    address _owner,
    uint256 _chainIdFrom,
    uint256 _nonce,
    uint8 _v,
    bytes32 _r,
    bytes32 _s
) external {
    require(msg.sender == bridge, "Caller is not owner of bridge");

    bytes32 dataHash = keccak256(
        abi.encodePacked(_itemId, _owner, _nonce, _chainIdFrom, chainId)
    );
    address signer = ECDSA.recover(dataHash.toEthSignedMessageHash(), _v, _r,
    _s);
    require(signer == msg.sender, "Signature is wrong");
    require(swaps[dataHash] != status.REDEEMED, "Already Redeemed");
    swaps[dataHash] = status.REDEEMED;
    Standart_ERC721(ERC721Token).mint(_owner, _itemId);
}
```

Листинг А.6 – Тест основного сценария работы мостов

```
describe("Bridges", () => {
  it("Bridges", async () => {
    await nft.deployed();
    await bsc.deployed();
    await bridge1.deployed();
    await bridge2.deployed();
    const BridgeRole1 = await nft.connect(owner).setBridge(bridge1.address);
    const BridgeRole2 = await bsc.connect(owner).setBridge(bridge2.address);
    await BridgeRole1.wait();
    await BridgeRole2.wait();
    const ChairePerson1 = await
bridge1.connect(owner).setChairePerson(addr3.address);
    const ChairePerson2 = await
bridge2.connect(owner).setChairePerson(addr3.address);
    await ChairePerson1.wait();
    await ChairePerson2.wait();
    const Mint1 = await nft.connect(owner).mint(addr1.address, 1234);
    await Mint1.wait();
    const SwapNFT = await bridge1.connect(addr1).swap(1234, 2, 1);
    await SwapNFT.wait();
    const signedDataHash = ethers.utils.solidityKeccak256(
      ["uint256", "address", "uint256", "uint256", "uint256"],
      [1234, addr1.address, 1, 1, 2]
    );
    const byteArray = ethers.utils.arrayify(signedDataHash);
    const flatSignature1 = await addr3.signMessage(byteArray);
    const signature1 = ethers.utils.splitSignature(flatSignature1);
    const RedeemBSC = await bridge2.connect(addr3).redeem(1234, addr1.address,
1, 1, signature1.v, signature1.r, signature1.s);
    await RedeemBSC.wait();
  });
});
```

Введите текст:

...или загрузите файл:

Файл не выбран...

Выбрать файл...

Укажите год публикации: 2022 ▾

Выберите коллекции

Все

Рефераты

Авторефераты

Иностранные конференции

PubMed

Википедия

Российские конференции

Иностранные журналы

Российские журналы

Энциклопедии

Англоязычная википедия

Анализировать

Год публикации: 2022.

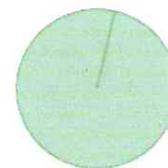
Оценка оригинальности документа - 100.0%

Процент условно корректных заимствований - 0.0%

Процент некорректных заимствований - 0.0%

Время выполнения: 8 с.

Заимствования отсутствуют

[Значимые оригинальные фрагменты](#)
[Искать в Интернете](#)


100.00%

[Дополнительно](#)

М.В.С. / Кичи К.С.
 14.06.2022

