Командный практический тур

Легенда

Как и любая база данных, блокчейн система имеет ограничения – ограничения на скорость обработки транзакций, на количество информации, которое может храниться и обрабатываться одним узлом-участником в блокчейн сети. К тому же, в публичных блокчейн сетях, где пользователю необходимо платить за транзакции, эти ограничения выливаются в увеличение стоимости транзакции.

Это называется проблемой масштабирования (scalability) – при увеличении количества обрабатываемой информации система должна позволять органически изменять свои свойства, чтобы системой могли продолжать пользоваться. При этом, очевидно, что в самом начале система (при маленьком количестве обрабатываемой информации) не должна быть избыточна, поскольку это удорожает ее поддержку на ранних стадиях.

Одним из решением проблем масштабирования в блокчейн системах является организация дополнительных блокчейн сетей. Каждая сеть может обслуживать конкретный проект или группу пользователей. Например, может быть отдельная сеть для децентрализованной биржи, отдельная сеть для IoT устройств, отдельная сеть для голосования и т.п. При этом между сетями выстраиваются каналы межблокчейнового взаимодействия, позволяющие передавать информацию из одной блокчейн системы в другую. Таким образом, формируется сеть блокчейн систем.

Каналы межблокчейнового взаимодействия могут быть выстроены следующим образом:

1. Атомарный обмен (atomic swap, hashed timelock contracts, https://www.investopedia.com/terms/h/hashed-timelock-contract.asp) применяется исключительно для обмена денежными средствами и осуществляется через контракты. Два пользователя должны предварительно договориться о сумме обмена, затем один из них (*Alice*) регистрирует в одной блокчейн сети контракт (`C1`), содержащий захешированное значение секретного ключа, и помещает договоренную сумму на счет данного контракта. Другой пользователь (Bob) полностью копирует данный контракт (а значит и хэш секретного ключа), регистрирует его (`C2`) в другой блокчейн сети и помещает договоренную сумму на счет этого контракта. Теперь *Alice* может обратиться к контракту `C2` для получения суммы обмена, но должна для этого предъявить контракту секретный ключ. Как только контракт исполняется, ключ становится известен Bob, что позволяет ему забрать средства с контракта `C1`.
2. Мост, собранный по нотариальной схеме (federated notaries), позволяет передавать, в отличие от atomic swap, любую информацию между двумя блокчейн сетями. В основе моста лежит оракул – сущность, следящая за изменением состояния одной блокчейн системы в результате обработки транзакций, и доставляющая информацию об изменении в другую блокчейн систему. Продолжая аналогию с мостом, можно эти две разные блокчейн системы назвать сторонами моста. Следовательно, суть оракула – передать информации с одной

стороны моста на другую сторону. Для увеличения надежности и уменьшения возможности цензурирования используют несколько независимых оракулов. Тогда переданная информация считается достоверной, если ее подтвердят несколько оракулов. Например, если всего оракулов 5, то передача информации достоверна при наличии трех подтверждений.

3. Релейный мост также позволяет передавать любую информацию между двумя блокчейн системами. При этом является более децентрализованным, чем мост с нотариальной схемой. Но сложность в разработке является существенным недостатком данного моста. По факту, данное решение требует реализации легких клиентов (light clients) в виде контрактов, которые опираясь на информацию, хранимую в заголовке блока, и предоставляемое доказательство наличия искомой информации в теле блока, позволяли бы оперировать самой информацией. При этом релееры (relays) ответственны за перенос заголовков блоков одной блокчейн сети в другую, за счет чего и происходит передача событий из одного блокчейна в другой: блокчейн `N1` содержит заголовки блоков блокчейна `N2`, что позволяет любому пользователю блокчейна `N1`, предъявив доказательство (Merkle Proof) наличия события в блоке из `N2`, инициировать обработку этого события в блокчейн `N1`.

В финальной задаче сезона 2020/21 годов предлагается разработать простейший федеративный мост (пункт 2 в списке выше) для передачи токенов между двумя блокчейн сетями.

Для простоты мост будет работать только с нативными токенами - "техническими" токенами блокчейн сети, используемыми для оплаты комиссий (например, *ether* для Ethereum Mainnet или *SPOA* для тестовой сети Sokol).

Тогда типовой сценарий взаимодействия с мостом будет следующий:

1. *Alice* посылает токены на контракт моста в одной блокчейн сети (назовем ее "левая" сторона моста). При этом транзакция, посылающая токены, генерирует определенное событие.
2. Каждый из оракулов моста независимо от других оракулов следит за появлением событий в сети, находящейся на "левой" стороне, и, как только событи обнаружено, посылает подтверждение в контракт моста в блокчейне, находящемся на "правой" стороне.
3. Как только контракт моста собирает достаточное количество подтверждений, то он пересылает соответствующее количество токенов *Alice*.

В итоге, у *Alice* создается впечатление, что она послала токены в одной блокчейн сети, а получила их в другой.

Набор заданий

Решение командной задачи разбито на подзадачи, сгруппированные в 3 набора.

Каждая подзадача (*user story*) формулирует необходимый функционал, который должен быть реализован командой, а также набор приемочных тестов (*acceptance criteria*), позволяющих проверить в полном ли объеме решена данная подзадача.

Каждый набор подзадач предлагается решать в отдельном этапе (*итерации*).

На каждом этапе решение подзадач проверялось с помощью системы автома-

тической системы оценивания, которая запускала решение участников с теми или иными параметрами в соответствии с приемочными тестами.

Первая итерация

Участникам необходимо разработать базовый функционал контрактов федеративного моста, а также скрипт их регистрации (deployment) в блокчейн сетях.

user story (*)
US-001 Сборка docker образа
US-002 Регистрация контрактов из скрипта
US-003 Контракт управления набором валидаторов: возможность добавления валидаторов
US-004 Контракт управления набором валидаторов: возможность удаления валидаторов
US-005 Контракт управления набором валидаторов: изменения порогового значения
US-006 Контракт моста: добавление ликвидности
US-007 Контракт моста: установка верхнего лимита
US-008 Контракт моста: отправка токенов на контракт на левой стороне
US-009 Контракт моста: отправка подтверждения от валидатора на правую сторону
US-010 Контракт моста: отправка токенов на контракт на правой стороне
US-011 Контракт моста: отправка подтверждения от валидатора на левую сторону

(*) формулировки задач приведены в разделе "Подробное описание подзадач".

Решение подзадач было направлено на проверку следующих компетенций:

- реализация командной работы с использованием системы версифицирования кода Git;
- сборка контейнерованных сервисов с помощью *docker*;
- написание контракта *Ethereum* на языке Solidity;
- использование *EIP* стандартов для реализации стандартизированных контрактов;
- развертывание контракта и взаимодействие с ним из языка Python с использованием библиотеки `Web3`.

Вторая итерация

Участникам необходимо завершить работу над минимально достаточным прототипом системы, который бы позволял запустить оракулов федеративного моста, обрабатывающих запросы на передачу информации из одной блокчейн сети в другую.

Данный функционал покрывался следующими подзадачами:

user story (*)
US-012 Оракул: пересылка события об отправленном токене на правую сторону
US-013 Оракул: пересылка события об отправленном токене на левую сторону
US-014 Оракул: пересылка события об отправленном токене при одновременной работе нескольких оракулов
US-015 Оракул: Работа оракула с неверным приватным ключом
US-016 Оракул: Перезапуск оракула
US-017 Контракт моста: надежная доставка токенов

(*) формулировки задач приведены в разделе "Подробное описание подзадач".

Решение подзадач было направлено на проверку следующих компетенций:

- взаимодействие с узлом Ethereum сети из кода на Python для отслеживание событий, сгенерированных контрактом *Ethereum*.
- взаимодействие с узлом Ethereum сети из кода на Python для вызова метода контрактов *Ethereum*.
- работа с простыми базами данных, сохраняющих свое состояние на жестком диске, из Python;
- запуск контейнерованных сервисов с помощью *docker-compose*.

Третья итерация

Работа участников должна быть направлена на наращивание функционала прототипа системы:

- достижение экономической эффективности контрактов и оракулов;
- предоставление пользователю CLI для дублирования некоторых действий оракулов моста;
- увеличение безопасности контрактов.

user story (*)
US-018 Контракт моста: экономичная посылка подтверждений в левую сторону
US-019 Оракул: экономичная посылка подтверждений в левую сторону
US-020 CLI: ручная пересылка пользователем собранных подтверждений в левую сторону
US-022 Контракт моста: лимиты на пересылку
US-023 Отключение моста через контракт
US-024 Контракт моста: ожидание финализации цепочки блоков

(*) формулировки задач приведены в разделе "Подробное описание подзадач".

Решение подзадач было направлено на проверку следующих компетенций:

- генерация цифровой подписи по приватному ключю из Python;
- проверка цифровой подписи в *Ethereum* контракте.

Условия проведения

- Участники во время командного этапа финального тура могут использовать интернет и заранее подготовленные библиотеки для решения задачи.
- Участники не ограничены в выборе языков программирования при решении задачи.
- Участники не могут пользоваться помощью тренера, наставника, сопровождающего лица или привлекать третьих лиц для решения задачи.
- Финальная задача формулируется участникам в первый день финального тура, но участники выполняют решение задачи поэтапно. Критерии прохождения каждого этапа формулируются для каждого дня финального тура. За подзадачи, решенные в конкретном этапе начисляются, баллы. Баллы за определенные

подзадачи можно получить только в день, закрепленный за конкретным этапом.

- В начале первого дня состязаний участники каждой команды получают доступ к репозиторию на серверах GitLab.com. Каждая команда имеет свой собственный репозиторий. Члены других команд не имеют доступ к чужим репозиториям.
- В течение дня не ведется учет количества изменений, которые команды регистрируют в Git-репозитории.
- В конце каждого дня финального этапа жюри проверяет решение участников на соответствие приемочным тестам для каждой подзадачи, входящей в набор для соответствующего этапа.
- Баллы за все подзадачи, для которых прошло приемочное тестирование, определяют баллы, набранные командой в данный день соревнований. Для некоторых позадач есть возможность получать баллы с дисконтом в последующие дни соревнований. Также есть поздадачи, которые позволяют набирать доп. баллы, если приемочные тесты для этих подзадач проходят в последующие дни (регрессионное тестирование).
- Поскольку решение участников запускается система автоматического тестирования в контейнеризированном окружении, то других технических требований к ней не предъявляется.
- После выставления баллов, командам может предоставляется доступ к системе автоматического тестирования, ответственной за проведение приемочных тестов в конкретный день состязаний, так что члены команды могут ознакомиться с логикой проверки и подать апелляцию, если не согласны с корректностью проведения тестов.
- После рассмотрения сути апелляции, жюри вправе провести тестирование еще раз и назначить команде баллы за соответствующие подзадачи.
- Описанные выше условия могут быть изменены членами жюри. Все изменения в условиях объявляются участникам перед началом каждого дня состязаний.

Процедура проведения приемочного тестирования и критерии оценки

Для каждого дня соревнований (для каждой итерации) справедлива следующая процедура приемочного тестирования:

- В конце каждого дня финального этапа команды должны сформировать запрос на слияние (*Merge Request*) из своей ветки исходного кода в основную ветку (*master*) в Git-репозитории.
- Команда ответственна за то, чтобы в запросе на слияние не должно быть конфликтов. Запрос на слияние с конфликтами может не рассматриваться жюри для выполнения приемочного тестирования.
- После того, как все команды отправили запросы на слияние, жюри одобряет все запросы и приступает к приемочному тестированию, для тех подзадач, которые входят в соответствующую итерацию. Для этого исходный код приложения команды загружается в систему автоматического тестирования (поддержку которой осуществляет функциональность GitLab CI/CD), где запуска-

ются автоматические тесты на соответствие решения участников требованиям к приемочным тестам (*acceptance criteria*).

- Если все приемочные тесты для данной подзадачи пройдены успешно, команда получает баллы за данную подзадачу. Если хотя бы один тест не проходит, то баллы за данное подзадачу не начисляются.

Приемочные тесты для каждой подзадачи описаны в разделе "Подробное описание подзадач".

Дальше перечислены баллы, которые получает команда за решение подзадач в каждой итерации.

Максимальное количество баллов, которое может набрать команда за решение всех подзадач — 400.

Первая итерация

user story	**баллы**
US-001 Сборка docker образа	4
US-002 Регистрация контрактов из скрипта	12
US-003 Контракт управления набором валидаторов: возможность добавления валидаторов	10
US-004 Контракт управления набором валидаторов: возможность удаления валидаторов	10
US-005 Контракт управления набором валидаторов: изменения порогового значения	10
US-006 Контракт моста: добавление ликвидности	7
US-007 Контракт моста: установка верхнего лимита	8
US-008 Контракт моста: отправка токенов на контракт на левой стороне	10
US-009 Контракт моста: отправка подтверждения от валидатора на правую сторону	30
US-010 Контракт моста: отправка токенов на контракт на правой стороне	4
US-011 Контракт моста: отправка подтверждения от валидатора на левую сторону	7

Максимальное количество баллов за итерацию — 112.

Вторая итерация

user story	**баллы**
US-008 Контракт моста: отправка токенов на контракт на левой стороне	4
US-009 Контракт моста: отправка подтверждения от валидатора на правую сторону	4
US-010 Контракт моста: отправка токенов на контракт на правой стороне	4
US-011 Контракт моста: отправка подтверждения от валидатора на левую сторону	4
US-012 Оракул: пересылка события об отправленном токене на правую сторону	45
US-013 Оракул: пересылка события об отправленном токене на левую сторону	10
US-014 Оракул: пересылка события об отправленном токене при одновременной работе нескольких оракулов	10
US-015 Оракул: Работа оракула с неверным приватным ключом	10
US-016 Оракул: Перезапуск оракула	17
US-017 Контракт моста: надежная доставка токенов	18

Максимальное количество баллов за итерацию — 126.

Третья итерация

user story	баллы
US-008 Контракт моста: отправка токенов на контракт на левой стороне	4
US-009 Контракт моста: отправка подтверждения от валидатора на правую сторону	4
US-010 Контракт моста: отправка токенов на контракт на правой стороне	4
US-011 Контракт моста: отправка подтверждения от валидатора на левую сторону	4
US-012 Оракул: пересылка события об отправленном токене на правую сторону	4
US-013 Оракул: пересылка события об отправленном токене на левую сторону	4
US-014 Оракул: пересылка события об отправленном токене при одновременной работе нескольких оракулов	4
US-015 Оракул: Работа оракула с неверным приватным ключом	4
US-016 Оракул: Перезапуск оракула	4
US-017 Контракт моста: надежная доставка токенов	4
US-018 Контракт моста: экономичная посылка подтверждений в левую сторону	30
US-019 Оракул: экономичная посылка подтверждений в левую сторону	22
US-020 CLI: ручная пересылка пользователем собранных подтверждений в левую сторону	8
US-022 Контракт моста: лимиты на пересылку	28
US-023 Отключение моста через контракт	10
US-024 Контракт моста: ожидание финализации цепочки блоков	24

Максимальное количество баллов за итерацию — 162.

Критерии определения команды-победителя командного тура

- Сумма баллов, набранных за решения подзадач командного тура финального этапа, определяет итоговую результативность команды (измеряемую в баллах).
- Команды ранжируются по результативности.
- Команда победитель определяется, как команда с максимальной результативностью.

Подробное описание подзадач

Образ с решением задачи

Решение задачи должно быть представленно в виде Docker образа `n2n-oracle`. Образ должен включать контракты, скрипт регистрации контрактов, оракула и т.д.

Требования к контрактам (ABI)

Контракт управления набором валидаторов:

- метод `addValidator(address newvalidator)` позволяет владельцу контракта добавлять нового валидатора.
- метод `removeValidator(address validator)` позволяет владельцу контракта удалить одного из существующих валидаторов.
- метод `changeThreshold(uint256 thresh)` позволяет владельцу контракта изменить пороговое значение необходимых подвтерждений от валидаторов, чтобы

передача информации из одной блокчейн системы в другу считалась достоверной.

- метод `getValidators()` позволяет получить текущий набор валидаторов.
- метод `getThreshold()` позволяет получить текущее пороговое значение.

Контракт моста:

- `changeValidatorSet(address newvalidatorset)` позволяет владельцу контракта изменить адрес контракта управления набором валидаторов.
- `addLiquidity() payable` позволяет владельцу контракта на правой стороне увеличить количество токенов, послав их вместе с вызовом метода.
- `updateLiquidityLimit(uint256 newlimit)` позволяет владельцу контракта на левой стороне синхронизировать лимит на передачу токенов из левой стороны с количеством токенов добавленных в контракт моста.
- `getLiquidityLimit()` позволяет получить какое максмиальное количество токенов может быть передано через мост.
- `commit(address recipient, uint256 amount, bytes32 id)` позволяет валидатору моста подтвердить факт инициации передачи токенов через мост.
- `() payable` позволяет пользователям блокчейн сети и другим контрактам инициировать передачу нативных токенов на другую сторону моста, просто отправив токены на адрес контракта моста. Испускает событие `bridgeActionInitiated(address recipient, uint256 amount)`.

Скрипт регистрации контрактов

Скрипт, производящий регистрацию (deployment) контракта в блокчейн сети, опирается на следующие переменные окружения, заданные перед запуском скрипта:

```
# Приватный ключ, который используется для подписи транзакции, развертывающей
# контракты моста. На соответствующем приватному ключу счете должен быть
# достаточный баланс для исполнения операции развертывания
PRIVKEY=cafecafe...cafecafe

# URL, по которому будут доступны узлы, предоставляющий web3 сервис для
# взаимодействия с блокчейном с левой или правой стороны моста
LEFT_RPCURL=https://sokol.poa.network
RIGHT_RPCURL=https://sokol.poa.network

# Цена за единицу газа, которая будет выставленна в транзакции, отправленные
# в блокчейн для левой или правой сторон моста
LEFT_GASPRICE=5000000000
RIGHT_GASPRICE=5000000000

# Адреса валидаторов моста
VALIDATORS=0xdeadbeaf...deadbeaf 0xbaddad...baddad

# Пороговое значение, определяющее какое количество валидаторов моста должны
# выслать подвтерждения, чтобы передача токенов через мост считалась достоверной
THRESHOLD=...
```

Запуск скрипта регистрации контрактов через контейнер:

```
docker run -ti --rm --env-file .env n2n-oracle /deployment/run.sh
```

Оракул

При запуске сервис оракула подразумевает в наличии следующих переменных окружения:

```
# Приватный ключ, соответствующий аккаунту валидатора моста
PRIVKEY=cafecafe...cafecafe

# URL, по которому будут доступны узлы, предоставляющий web3 сервис для
# взаимодействия с блокчейном с левой или правой стороны моста
LEFT_RPCURL=https://sokol.poa.network
RIGHT_RPCURL=https://sokol.poa.network

# Цена за единицу газа, которая будет выставленна в транзакции, отправленные
# в блокчейн для левой или правой сторон моста
LEFT_GASPRICE=5000000000
RIGHT_GASPRICE=5000000000

# Адреса контрактов моста
LEFT_ADDRESS=0x
RIGHT_ADDRESS=0x

# Номера блоков, в котором были обработаны транзакции, регистрирующие контракты
# моста в блокчейн с левой и правой стороны моста
LEFT_START_BLOCK=X
RIGHT_START_BLOCK=X

# Путь до директории, в которой могут располагаться данные оракула, которые
# должны быть доступны после перезагрузки сервиса. Данная переменная должна
# использоваться в docker-compose.yml в секции, предоставляющей контейнеру
# получить доступ к файловой системе машины, на которой контейнер запускается.
# Пример,
#    volumes:
#        - ${ORACLE_DATA}/db:/data
# где db -- директория на системе, запускающей контейнер
#     /data -- директория внутри контенера
ORACLE_DATA=/some/path
```

Оракул и все необходимые подсистемы должны автоматически стартануть после выполнения команды:

```
docker-compose up -d
```

Комментарии, связанные с проверяющей системой::

- Избегайте установки конкретного имени контейнера
- Не производите маппинг портов

Вот пример корректного `docker-compose.yml`:

```
version: "3.9"

services:
  oracle:
    image: n2n-oracle
    command: sh /wait-for.sh redis:5432 -- python oracle.py
    working_dir: /oracle
```

```
    env_file: .env

  redis:
    image: redis:4.0.5-alpine
    command: ["redis-server", "--appendonly", "yes", "--port", "5432"]
    hostname: redis
    volumes:
        - ${ORACLE_DATA}/redis:/data
```

Взаимодействие python с системой, выполняющей приемочное тестирование

1. Особенности системы запуска приемочного тестирования таковы, что следующий шаблон должен использоваться для подключения к RPC узлу:

```
from web3 import Web3, HTTPProvider

w3 = Web3(HTTPProvider('http://some.rpc.url/'))

from web3.middleware import geth_poa_middleware

w3.middleware_onion.inject(geth_poa_middleware, layer=0)
```

2. Можно из скрипта регистрации и оракула выводить сообщения в stderr – они будут доступны после завершения тестирования в GitLab CI/CD pipeline artifacts. Из python кода удобно использовать библиотеку `logger`:

```
import logging

logging.basicConfig(level=logging.INFO)

logging.info('Here to stderr')
```

US-001 Сборка docker образа

Описание: Я, как пользователь блокчейн системы, могу собрать docker образ, содержащий программное обеспечение моста.

Критерий оценивания (AC-001-01) : Сборка

---

1. Исходный код оракула и контрактов получен из git репозитория.
2. После запуска команды в директории с исходным кодом оракула:

   ```
   docker build -t n2n-oracle .
   ```

   Сборка docker образа выполняется успешно.
3. Запуск команды:

   ```
   docker images | grep n2n-oracle
   ```

   выводит строку с информацией о собранном контейнере.
4. Запуск команды:

   ```
   docker run -ti n2n-oracle ls -la /deployment/run.sh
   ```

выводит информацию о файле `/deployment/run.sh`, расположенном внутри контейнера

US-002 Регистрация контрактов

Описание: Я, как пользователь блокчейн системы, могу используя docker образ зарегистрировать контракты, необходимые для успешного функционирвоания моста.

Критерий оценивания (AC-002-01) : Регистрация контрактов

---

1. В директории `/some/path` находится `.env` файл со следующим содержимым:

   ```
   PRIVKEY=7e2426debde689e756c6290d855a1068aef06db7a2b076be5fe6f2eefc452ca1
   LEFT_RPCURL=https://sokol.poa.network
   RIGHT_RPCURL=https://data-seed-prebsc-1-s1.binance.org:8545/
   LEFT_GASPRICE=5000000000
   RIGHT_GASPRICE=5000000000
   VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
   ↪  0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
   ↪  0x61365C58E44A6Fc166897f4A30641dba82E606c0
   THRESHOLD=2
   ```

2. После запуска команды:

   ```
   docker run -ti --rm --env-file /some/path/.env n2n-oracle /deployment/run.sh
   ```

   в терминале выводится

   ```
   [LEFT] Validators Set deployed at 0x...
   [LEFT] Bridge deployed at 0x...
   [RIGHT] Validators Set deployed at 0x...
   [RIGHT] Bridge deployed at 0x...
   [LEFT] Bridge deployed at block ...
   [RIGHT] Bridge deployed at block ...
   ```

   Контрактов с указанными адресами не существовало в блокчейн базе данных до исполнения команды.

3. Вызов метода `getValidators` у контракта с адресом из строчки с номером 1 из вывода команды на шаге 2 возвращает массив состоящий из трех элементов:

   ```
   ["0x130930e3E3D30bF8F975a729e948CdCc212ECFBB",
   ↪  "0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70",
   ↪  "0x61365C58E44A6Fc166897f4A30641dba82E606c0"]
   ```

   Порядок элементов в массиве значения не имеет.

4. Вызов метода `getValidators` у контракта с адресом из строчки с номером 3 из вывода команды на шаге 2 возвращает массив состоящий из трех элементов:

   ```
   ["0x130930e3E3D30bF8F975a729e948CdCc212ECFBB",
   ↪  "0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70",
   ↪  "0x61365C58E44A6Fc166897f4A30641dba82E606c0"]
   ```

   Порядок элементов в массиве значения не имеет.

5. Аккаунт, которому соответствует приватный ключ `0x7e2426debde689e756c6290d855a1068aef06db7a2b076be5fe6f2eefc452ca1`, отправляет транзакцию, вызывающую метод `addLiquidity` у контракта с адре-

сом из строчки с номером 4 из вывода команды на шаге 2. Данная транзакция также пересылает 100 нативных токенов. Транзакция исполняется успешно.

6. Аккаунт, которому соответствует приватный ключ `0x7e2426debde689e756c6290d855a1068aef06db7a2b076be5fe6f2eefc452ca1`, отправляет транзакцию, вызывающую метод `updateLiquidityLimit` у контракта с адресом из строчки с номером 2 из вывода команды на шаге 2 с параметром `(100000000000000000000)`

   Транзакция исполняется успешно.

7. Аккаунт, которому соответствует приватный ключ `0x7e2426debde689e756c6290d855a1068aef06db7a2b076be5fe6f2eefc452ca1`, отправляет транзакцию, вызывающую метод `changeValidatorSet` у контракта с адресом из строчки с номером 2 из вывода команды на шаге 2 с параметром `("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e")`

   Транзакция исполняется успешно.

8. Аккаунт, которому соответствует приватный ключ `0x7e2426debde689e756c6290d855a1068aef06db7a2b076be5fe6f2eefc452ca1`, отправляет транзакцию, вызывающую метод `changeValidatorSet` у контракта с адресом из строчки с номером 4 из вывода команды на шаге 2 с параметром `("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e")`

   Транзакция исполняется успешно.

US-003 Контракт управления набором валидаторов: добавление валидаторов

Описание: Я, как администратор контракта управления набором валидаторов, через прямое взаимодействие с контрактом могу добавлять новых валидаторов.

Критерий оценивания (AC-003-01) : Добавление валидатора

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `addValidator` с параметром `("0x61365C58E44A6Fc166897f4A30641dba82E606c0")`

   Транзакция исполняется успешно.

3. Вызов метода `getValidators` возвращает массив состоящий из трех элементов:

```
["0x130930e3E3D30bF8F975a729e948CdCc212ECFBB",
↪ "0x61365C58E44A6Fc166897f4A30641dba82E606c0",
↪ "0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70"]
```

   Порядок элементов в массиве значения не имеет.

Критерий оценивания (AC-003-02) : Добавление существующего валидатора

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `addValidator` с параметром

   `("0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70")`

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

Критерий оценивания (AC-003-03) : Несанкционированное добавление валидатора

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
```

2. Аккаунт с адресом `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `addValidator` с параметром

   `("0x61365C58E44A6Fc166897f4A30641dba82E606c0")`

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-004 Контракт управления набором валидаторов: удаление валидаторов

Описание: Я, как администратор контракта управления набором валидаторов, через прямое взаимодействие с контрактом могу удалять существующих валидаторов.

Критерий оценивания (AC-004-01) : Удаление валидатора

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `removeValidator` с параметром

   `("0x61365C58E44A6Fc166897f4A30641dba82E606c0")`

   Транзакция исполняется успешно.
3. Вызов метода `getValidators` возвращает массив состоящий из двух элементов:

   ```
   ["0x130930e3E3D30bF8F975a729e948CdCc212ECFBB",
   ↪ "0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70"]
   ```

   Порядок элементов в массиве значения не имеет.

Критерий оценивания (AC-004-02) : Удаление несуществующего валидатора

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=1
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `removeValidator` с параметром

   `("0x61365C58E44A6Fc166897f4A30641dba82E606c0")`

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта вполнилась команда `revert`.

Критерий оценивания (AC-004-03) : Изменение количества валидаторов ниже порогового значения

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=2
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `removeValidator` с параметром

   `("0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70")`

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

Критерий оценивания (AC-004-04) : Добавление валидатора после удаления

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪  0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪  0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪  0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `removeValidator` с параметром

   `("0x61365C58E44A6Fc166897f4A30641dba82E606c0")`

   Транзакция исполняется успешно.
3. Вызов метода `getValidators` возвращает массив состоящий из двух элементов:

   ```
   ["0x130930e3E3D30bF8F975a729e948CdCc212ECFBB",
   ↪  "0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70"]
   ```

   Порядок элементов в массиве значения не имеет.
4. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `addValidator` с параметром

   `("0x61365C58E44A6Fc166897f4A30641dba82E606c0")`

   Транзакция исполняется успешно.
5. Вызов метода `getValidators` возвращает массив состоящий из трех элементов:

   ```
   ["0x130930e3E3D30bF8F975a729e948CdCc212ECFBB",
   ↪  "0x61365C58E44A6Fc166897f4A30641dba82E606c0",
   ↪  "0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70"]
   ```

   Порядок элементов в массиве значения не имеет.

Критерий оценивания (AC-004-05) : Несанкционированное удаление валидатора

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

2. Аккаунт с адресом `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `removeValidator` с параметром

   `("0x61365C58E44A6Fc166897f4A30641dba82E606c0")`

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-005 Контракт управления набором валидаторов: изменение порогового значения

Описание: Я, как администратор контракта управления набором валидаторов, через прямое взаимодействие с контрактом могу изменять порогове значение, определяющее необходимое количество подвтерждений от разных валидаторов, после которого считается запрос на межблокчейновую передачу средств достоверным.

Критерий оценивания (AC-005-01) : Изменение порогового значения

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `changeThreshold` с параметром

   `(3)`

   Транзакция исполняется успешно.
3. Вызов метода `getThreshold` возвращает значение `3`.

Критерий оценивания (AC-005-02) : Изменение порогового значения выше количества валидаторов

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=3
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `changeThreshold` с параметром

   `(4)`

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

Критерий оценивания (AC-005-03) : Несанкционированное изменение значения

---

1. Контракт управления набором валидаторов зарегистрирован в блокчейн сети скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=3
```

2. Аккаунт с адресом `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `changeThreshold` с параметром

   `(2)`

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-006 Контракт моста: добавление ликвидности на правую сторону моста

Описание: Я, как администратор контракта моста, через прямое взаимодействие с контрактом могу инициализировать контракт на правой стороне моста добавлением на него ликвидности.

Критерий оценивания (AC-006-01) : Начальное формирование ликвидности

---

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `addLiquidity`. Данная транзакция также пересылает 100 нативных токенов. Транзакция исполняется успешно.
3. Баланс контракта моста с правой стороны увеличивается до 100 токенов.
4. Вызов метода `getLiquidityLimit` возвращает: `10000000000000000000`.

Критерий оценивания (AC-006-02) : Добавление ликвидности

---

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪  0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `addLiquidity`. Данная транзакция также пересылает 100 нативных токенов. Транзакция исполняется успешно.
3. Баланс контракта моста с правой стороны увеличивается до 100 токенов.
4. Вызов метода `getLiquidityLimit` возвращает: `10000000000000000000`.
5. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `addLiquidity`. Данная транзакция также пересылает 50 нативных токенов. Транзакция исполняется успешно.
6. Баланс контракта моста с правой стороны увеличивается до 150 токенов.
7. Вызов метода `getLiquidityLimit` возвращает: `15000000000000000000`.

Критерий оценивания (AC-006-03) : Некорректное добавление ликвидности

---

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪  0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `addLiquidity` с транзакционной суммой в 0 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

Критерий оценивания (AC-006-04) : Несанкционированное добавление ликвидности

---

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
```

2. Аккаунт с адресом `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `addLiquidity` с транзакционной суммой в 100 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-007 Контракт моста: установка верхнего лимита на левой стороне моста

Описание: Я, как администратор контракта моста, через прямое взаимодействие с контрактом могу инициализировать контракт на левой стороне моста установкой верхнего лимита на количество пересылаемых токенов из левой стороны.

Критерий оценивания (AC-007-01) : Начальная установка верхнего лимита

---

1. Контракт моста зарегистрирован в блокчейн сети с левой стороны скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `updateLiquidityLimit` с параметром `(10000000000000000000)`

   Транзакция исполняется успешно.
3. Вызов метода `getLiquidityLimit` возвращает: `10000000000000000000`.

Критерий оценивания (AC-007-02) : Изменение верхнего лимита

---

1. Контракт моста зарегистрирован в блокчейн сети с левой стороны скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `updateLiquidityLimit` с параметром `(10000000000000000000)`

   Транзакция исполняется успешно.
3. Вызов метода `getLiquidityLimit` возвращает: `10000000000000000000`.
4. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию, вызывающую метод `updateLiquidityLimit` с параметром `(15000000000000000000)`

Транзакция исполняется успешно.

5. Вызов метода `getLiquidityLimit` возвращает: `150000000000000000000`.

Критерий оценивания (AC-007-03) : Несанкционированное изменение верхнего лимита

---

1. Контракт моста зарегистрирован в блокчейн сети с левой стороны скриптом при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪  0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
```

2. Аккаунт с адресом `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции, вызывающей метод `updateLiquidityLimit` с параметром `(100000000000000000000)`

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-008 Контракт моста: отправка токенов на контракт на левой стороне

Описание: Я, как пользователь моста, могу инициировать посылку нативных токенов из левой стороны моста в правую сторону, послав их на адрес контракта моста.

Критерий оценивания (AC-008-01) : Запрос посылки токенов

---

1. Контракт моста зарегистрирован в блокчейн сети с левой стороны по адресу `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`.
2. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 50 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
3. Администратор моста инициирует контракт моста, отправив транзакцию, вызывающую метод `updateLiquidityLimit` с параметром `(100000000000000000000)`
4. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 50 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов. В выписке транзакции есть событие:

```
bridgeActionInitiated("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e",
↪  50000000000000000000)
```

5. Вызов метода `getLiquidityLimit` возвращает: `50000000000000000000`.

6. Пользователь, чей приватный ключ соответствует адресу `0x61365C58E44A6Fc166897f4A30641` отправляет 40 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 40 токенов. Баланс моста увеличивается на 40 токенов. В выписке транзакции есть событие:

   ```
   bridgeActionInitiated("0x61365C58E44A6Fc166897f4A30641dba82E606c0",
   ↪  40000000000000000000)
   ```

7. Вызов метода `getLiquidityLimit` возвращает: `10000000000000000000`.
8. Пользователь, чей приватный ключ соответствует адресу `0x61365C58E44A6Fc166897f4A30641` отправляет 5 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 5 токенов. Баланс моста увеличивается на 5 токенов. В выписке транзакции есть событие:

   ```
   bridgeActionInitiated("0x61365C58E44A6Fc166897f4A30641dba82E606c0",
   ↪  50000000000000000000)
   ```

9. Вызов метода `getLiquidityLimit` возвращает: `50000000000000000000`.
10. Пользователь, чей приватный ключ соответствует адресу `0x61365C58E44A6Fc166897f4A30641` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 10 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
11. Администратор моста отправляет транзакцию, вызывающую метод `updateLiquidityLimit` с параметром

    ```
    (150000000000000000000)
    ```

    Транзакция исполняется успешно.
12. Пользователь, чей приватный ключ соответствует адресу `0x61365C58E44A6Fc166897f4A30641` отправляет 10 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 10 токенов. Баланс моста увеличивается на 10 токенов. В выписке транзакции есть событие:

    ```
    bridgeActionInitiated("0x61365C58E44A6Fc166897f4A30641dba82E606c0",
    ↪  10000000000000000000)
    ```

13. Вызов метода `getLiquidityLimit` возвращает: `50000000000000000000`.
14. Пользователь, чей приватный ключ соответствует адресу `0x61365C58E44A6Fc166897f4A30641` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 0 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-009 Контракт моста: отправка подтверждения от валидатора на правую сторону

Описание: Я, как валидатор моста, могу подтвердить посылку нативных токенов из левой стороны моста в правую сторону, послав соответствующее сообщение в контракт моста на правой стороне.

Критерий оценивания (AC-009-01) : 1 of 1

---

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны по адресу `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=1
```

2. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающей метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 50000000000000000000,
   ↪ "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

3. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `addLiquidity`. Данная транзакция также пересылает 100 нативных токенов. Транзакция исполняется успешно.

4. Вызов метода `getLiquidityLimit` у контракта `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10` возвращает: `100000000000000000000`.

5. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 50000000000000000000,
   ↪ "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. У пользователя `0xf712a82DD8e2Ac923299193e9d6dAEda2d5` после исполнения транзакции становится на 50 нативных токенов больше. Баланс контракта моста меняется до 50 нативных токенов.

6. Вызов метода `getLiquidityLimit` у контракта `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10` возвращает: `50000000000000000000`.

7. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающей метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 51000000000000000000,
   ↪ "0xbe4da129fc927472b7b5ab00c04a0c19124fd3915a51650e672f332dcc8d2f3d")
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

8. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

   ```
   ("0xC1d9971bcd39bd96f64787765221be86d1a928e5", 50000000000000000000,
   ↪ "0xbe4da129fc927472b7b5ab00c04a0c19124fd3915a51650e672f332dcc8d2f3d")
   ```

   Транзакция исполняется успешно. У пользователя `0xC1d9971bcd39bd96f64787765221be86d1a` после исполнения транзакции становится на 5 нативных токенов больше. Баланс контракта моста меняется до 45 нативных токенов.

9. Вызов метода `getLiquidityLimit` у контракта `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10` возвращает: `45000000000000000000`.
10. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающей метод `commit` с параметром

    ```
    ("0x737b791E227f0395d39F7496198F9EC751A57Dc0", 5000000000000000000,
    ↪  "0x5840f306777b4982348202e3ff40c17e6ae555b228d5f0290aa23c23d36423d1")
    ```

    Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
11. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

    ```
    ("0xC1d9971bcd39bd96f64787765221be86d1a928e5", 5000000000000000000,
    ↪  "0x5d46b2de62ae6807ad8a2cf1342b2b7bf547328f23a5439511297798b9cadfbc")
    ```

    Транзакция исполняется успешно. У пользователя `0xC1d9971bcd39bd96f64787765221be86d1a` после исполнения транзакции становится на 5 нативных токенов больше. Баланс контракта моста меняется до 40 нативных токенов.
12. Вызов метода `getLiquidityLimit` у контракта `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10` возвращает: `40000000000000000000`.
13. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающей метод `commit` с параметром

    ```
    ("0xC1d9971bcd39bd96f64787765221be86d1a928e5", 5000000000000000000,
    ↪  "0x5d46b2de62ae6807ad8a2cf1342b2b7bf547328f23a5439511297798b9cadfbc")
    ```

    Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

Критерий оценивания (AC-009-02) : N of M

---

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны по адресу `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪  0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪  0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪  0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

2. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `addLiquidity`. Данная транзакция также пересылает 100 нативных токенов. Транзакция исполняется успешно.
3. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

```
("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 50000000000000000000,
↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
```

Транзакция исполняется успешно. Баланс у пользователя 0xf712a82DD8e2Ac923299193e9d6d не меняется. Баланс контракта моста не меняется.

4. Аккаунт с адресом 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB вызывает JSON RPC метод eth_estimateGas для оценки потребления газа перед посылкой транзакции на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающей метод commit с параметром

```
("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 50000000000000000000,
↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
```

Вызов eth_estimateGas возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда revert.

5. Аккаунт с адресом 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB отправляет транзакцию на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающую метод commit с параметром

```
("0xBeC92cb65ad0514eb8847a0c48Bd7ee4f6E88EeF ", 15000000000000000000,
↪  "0xeb096ea9048b5287eba951adda5b20a8eae9bf9dccc8ff34a44f984b0ed44fd2")
```

Транзакция исполняется успешно. Баланс у пользователя 0xBeC92cb65ad0514eb8847a0c48Bd не меняется. Баланс контракта моста не меняется.

6. Аккаунт с адресом 0x61365C58E44A6Fc166897f4A30641dba82E606c0 отправляет транзакцию на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающую метод commit с параметром

```
("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 50000000000000000000,
↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
```

Транзакция исполняется успешно. У пользователя 0xf712a82DD8e2Ac923299193e9d6dAEda2d5 после исполнения транзакции становится на 50 нативных токенов больше. Баланс контракта моста меняется до 50 нативных токенов.

7. Аккаунт с адресом 0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70 отправляет транзакцию на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающую метод commit с параметром

```
("0xBeC92cb65ad0514eb8847a0c48Bd7ee4f6E88EeF ", 15000000000000000000,
↪  "0xeb096ea9048b5287eba951adda5b20a8eae9bf9dccc8ff34a44f984b0ed44fd2")
```

Транзакция исполняется успешно. У пользователя 0xBeC92cb65ad0514eb8847a0c48Bd7ee4f6 после исполнения транзакции становится на 15 нативных токенов больше. Баланс контракта моста меняется до 35 нативных токенов.

8. Аккаунт с адресом 0x61365C58E44A6Fc166897f4A30641dba82E606c0 вызывает JSON RPC метод eth_estimateGas для оценки потребления газа перед посылкой транзакции на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающей метод commit с параметром

```
("0xBeC92cb65ad0514eb8847a0c48Bd7ee4f6E88EeF", 15000000000000000000,
↪  "0xeb096ea9048b5287eba951adda5b20a8eae9bf9dccc8ff34a44f984b0ed44fd2")
```

Вызов eth_estimateGas возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда revert.

US-010 Контракт моста: отправка токенов на контракт на правой стороне

Описание: Я, как пользователь моста, могу инициировать посылку нативных токенов из правой стороны моста в левую сторону, послав их на адрес контракта моста.

Критерий оценивания (AC-010-01) : Запрос посылки токенов

---

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны по адресу `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` при следующем содержимом переменных окружения:

```
VALIDATORS=0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=1
```

2. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 5 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
3. Администратор моста инициирует контракт моста, отправив транзакцию, вызывающую метод `addLiquidity`. Данная транзакция также пересылает 200 нативных токенов. Баланс моста увеличивается до 200 токенов.
4. Вызов метода `getLiquidityLimit` возвращает: `200000000000000000000`.
5. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 5 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
6. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 100000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. У пользователя `0xf712a82DD8e2Ac923299193e9d6dAEda2d5` после исполнения транзакции становится на 100 нативных токенов больше. Баланс контракта моста меняется до 100 нативных токенов.
7. Вызов метода `getLiquidityLimit` возвращает: `100000000000000000000`.
8. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 5 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 5 токенов. Баланс моста увеличивается до 105 токенов. В выписке транзакции есть событие:

   ```
   bridgeActionInitiated("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e",
   ↪  5000000000000000000)
   ```

9. Вызов метода `getLiquidityLimit` возвращает: `105000000000000000000`.
10. Пользователь, чей приватный ключ соответствует адресу `0xf712a82DD8e2Ac923299193e9d6dA` отправляет 10 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 10 токенов. Баланс моста увеличивается до 115 токенов. В выписке транзакции есть событие:

    ```
    bridgeActionInitiated("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd",
    ↪  10000000000000000000)
    ```

11. Вызов метода `getLiquidityLimit` возвращает: `11500000000000000000`.
12. Пользователь, чей приватный ключ соответствует адресу `0xf712a82DD8e2Ac923299193e9d6dA` отправляет 15 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 15 токенов. Баланс моста увеличивается до 130 токенов. В выписке транзакции есть событие:

    ```
    bridgeActionInitiated("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd",
    ↪  15000000000000000000)
    ```

13. Вызов метода `getLiquidityLimit` возвращает: `13000000000000000000`.
14. Пользователь, чей приватный ключ соответствует адресу `0xf712a82DD8e2Ac923299193e9d6dA` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 75 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
15. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

    ```
    ("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e", 50000000000000000000,
    ↪  "0xc05af7a8256ab341d0f51a464e9d84dbde0efeb1f36b48ae815f179410e7a04e")
    ```

    Транзакция исполняется успешно. У пользователя `0x79dD14623c4D33413c0c28fDAbC2285Fdb` после исполнения транзакции становится на 50 нативных токенов больше. Баланс контракта моста меняется до 80 нативных токенов.
16. Вызов метода `getLiquidityLimit` возвращает: `80000000000000000000`.
17. Пользователь, чей приватный ключ соответствует адресу `0xf712a82DD8e2Ac923299193e9d6dA` отправляет 75 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 75 токенов. Баланс моста увеличивается до 155 токенов. В выписке транзакции есть событие:

    ```
    bridgeActionInitiated("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd",
    ↪  75000000000000000000)
    ```

18. Вызов метода `getLiquidityLimit` возвращает: `15500000000000000000`.
19. Пользователь, чей приватный ключ соответствует адресу `0xf712a82DD8e2Ac923299193e9d6dA` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 0 нативных токенов. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-011 Контракт моста: отправка подтверждения от валидатора на левую сторону

Описание: Я, как валидатор моста, могу подтвердить посылку нативных токенов из правой стороны моста в левую сторону, послав соответствующее сообщение в контракт моста на левой стороне.

Критерий оценивания (AC-011-01) : 1 of 1

---

1. Контракт моста зарегистрирован в блокчейн сети с левой стороны по адресу

0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e при следующем содержимом переменных окружения:

```
VALIDATORS=0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=1
```

2. Аккаунт с адресом 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB вызывает JSON RPC метод eth_estimateGas для оценки потребления газа перед посылкой транзакции на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающей метод commit с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 50000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Вызов eth_estimateGas возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда revert.

3. Администратор моста инициирует контракт моста, отправляет транзакцию на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающую метод updateLiquidityLimit с параметром

   ```
   (100000000000000000000)
   ```

4. Вызов метода getLiquidityLimit у контракта 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10 возвращает: 100000000000000000000.

5. Аккаунт с адресом 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB вызывает JSON RPC метод eth_estimateGas для оценки потребления газа перед посылкой транзакции на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающей метод commit с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 50000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Вызов eth_estimateGas возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда revert.

6. Пользователь, чей приватный ключ соответствует адресу 0x79dD14623c4D33413c0c28fDAbC22 отправляет 75 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 75 токенов. Баланс моста увеличивается до 75 токенов. В выписке транзакции есть событие:

   ```
   bridgeActionInitiated("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e",
   ↪  75000000000000000000)
   ```

7. Вызов метода getLiquidityLimit у контракта 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10 возвращает: 25000000000000000000.

8. Аккаунт с адресом 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB отправляет транзакцию на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающую метод commit с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 50000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. У пользователя 0xf712a82DD8e2Ac923299193e9d6dAEda2d5 после исполнения транзакции становится на 50 нативных токенов больше. Баланс контракта моста меняется до 25 нативных токенов.

9. Вызов метода getLiquidityLimit у контракта 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10 возвращает: 75000000000000000000.

10. Аккаунт с адресом 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB вызывает JSON RPC метод eth_estimateGas для оценки потребления газа перед посылкой транзакции на контракт 0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e, вызывающей метод commit с параметром

```
("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 26000000000000000000,
↪  "0xbe4da129fc927472b7b5ab00c04a0c19124fd3915a51650e672f332dcc8d2f3d")
```

Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

11. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

    ```
    ("0xC1d9971bcd39bd96f64787765221be86d1a928e5", 5000000000000000000,
    ↪  "0xbe4da129fc927472b7b5ab00c04a0c19124fd3915a51650e672f332dcc8d2f3d")
    ```

    Транзакция исполняется успешно. У пользователя `0xC1d9971bcd39bd96f64787765221be86d1a` после исполнения транзакции становится на 5 нативных токенов больше. Баланс контракта моста меняется до 20 нативных токенов.

12. Вызов метода `getLiquidityLimit` у контракта `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10` возвращает: `80000000000000000000`.

13. Аккаунт с адресом `0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающей метод `commit` с параметром

    ```
    ("0x737b791E227f0395d39F7496198F9EC751A57Dc0", 5000000000000000000,
    ↪  "0x5840f306777b4982348202e3ff40c17e6ae555b228d5f0290aa23c23d36423d1")
    ```

    Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

14. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

    ```
    ("0xC1d9971bcd39bd96f64787765221be86d1a928e5", 5000000000000000000,
    ↪  "0x5d46b2de62ae6807ad8a2cf1342b2b7bf547328f23a5439511297798b9cadfbc")
    ```

    Транзакция исполняется успешно. У пользователя `0xC1d9971bcd39bd96f64787765221be86d1a` после исполнения транзакции становится на 5 нативных токенов больше. Баланс контракта моста меняется до 15 нативных токенов.

15. Вызов метода `getLiquidityLimit` у контракта `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD10` возвращает: `85000000000000000000`.

16. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающей метод `commit` с параметром

    ```
    ("0xC1d9971bcd39bd96f64787765221be86d1a928e5", 5000000000000000000,
    ↪  "0x5d46b2de62ae6807ad8a2cf1342b2b7bf547328f23a5439511297798b9cadfbc")
    ```

    Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

Критерий оценивания (AC-011-02) : N of M

---

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны по адресу `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` при следующем содержимом переменных окружения:

```
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

2. Администратор моста инициирует контракт моста, отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `updateLiquidityLimit` с параметром

   `(20000000000000000000)`

3. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 100 нативных токенов на адрес контракта моста. Транзакция исполняется успешно. Баланс пользователя уменьшается на 100 токенов. Баланс моста увеличивается до 100 токенов. В выписке транзакции есть событие:

   ```
   bridgeActionInitiated("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e",
   ↪ 100000000000000000000)
   ```

4. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 5000000000000000000,
   ↪ "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. Баланс у пользователя `0xf712a82DD8e2Ac923299193e9d6d` не меняется. Баланс контракта моста не меняется.

5. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающей метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 5000000000000000000,
   ↪ "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

6. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

   ```
   ("0xBeC92cb65ad0514eb8847a0c48Bd7ee4f6E88EeF ", 15000000000000000000,
   ↪ "0xeb096ea9048b5287eba951adda5b20a8eae9bf9dccc8ff34a44f984b0ed44fd2")
   ```

   Транзакция исполняется успешно. Баланс у пользователя `0xBeC92cb65ad0514eb8847a0c48Bd` не меняется. Баланс контракта моста не меняется.

7. Аккаунт с адресом `0x61365C58E44A6Fc166897f4A30641dba82E606c0` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 5000000000000000000,
   ↪ "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. У пользователя `0xf712a82DD8e2Ac923299193e9d6dAEda2d5` после исполнения транзакции становится на 50 нативных токенов больше. Баланс контракта моста меняется до 50 нативных токенов.

8. Аккаунт с адресом `0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

```
("0xBeC92cb65ad0514eb8847a0c48Bd7ee4f6E88EeF ", 15000000000000000000,
↪  "0xeb096ea9048b5287eba951adda5b20a8eae9bf9dccc8ff34a44f984b0ed44fd2")
```

Транзакция исполняется успешно. У пользователя `0xBeC92cb65ad0514eb8847a0c48Bd7ee4f6` после исполнения транзакции становится на 15 нативных токенов больше. Баланс контракта моста меняется до 35 нативных токенов.

9. Аккаунт с адресом `0x61365C58E44A6Fc166897f4A30641dba82E606c0` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающей метод `commit` с параметром

```
("0xBeC92cb65ad0514eb8847a0c48Bd7ee4f6E88EeF ", 15000000000000000000,
↪  "0xeb096ea9048b5287eba951adda5b20a8eae9bf9dccc8ff34a44f984b0ed44fd2")
```

Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-012 Оракул: пересылка события об отправленном токене на правую сторону

Описание: Я, как пользователь моста, могу инициировать посылку нативных токенов, послав их на адрес контракта моста на левой стороне, и после некоторого количества времени получить соответствующее количесво нативных токенов на правой стороне.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`
2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪  0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=1
```

3. Установлена ликвидность моста в 100 токенов.
4. При содежимом файла `.env`:

```
PRIVKEY=<private key corresponding to
↪  0x130930e3E3D30bF8F975a729e948CdCc212ECFBB>
LEFT_RPCURL=https://sokol.poa.network
LEFT_ADDRESS=0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441
```

```
LEFT_START_BLOCK=17290597
LEFT_GASPRICE=5000000000
RIGHT_RPCURL=https://data-seed-prebsc-1-s1.binance.org:8545/
RIGHT_ADDRESS=0x159AE4d7012B18A0dE4e390714a6efa036487f0b
RIGHT_START_BLOCK=5847797
RIGHT_GASPRICE=10000000000
ORACLE_DATA=/some/path
```

и пустой директории `/some/path` запускается команда:

```
$ docker-compose up -d
```

При использовании `docker ps -a` видно, что, как минимум, один контейнеры основывается на образе `n2n-oracle`.

Критерий оценивания (AC-012-01) : Пересылка токенов

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 10 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 10 токенов. Баланс моста увеличивается до 10 токенов.
2. В сети `RIGHT` в одном из блоков ‘B*‘, выпущенном не позднее чем через 15 секунд после появления блока `B`, есть транзакция от отправителя `0x130930e3E3D30bF8F975a729` При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `RIGHT` на момент завершения блока ‘B*‘ видно, что на 10 нативных токенов стало больше. При аналогичном запросе, баланс контракта моста – 90 нативных токенов.
3. Пользователь, чей приватный ключ соответствует адресу `0x97bdb4071396b7f60b65e0eb62ce2` отправляет 15 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B+n` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 15 токенов. Баланс моста увеличивается до 25 токенов.
4. В сети `RIGHT` в одном из блоков ‘B**‘, выпущенном не позднее чем через 15 секунд после появления блока `B+n`, есть транзакция от отправителя `0x130930e3E3D30bF8F975a7` При опросе баланса пользователя `0x97bdb4071396b7f60b65e0eb62ce212a699f4b08` в сети `RIGHT` на момент завершения блока ‘B**‘ видно, что на 15 нативных токенов стало больше. При аналогичном запросе, баланс контракта моста – 75 нативных токенов.
5. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 20 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B+n+m` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 20 токенов. Баланс моста увеличивается до 45 токенов.
6. В сети `RIGHT` в одном из блоков ‘B***‘, выпущенном не позднее чем через 15 секунд после появления блока `B+n+m`, есть транзакция от отправителя `0x130930e3E3D30bF8F975` При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `RIGHT` на момент завершения блока ‘B***‘ видно, что на 20 нативных токенов стало больше. При аналогичном запросе, баланс контракта моста – 55 нативных токенов.

Критерий оценивания (AC-012-02) : Обработка запросов на пересылку токенов, инициированных до запуска оракула

---

1. Перед самым первым запуском оракула пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` отправляет 10 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 10 токенов. Баланс моста увеличивается до 10 токенов.
2. Запускается оракул после того (конкретный промежуток времени не является существенным), как в `LEFT` появился блок `B`.
3. В сети `RIGHT` в одном из блоков 'B*', ввыпущенном не позднее чем через 30 секунд после запуска оракула, есть транзакция от отправителя `0x130930e3E3D30bF8F975a729e9` При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `RIGHT` на момент завершения блока 'B*' видно, что на 10 нативных токенов стало больше. При аналогичном запросе, баланс контракта моста – 90 нативных токенов.

US-013 Оракул: пересылка события об отправленном токене на левую сторону

Описание: Я, как пользователь моста, могу инициировать посылку нативных токенов, послав их на адрес контракта моста на правой стороне, и после некоторого количества времени получить соответствующее количесво нативных токенов на левой стороне.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`
2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪  0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=1
```

3. Установлена ликвидность моста в 100 токенов.
4. При содежимом файла `.env`:

```
PRIVKEY=<private key corresponding to
↪  0x130930e3E3D30bF8F975a729e948CdCc212ECFBB>
LEFT_RPCURL=https://sokol.poa.network
LEFT_ADDRESS=0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441
LEFT_START_BLOCK=17290597
LEFT_GASPRICE=5000000000
RIGHT_RPCURL=https://data-seed-prebsc-1-s1.binance.org:8545/
RIGHT_ADDRESS=0x159AE4d7012B18A0dE4e390714a6efa036487f0b
RIGHT_START_BLOCK=5847797
RIGHT_GASPRICE=10000000000
ORACLE_DATA=/some/path
```

и пустой директории `/some/path` запускается команда:

```
$ docker-compose up -d
```

При использовании `docker ps -a` видно, что, как минимум, один контейнеры основывается на образе `n2n-oracle`.

Критерий оценивания (AC-013-01) : Пересылка токенов

---

1. Пользователь, чей приватный ключ соответствует адресу `0x8bC8c4b73a6FcbA35e26817c9dbf8` отправляет 100 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 100 токенов. Через некоторое время баланс пользователя `0x8bC8c4b73a6FcbA35e26817c9dbf86A85913Db33` в сети `RIGHT` увеличивается на 100 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 10 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 10 токенов. Баланс моста увеличивается до 10 токенов.
3. В сети `LEFT` в одном из блоков ‘B*‘, выпущенном не позднее чем через 15 секунд после появления блока `B`, есть транзакция от отправителя `0x130930e3E3D30bF8F975a729e948` При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `LEFT` на момент завершения блока ‘B*‘ видно, что на 10 нативных токенов стало больше. При аналогичном запросе, баланс контракта моста – 90 нативных токенов.
4. Пользователь, чей приватный ключ соответствует адресу `0x97bdb4071396b7f60b65e0eb62ce2` отправляет 15 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция входит в блок `B+n` и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 15 токенов. Баланс моста увеличивается до 15 токенов.
5. В сети `LEFT` в одном из блоков ‘B**‘, выпущенном не позднее чем через 15 секунд после появления блока `B+n`, есть транзакция от отправителя `0x130930e3E3D30bF8F975a7` При опросе баланса пользователя `0x97bdb4071396b7f60b65e0eb62ce212a699f4b08` в сети `LEFT` на момент завершения блока ‘B**‘ видно, что на 15 нативных токенов стало больше. При аналогичном запросе, баланс контракта моста – 75 нативных токенов.
6. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 20 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`.

Транзакция входит в блок `B+n+m` и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 20 токенов. Баланс моста увеличивается до 45 токенов.

7. В сети `LEFT` в одном из блоков ‘B***‘, выпущенном не позднее чем через 15 секунд после появления блока `B+n+m`, есть транзакция от отправителя `0x130930e3E3D30bF8F975`
При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `LEFT` на момент завершения блока ‘B***‘ видно, что на 20 нативных токенов стало больше. При аналогичном запросе, баланс контракта моста – 55 нативных токенов.

US-014 Оракул: пересылка события об отправленном токене при одновременной работе нескольких оракулов

Описание: Я, как пользователь моста, могу доверять мосту пересылку средств из одной блокчейн сети в другу, поскольку решение об подверждении пересылки принимается децентрализовано: нет единой точки отказа и цензурирования.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12’500’000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12’500’000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`
2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
PRIVKEY=<private key corresponding to
↪ 0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e>
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

3. Установлена ликвидность моста в 1000 токенов.
4. Запущены три оракула. В `PRIVKEY` `.env`-файла каждого оракула указан свой собственный приватный ключ, который соответствует одному из адресов, перечисленных в `VALIDATORS` при регистрации контрактов моста. Порядок запуска оракулов не определен. Промежутки времени между запусками оракулов не определены.

Критерий оценивания (AC-014-01) : Пересылка токенов

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3`
   отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`
   Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
   В сети `RIGHT` в одном или нескольких блоках, выпущенном не позднее чем через 15 секунд после появления блока `B`, есть транзакции от двух или трех оракулов. При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `RIGHT` на момент завершения блока, в который включена последняя успешная транзакция от оракулов, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 0 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3`
   отправляет `X` ($0.001 < X < 50$) нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6`
   Транзакция входит в блок ‘B*‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на `X` токенов. Баланс моста увеличивается на `X` токенов.
   В сети `LEFT` в одном или нескольких блоках, выпущенном не позднее чем через 15 секунд после появления блока ‘B*‘, есть транзакции от двух или трех оракулов.. При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `LEFT` на момент завершения блока, в который включена последняя успешная транзакция от оракулов, видно, что на `X` нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на `X` нативных токенов.
3. Пользователь, чей приватный ключ соответствует адресу `0x97bdb4071396b7f60b65e0eb62ce2`
   отправляет `Y` ($0.001 < Y < 50$) нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103`
   Транзакция входит в блок ‘B**‘ и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на `Y` токенов. Баланс моста увеличивается на `Y` токенов.
   В сети `RIGHT` в одном или нескольких блоках, выпущенном не позднее чем через 15 секунд после появления блока ‘B**‘, есть транзакции от двух или трех оракулов. При опросе баланса пользователя `0x97bdb4071396b7f60b65e0eb62ce212a699f4b08` в сети `RIGHT` на момент завершения блока, в который включена последняя успешная транзакция от оракулов, видно, что на `Y` нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на `Y` нативных токенов.

US-015 Оракул: Работа оракула с неверным приватным ключом

Описание: Я, как администратор системы, где запущен оракул, могу быть уверен, что оракул автоматически начнет работать, как только соответствующий валидатор будет добавлен в контракт управления набором валидаторов, а до этого времени не будет тратить средства на отправку транзакций, которые будет отклонены в любом случае.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:

- Сеть LEFT:
  - максимальное количество газа, которое может быть потрачено транзакциями в блоке 12’500’000
  - время выпуска блоков в среднем раз в 5 секунд
  - контракт моста находится по адресу 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c
- Сеть RIGHT:
  - максимальное количество газа, которое может быть потрачено транзакциями в блоке 12’500’000
  - время выпуска блоков в среднем раз в 5 секунд
  - контракт моста находится по адресу 0x159AE4d7012B18A0dE4e390714a6efa036487

2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
VALIDATORS=0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=1
```

3. Установлена ликвидность моста в 2000 токенов.
4. Запущен один оракул. В PRIVKEY .env-файла оракула указан свой приватный ключ, который соответствует адресу 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB.

Критерий оценивания (AC-015-01) : Смена валидатора во время работы оракула

---

1. Пользователь, чей приватный ключ соответствует адресу 0x5580ba66f8d6dc71adb0ca1d1c6b3
отправляет 1000 нативных токенов на адрес 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441
Транзакция входит в блок B и исполняется успешно. Баланс пользователя в сети LEFT уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
В сети RIGHT в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока B, есть транзакция от аккаунта 0x130930e3E3D30bF8F975a729e948CdCc21
При опросе баланса пользователя 0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca в сети RIGHT на момент завершения блока, в который включена транзакция от оракула, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 1000 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу 0x5580ba66f8d6dc71adb0ca1d1c6b3
отправляет 10 нативных токенов на адрес 0x159AE4d7012B18A0dE4e390714a6efa036487f0b.
Транзакция входит в блок ‘B*‘ и исполняется успешно. Баланс пользователя в сети RIGHT уменьшается на 10 токенов. Баланс моста увеличивается на 10 токенов.
В сети LEFT в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока ‘B*‘, есть транзакция от аккаунта 0x130930e3E3D30bF8F975a729e948CdC
При опросе баланса пользователя 0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca в сети LEFT на момент завершения блока, в который включена транзакция от оракула, видно, что на 10 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 10 нативных токенов.
3. Администратор моста отправляет в сеть LEFT транзакцию, вызывающую метод removeValidator с параметром

```
("0x130930e3E3D30bF8F975a729e948CdCc212ECFBB")
```

в контракте управления набором валидаторов. Транзакция исполняется успешно.

4. Пользователь, чей приватный ключ соответствует адресу `0x666a479f910d0ca5418b00316b93c` отправляет 27 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция входит в блок ‘B**‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 27 токенов. Баланс моста увеличивается на 27 токенов.
   В сети `LEFT` в последующих выпущенных блоках нет транзакции от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB`. Изменения балансов пользователя `0x666a479f910d0ca5418b00316b93cf5af6b6baa1` и контрактов моста не происходит.
5. Пользователь, чей приватный ключ соответствует адресу `0x97bdb4071396b7f60b65e0eb62ce2` отправляет 42 нативных токена на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок ‘B***‘ и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 42 токена. Баланс моста увеличивается на 42 токена.
   В сети `RIGHT` в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока ‘B****‘, есть транзакция от аккаунта `0x130930e3E3D30bF8F975a729e948` При опросе баланса пользователя `0x97bdb4071396b7f60b65e0eb62ce212a699f4b08` в сети `RIGHT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 42 нативных токена стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 42 нативных токена.
6. Администратор моста отправляет в сеть `RIGHT` транзакцию, вызывающую метод `removeValidator` с параметром

   ```
   ("0x130930e3E3D30bF8F975a729e948CdCc212ECFBB")
   ```

   в контракте управления набором валидаторов. Транзакция исполняется успешно.
7. Пользователь, чей приватный ключ соответствует адресу `0xb54edc6dd623e5a50e8aa4e52ff90` отправляет 15 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок ‘B***‘ и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 15 токенов. Баланс моста увеличивается на 15 токенов.
   В сети `RIGHT` в последующих выпущенных блоках нет транзакции от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB`. Изменения балансов пользователя `0xb54edc6dd623e5a50e8aa4e52ff906b63c9bd9cb` и контрактов моста не происходит.
8. Администратор моста отправляет в сеть `LEFT` транзакцию, вызывающую метод `addValidator` с параметром

   ```
   ("0x130930e3E3D30bF8F975a729e948CdCc212ECFBB")
   ```

   в контракте управления набором валидаторов. Транзакция исполняется успешно.
9. В сети `LEFT` в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока с тразнакцией исполняющей метод `addValidator`, есть транзакция от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB`. При опросе баланса пользователя `0x666a479f910d0ca5418b00316b93cf5af6b6baa1` в сети `LEFT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 27 нативных токенов стало больше. При аналогичном

запросе для баланса моста видно его уменьшение на 27 нативных токенов.

10. Администратор моста отправляет в сеть `RIGHT` транзакцию, вызывающую метод `addValidator` с параметром

    `("0x130930e3E3D30bF8F975a729e948CdCc212ECFBB")`

    в контракте управления набором валидаторов. Транзакция исполняется успешно.
11. В сети `RIGHT` в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока с тразнакцией исполняющей метод `addValidator`, есть транзакция от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB`. При опросе баланса пользователя `0xb54edc6dd623e5a50e8aa4e52ff906b63c9bd9cb` в сети `RIGHT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 15 нативных токена стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 15 нативных токена.

US-016 Оракул: Перезапуск оракула

Описание: Я, как администратор системы, где запущен оракул, могу быть уверен, что оракул автоматически продолжит работать после перезапуска или переустановки системы, при этом он не будет тратить средства на посылку подвтерждений, которые были уже отправлены до перезапуска/переустановки системы.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`
2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
VALIDATORS=0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=1
```

3. Установлена ликвидность моста в 2000 токенов.
4. При содежимом файла `.env`:

```
PRIVKEY=<private key corresponding to
↪  0x130930e3E3D30bF8F975a729e948CdCc212ECFBB>
LEFT_RPCURL=https://sokol.poa.network
LEFT_ADDRESS=0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441
LEFT_START_BLOCK=17290597
LEFT_GASPRICE=5000000000
```

```
RIGHT_RPCURL=https://data-seed-prebsc-1-s1.binance.org:8545/
RIGHT_ADDRESS=0x159AE4d7012B18A0dE4e390714a6efa036487f0b
RIGHT_START_BLOCK=5847797
RIGHT_GASPRICE=10000000000
ORACLE_DATA=/some/path
```

и пустой директории `/some/path` запускается команда:

```
$ docker-compose up -d
```

При использовании `docker ps -a` видно, что, как минимум, один контейнеры основывается на образе `n2n-oracle`.

Критерий оценивания (AC-016-01) : Сохранение состояния оракулом

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3`
отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`.
Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
В сети `RIGHT` в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока `B`, есть транзакция от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc21`
При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `RIGHT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 1000 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3`
отправляет 10 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`.
Транзакция входит в блок ‘B*‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 10 токенов. Баланс моста увеличивается на 10 токенов.
В сети `LEFT` в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока ‘B*‘, есть транзакция от аккаунта `0x130930e3E3D30bF8F975a729e948CdC`
При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `LEFT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 10 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 10 нативных токенов.
3. Оракул останавливается с помощью команды, запущенной в директории с исходным кодом оракула

```
$ docker-compose down
```

Через 10 секунд (максимум) соответствующих docker контейнеров нет в системе.
4. Пользователь, чей приватный ключ соответствует адресу `0x666a479f910d0ca5418b00316b93c`
отправляет 27 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`.
Транзакция входит в блок ‘B**‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 27 токенов. Баланс моста увеличивается на 27 токенов.
В сети `LEFT` в последующих выпущенных блоках нет транзакции от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB`. Изменения балансов пользо-

вателя 0x666a479f910d0ca5418b00316b93cf5af6b6baa1 и контрактов моста не происходит.

5. Пользователь, чей приватный ключ соответствует адресу 0xb54edc6dd623e5a50e8aa4e52ff90
отправляет 15 нативных токенов на адрес 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441.
Транзакция входит в блок ‘B***‘ и исполняется успешно. Баланс пользователя в сети LEFT уменьшается на 15 токенов. Баланс моста увеличивается на 15 токенов.
В сети RIGHT в последующих выпущенных блоках нет транзакции от аккаунта 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB. Изменения балансов пользователя 0xb54edc6dd623e5a50e8aa4e52ff906b63c9bd9cb и контрактов моста не происходит.
6. Оракул запускается через

```
$ docker-compose up -d
```

При использовании docker ps -a видно, что, как минимум, один контейнеры основывается на образе n2n-oracle.

7. В сети LEFT в одном из блоков, выпущенном не позднее чем через 30 секунд после запуска оракула, есть только одна транзакция от аккаунта 0x130930e3E3D30bF8F975a729e
При опросе баланса пользователя 0x666a479f910d0ca5418b00316b93cf5af6b6baa1
в сети LEFT на момент завершения блока, в который включена транзакция от оракула, видно, что на 27 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 27 нативных токенов.
8. В сети RIGHT в одном из блоков, выпущенном не позднее чем через 30 секунд послезапуска оракула, есть только одна транзакция от аккаунта 0x130930e3E3D30bF8F975a729e
При опросе баланса пользователя 0xb54edc6dd623e5a50e8aa4e52ff906b63c9bd9cb
в сети RIGHT на момент завершения блока, в который включена транзакция от оракула, видно, что на 15 нативных токена стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 15 нативных токена.

Критерий оценивания (AC-016-02) : Восстановления состояния оракулом после очистки системы

---

1. Пользователь, чей приватный ключ соответствует адресу 0x5580ba66f8d6dc71adb0ca1d1c6b3
отправляет 1000 нативных токенов на адрес 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441.
Транзакция входит в блок B и исполняется успешно. Баланс пользователя в сети LEFT уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
В сети RIGHT в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока B, есть транзакция от аккаунта 0x130930e3E3D30bF8F975a729e948CdCc21
При опросе баланса пользователя 0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca
в сети RIGHT на момент завершения блока, в который включена транзакция от оракула, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 1000 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу 0x5580ba66f8d6dc71adb0ca1d1c6b3
отправляет 10 нативных токенов на адрес 0x159AE4d7012B18A0dE4e390714a6efa036487f0b.
Транзакция входит в блок ‘B*‘ и исполняется успешно. Баланс пользователя в сети RIGHT уменьшается на 10 токенов. Баланс моста увеличивается на 10 токенов.

В сети LEFT в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока ‘B*‘, есть транзакция от аккаунта 0x130930e3E3D30bF8F975a729e948CdC
При опросе баланса пользователя 0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca в сети LEFT на момент завершения блока, в который включена транзакция от оракула, видно, что на 10 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 10 нативных токенов.

3. Оракул останавливается с помощью команды, запущенной в директории с исходным кодом оракула

```
$ docker-compose down
```

Через 10 секунд (максимум) соответствующих docker контейнеров нет в системе.

4. Пользователь, чей приватный ключ соответствует адресу 0x666a479f910d0ca5418b00316b93c отправляет 27 нативных токенов на адрес 0x159AE4d7012B18A0dE4e390714a6efa036487f0b.
Транзакция входит в блок ‘B**‘ и исполняется успешно. Баланс пользователя в сети RIGHT уменьшается на 27 токенов. Баланс моста увеличивается на 27 токенов.
В сети LEFT в последующих выпущенных блоках нет транзакции от аккаунта 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB. Изменения балансов пользователя 0x666a479f910d0ca5418b00316b93cf5af6b6baa1 и контрактов моста не происходит.
5. Пользователь, чей приватный ключ соответствует адресу 0xb54edc6dd623e5a50e8aa4e52ff90 отправляет 15 нативных токенов на адрес 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441.
Транзакция входит в блок ‘B***‘ и исполняется успешно. Баланс пользователя в сети LEFT уменьшается на 15 токенов. Баланс моста увеличивается на 15 токенов.
В сети RIGHT в последующих выпущенных блоках нет транзакции от аккаунта 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB. Изменения балансов пользователя 0xb54edc6dd623e5a50e8aa4e52ff906b63c9bd9cb и контрактов моста не происходит.
6. Удаляется все содержимое директории /some/path. После чего запускается оракул через

```
$ docker-compose up -d
```

При использовании docker ps -a видно, что, как минимум, один контейнеры основывается на образе n2n-oracle.

7. В сети LEFT в одном из блоков, выпущенном не позднее чем через 30 секунд после запуска оракула, есть только одна транзакция от аккаунта 0x130930e3E3D30bF8F975a729e
При опросе баланса пользователя 0x666a479f910d0ca5418b00316b93cf5af6b6baa1 в сети LEFT на момент завершения блока, в который включена транзакция от оракула, видно, что на 27 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 27 нативных токенов.
8. В сети RIGHT в одном из блоков, выпущенном не позднее чем через 30 секунд послезапуска оракула, есть только одна транзакция от аккаунта 0x130930e3E3D30bF8F975a729e
При опросе баланса пользователя 0xb54edc6dd623e5a50e8aa4e52ff906b63c9bd9cb в сети RIGHT на момент завершения блока, в который включена транзакция от оракула, видно, что на 15 нативных токена стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 15 нативных токена.

Критерий оценивания (AC-016-03) : Восстановления состояния оракулом после смены валидатора

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3`
   отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`.
   Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
   В сети `RIGHT` в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока `B`, есть транзакция от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc21`
   При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca`
   в сети `RIGHT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 1000 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3`
   отправляет 10 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`.
   Транзакция входит в блок ‘B*‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 10 токенов. Баланс моста увеличивается на 10 токенов.
   В сети `LEFT` в одном из блоков, выпущенном не позднее чем через 15 секунд после появления блока ‘B*‘, есть транзакция от аккаунта `0x130930e3E3D30bF8F975a729e948CdC`
   При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca`
   в сети `LEFT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 10 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 10 нативных токенов.
3. Оракул останавливается с помощью команды, запущенной в директории с исходным кодом оракула

```
$ docker-compose down
```

   Через 10 секунд (максимум) соответствующих docker контейнеров нет в системе.
4. Пользователь, чей приватный ключ соответствует адресу `0x666a479f910d0ca5418b00316b93c`
   отправляет 27 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`.
   Транзакция входит в блок ‘B**‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 27 токенов. Баланс моста увеличивается на 27 токенов.
   В сети `LEFT` в последующих выпущенных блоках нет транзакции от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB`. Изменения балансов пользователя `0x666a479f910d0ca5418b00316b93cf5af6b6baa1` и контрактов моста не происходит.
5. Пользователь, чей приватный ключ соответствует адресу `0xb54edc6dd623e5a50e8aa4e52ff90`
   отправляет 15 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`.
   Транзакция входит в блок ‘B***‘ и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 15 токенов. Баланс моста увеличивается на 15 токенов.
   В сети `RIGHT` в последующих выпущенных блоках нет транзакции от аккаунта `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB`. Изменения балансов пользователя `0xb54edc6dd623e5a50e8aa4e52ff906b63c9bd9cb` и контрактов моста не происходит.

6. Администратор моста отправляет в обе сети по транзакции, вызывающих метод `addValidator` с параметром

   `("0x61365C58E44A6Fc166897f4A30641dba82E606c0")`

   в контрактах управления набором валидаторов. Транзакции исполняется успешно.

7. Администратор моста отправляет в обе сети по транзакции, вызывающих метод `removeValidator` с параметром

   `("0x130930e3E3D30bF8F975a729e948CdCc212ECFBB")`

   в контрактах управления набором валидаторов. Транзакции исполняется успешно.

8. Удаляется все содержимое директории `/some/path`. Затем, содержимое файла `.env` изменяется следующим образом:

```
PRIVKEY=<private key corresponding to
↪  0x61365C58E44A6Fc166897f4A30641dba82E606c0>
LEFT_RPCURL=https://sokol.poa.network
LEFT_ADDRESS=0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441
LEFT_START_BLOCK=17290597
LEFT_GASPRICE=5000000000
RIGHT_RPCURL=https://data-seed-prebsc-1-s1.binance.org:8545/
RIGHT_ADDRESS=0x159AE4d7012B18A0dE4e390714a6efa036487f0b
RIGHT_START_BLOCK=5847797
RIGHT_GASPRICE=10000000000
ORACLE_DATA=/some/path
```

   После чего запускается оракул через

```
$ docker-compose up -d
```

   При использовании `docker ps -a` видно, что, как минимум, один контейнеры основывается на образе `n2n-oracle`.

9. В сети `LEFT` в одном из блоков, выпущенном не позднее чем через 30 секунд после запуска оракула, есть только одна транзакция от аккаунта `0x61365C58E44A6Fc166897f4A3` При опросе баланса пользователя `0x666a479f910d0ca5418b00316b93cf5af6b6baa1` в сети `LEFT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 27 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 27 нативных токенов.

10. В сети `RIGHT` в одном из блоков, выпущенном не позднее чем через 30 секунд послезапуска оракула, есть только одна транзакция от аккаунта `0x61365C58E44A6Fc166897f4A3` При опросе баланса пользователя `0xb54edc6dd623e5a50e8aa4e52ff906b63c9bd9cb` в сети `RIGHT` на момент завершения блока, в который включена транзакция от оракула, видно, что на 15 нативных токена стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 15 нативных токена.

US-017 Контракт моста: надежная доставка токенов

Описание: Я, как администратор моста, хочу быть уверен, что после получения достаточного количества подтверждений от валидаторов моста, нативные токены могут быт переданы на адрес контракта даже, если он явно это запрещает.

Критерий оценивания (AC-017-01) : Пересылка на адрес контракта

1. Контракт моста зарегистрирован в блокчейн сети с правой стороны по адресу `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e` при следующем содержимом переменных окружения:

```
VALIDATORS=0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
THRESHOLD=1
```

2. В блокчейн сети с правой стороны зарегистрирован контракт по адресу `0xf712a82DD8e2Ac923`

```
pragma solidity 0.7.5;

contract familyWallet {
    event Received(address sender, uint256 value);
    address owner1;
    address owner2;

    constructor (address husband, address wife) {
        require(husband != wife, "the same");
        require(wife != address(0), "is zero");

        owner1 = husband;
        owner2 = wife;
    }

    function sendFunds(address payable receiver, uint256 value) external {
        require(msg.sender == owner1 || msg.sender == owner2, "Not allowed");
        require(value <= address(this).balance, "not enough");
        receiver.transfer(value);
    }

    receive () payable external {
        emit Received(msg.sender, msg.value);
    }
}
```

3. В блокчейн сети с правой стороны зарегистрирован контракт по адресу `0xC1d9971bcd39bd96f`

```
    pragma solidity 0.7.5;

    contract familyWallet {
        event Received(address sender, uint256 value);
        address owner1;
        address owner2;

        constructor (address husband, address wife) {
            require(husband != wife, "the same");
            require(wife != address(0), "is zero");

            owner1 = husband;
            owner2 = wife;
        }

        function sendFunds(address payable receiver, uint256 value) external {
            require(msg.sender == owner1 || msg.sender == owner2, "Not
            ↪  allowed");
            require(value <= address(this).balance, "not enough");
            receiver.transfer(value);
        }
```

```
21
22          function receiveFunds() payable external {
23              emit Received(msg.sender, msg.value);
24          }
25
26          receive () payable external {
27              revert("Not supported");
28          }
29      }
```

4. Администратор контракта моста отправляет транзакцию на контракт `0x33E0E07cA86c869adE` вызывающую метод `addLiquidity`. Данная транзакция также пересылает 100 нативных токенов. Транзакция исполняется успешно.
5. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 5000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. Баланс контракта `0xf712a82DD8e2Ac923299193e9d6dAEda` после исполнения транзакции становится 50 нативных токенов. Баланс контракта моста меняется до 50 нативных токенов.
6. Аккаунт с адресом `0x130930e3E3D30bF8F975a729e948CdCc212ECFBB` отправляет транзакцию на контракт `0x33E0E07cA86c869adE3fc9DE9126f6C73DAD105e`, вызывающую метод `commit` с параметром

   ```
   ("0xC1d9971bcd39bd96f64787765221be86d1a928e5", 5000000000000000000,
   ↪  "0xbe4da129fc927472b7b5ab00c04a0c19124fd3915a51650e672f332dcc8d2f3d")
   ```

   Транзакция исполняется успешно. Баланс контракта `0xC1d9971bcd39bd96f64787765221be86` после исполнения транзакции становится 5 нативных токенов. Баланс контракта моста меняется до 45 нативных токенов.

US-018 Контракт моста: экономичная посылка подтверждений в левую сторону

Описание: Я, как валидатор моста, хочу, чтобы мост потреблял меньше средств на комиссии транзакций. Если рассмотрим ситуацию, что в левой части моста сеть с большими комиссиями, то ПО моста должно минимизировать количество транзакций в левую часть, сохраняя тот же уровень безопасности и децентрализации.

В данной US подразумевается, что ABI контрактов мостов изменяется. Добавляются следуюшие методы:

- Контракт моста с правой стороны:
  - `enableRobustMode()`

    позволяет администратору включить режим экономичной пересылки. Обратной силы настройка не имеет.
  - `registerCommit(address recipient, uint256 amount, bytes32 id, uint256 r, uint256 s, uint8 v)`

    позволяет валидатору моста отправить в виде цифровой подписи подтвреждение о наличии конкретного запроса на отправку нативных токенов через мост из правой стороны. В тот момент, когда достаточно подтверждений получено, испускает событие

`commitsCollected(bytes32 id, uint8 commits)`

где `commits` – количество собранных подтверждений. Метод работает только если был заранее вызван `enableRobustMode()`.

- `getTransferDetails(bytes32 id)`

  возвращает
  `(address recipient, uint256 amount)`

  для конкретного запроса на отправку нативных токенов через мост из правой стороны.
- `getCommit(bytes32 id, uint8 index)`

  возвращает
  `(uint256 r, uint256 s, uint8 v)`

  – подпись одного из валидаторов для конкретного запроса на отправку нативных токенов через мост из правой стороны.
- `getRobustModeMessage(address recipient, uint256 amount, bytes32 id) ↪ returns(bytes)`

  возвращает набор данных, который бы использовался, как вход для `encode_defunct()` функции из `eth_account.messages` для последующего формирования цифровой подписи, передаваемой в `registerCommit`.

- Контракт моста с левой стороны:
  - `enableRobustMode()`

    позволяет администратору включить режим экономичной пересылки. Обратной силы настройка не имеет. После включения данного режима метод `commit` в контракте моста с левой стороны всегда возвращает `revert`.
  - `applyCommits(address recipient, uint256 amount, bytes32 id, uint256[] r, ↪ uint256[] s, uint8[] v)`

    позволяет любому пользователю блокчейн сети переслать в контракт моста с левой стороны собранные подтверждения о наличии конкретного запроса на отправку нативных токенов через мост из правой стороны. При успешной проверке подверждений контракт пересылает получателю соответствующее количество нативных токенов.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`
2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
VALIDATORS=0x17F26d6DEcA57CC5E4cF063a39d3c15AA07845cE ↪ 0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc ↪ 0x2c89fB4Ff72E7cf512E3BBfb571B8c1FdFaA4b5C
```

```
THRESHOLD=2
```

3. Установлена ликвидность моста в 2000 токенов.

Критерий оценивания (AC-018-01) : Пересылка токенов

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов. Через некоторое время в сети `RIGHT` видно, что баланс пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` увеличился на 1000 нативных токенов, а баланс моста уменьшился на эту же сумму.
2. Администратор моста отправляет в контаркт моста с правой стороны транзакцию, вызывающую метод `enableRobustMode`. Транзакция исполняется успешно.
3. Администратор моста отправляет в контаркт моста с левой стороны транзакцию, вызывающую метод `enableRobustMode`. Транзакция исполняется успешно.
4. Вызов метода `getRobustModeMessage` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804")
   ```

   возвращает такой набор данных, который будучи скомпонованный согласно EIP-191 и последующей генерации цифровой подписи с применением приватного ключа соответствующему адресу `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` выдает следующие значени `r,s,v`:

   ```
   (830706558590161589730662201458725670327443718495145312185968671274853922254 61,
   ↪  376526935438515435737171105557755011801940408129953736699465114599900585 69296,
   ↪  28)
   ```

5. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` отправляет транзакцию на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающую метод `registerCommit` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  830706558590161589730662201458725670327443718495145312185968671274853922254 61,
   ↪  376526935438515435737171105557755011801940408129953736699465114599900585 69296,
   ↪  28)
   ```

   Транзакция исполняется успешно. Выписка транзакции не содержит событие `commitsCollected`.

6. Аккаунт с адресом `0x2c89fB4Ff72E7cf512E3BBfb571B8c1FdFaA4b5C` отправляет транзакцию на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающую метод `registerCommit` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  481358271907542299630530434102432078767857409691846996950103811148970852212 21,
   ↪  522937085518996764545042919577545491740525425848351427065116780112188592346 19,
   ↪  28)
   ```

   Транзакция исполняется успешно. В выписке транзакции есть событие:

```
    ↪ commitsCollected("0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
    ↪ 2)
```

7. Аккаунт с адресом `0x17F26d6DEcA57CC5E4cF063a39d3c15AA07845cE` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающей метод `registerCommit` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪ "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪ 67740533537290287998916608516618072883933156133855606968773029368135942146398,
   ↪ 21337456040719236310399050803970139750915503576430763461362832593000564881009,
   ↪ 28)
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

8. Вызов метода `getTransferDetails` с параметром

   ```
   ("0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804")
   ```

   возвращает два значения: `0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34` и `10000000000000`

9. Вызов метода `getCommit` с параметрами

   ```
   ("0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804", 0)
   ```

   возвращает три значения:
   `830706558590161589730662201458725670327443718495145312185968671274853922254 61`,
   `376526935438515435737171105557755011801940408129953736699465114599900585692 96`
   и
   28.

10. Вызов метода `getCommit` с параметрами

    ```
    ("0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804", 1)
    ```

    возвращает три значения:
    `481358271907542299630530434102432078767857409691846996950103811148970852212 21`,
    `522937085518996764545042919577545491740525425848351427065116780112188592346 19`
    и
    28.

11. Аккаунт с адресом `0x4910988ab9ece5fe8ffe8260b5557e9572cddab5` отправляет транзакцию на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающую метод `applyCommits` с параметрами

    ```
    ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
    ↪ "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
    ↪ [830706558590161589730662201458725670327443718495145312185968671274853922254 61,
    ↪ 481358271907542299630530434102432078767857409691846996950103811148970852212 21],
    ↪ [376526935438515435737171105557755011801940408129953736699465114599900585692 96,
    ↪ 522937085518996764545042919577545491740525425848351427065116780112188592346 19],
    ↪ [28, 28])
    ```

    Транзакция исполняется успешно. После исполнения транзакции баланс пользователя `0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34` увеличился на 10 нативных токенов, а баланс моста уменьшается на эту же сумму.

Критерий оценивания (AC-018-02) : Некорректная посылка подтверждений в контракт моста на правой стороне

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов. Через некоторое время в сети `RIGHT` видно, что баланс пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` увеличился на 1000 нативных токенов, а баланс моста уменьшился на эту же сумму.
2. Администратор моста отправляет в контаркт моста с правой стороны транзакцию, вызывающую метод `enableRobustMode`. Транзакция исполняется успешно.
3. Аккаунт с адресом `0x4910988ab9ece5fe8ffe8260b5557e9572cddab5` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающей метод `registerCommit` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  67740533537290287998916608516618072883933156133855606968773029368135942146398,
   ↪  21337456040719236310399050803970139750915503576430763461362832593000564881009,
   ↪  28)
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
4. Аккаунт с адресом `0x17F26d6DEcA57CC5E4cF063a39d3c15AA07845cE` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающей метод `registerCommit` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  83070655859016158973066220145872567032744371849514531218596867127485392225461,
   ↪  37652693543851543573717110555775501180194040812995373669946511459990058569296,
   ↪  28)
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
5. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающей метод `registerCommit` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 20000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  83070655859016158973066220145872567032744371849514531218596867127485392225461,
   ↪  37652693543851543573717110555775501180194040812995373669946511459990058569296,
   ↪  28)
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
6. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` отправляет транзакцию на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающую метод `registerCommit` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  83070655859016158973066220145872567032744371849514531218596867127485392225461,
   ↪  37652693543851543573717110555775501180194040812995373669946511459990058569296,
   ↪  28)
   ```

Транзакция исполняется успешно. Выписка транзакции не содержит событие `commitsCollected`.

7. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающей метод `registerCommit` с параметрами

```
("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
↪  830706558590161589730662201458725670327443718495145312185968671274853922254 61,
↪  376526935438515435737171105557755011801940408129953736699465114599900585692 96,
↪  28)
```

Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

8. Аккаунт с адресом `0x2c89fB4Ff72E7cf512E3BBfb571B8c1FdFaA4b5C` отправляет транзакцию на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающую метод `registerCommit` с параметрами

```
("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
↪  481358271907542299630530434102432078767857409691846996950103811148970852212 21,
↪  522937085518996764545042919577545491740525425848351427065116780112188592346 19,
↪  28)
```

Транзакция исполняется успешно. В выписке транзакции есть событие:

```
↪  commitsCollected("0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
↪  2)
```

9. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающей метод `registerCommit` с параметрами

```
("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
↪  830706558590161589730662201458725670327443718495145312185968671274853922254 61,
↪  376526935438515435737171105557755011801940408129953736699465114599900585692 96,
↪  28)
```

Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

Критерий оценивания (AC-018-03) : Некорректная посылка подтверждений в контракт моста на левой стороне

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов. Через некоторое время в сети `RIGHT` видно, что баланс пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` увеличился на 1000 нативных токенов, а баланс моста уменьшился на эту же сумму.

2. Администратор моста отправляет в контаркт моста с левой стороны транзакцию, вызывающую метод `enableRobustMode`. Транзакция исполняется успешно.
3. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающей метод `commit` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804")
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
4. Аккаунт с адресом `0x4910988ab9ece5fe8ffe8260b5557e9572cddab5` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающей метод `applyCommits` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  [830706558590161589730662201458725670327443718495145312185968671274853922254 61],
   ↪  [376526935438515435737171105557755011801940408129953736699465114599900585692 96],
   ↪  [28])
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
5. Аккаунт с адресом `0x4910988ab9ece5fe8ffe8260b5557e9572cddab5` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающей метод `applyCommits` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 20000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  [830706558590161589730662201458725670327443718495145312185968671274853922254 61,
   ↪  481358271907542299630530434102432078767857409691846996950103811148970852212 21],
   ↪  [376526935438515435737171105557755011801940408129953736699465114599900585692 96,
   ↪  522937085518996764545042919577545491740525425848351427065116780112188592346 19],
   ↪  [28, 28])
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
6. Аккаунт с адресом `0x4910988ab9ece5fe8ffe8260b5557e9572cddab5` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающей метод `applyCommits` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  [481358271907542299630530434102432078767857409691846996950103811148970852212 21,
   ↪  481358271907542299630530434102432078767857409691846996950103811148970852212 21],
   ↪  [522937085518996764545042919577545491740525425848351427065116780112188592346 19,
   ↪  522937085518996764545042919577545491740525425848351427065116780112188592346 19],
   ↪  [28, 28])
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
7. Аккаунт с адресом `0x4910988ab9ece5fe8ffe8260b5557e9572cddab5` отправляет транзакцию на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающую метод `applyCommits` с параметрами

```
("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
↪  [830706558590161589730662201458725670327443718495145312185968671274853922254 61,
↪  481358271907542299630530434102432078767857409691846996950103811148970852212 21],
↪  [376526935438515435737171105557755011801940408129953736699465114599900585692 96,
↪  522937085518996764545042919577545491740525425848351427065116780112188592346 19],
↪  [28, 28])
```

Транзакция исполняется успешно. После исполнения транзакции баланс пользователя `0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34` увеличился на 10 нативных токенов, а баланс моста уменьшается на эту же сумму.

8. Аккаунт с адресом `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающей метод `applyCommits` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  [830706558590161589730662201458725670327443718495145312185968671274853922254 61,
   ↪  481358271907542299630530434102432078767857409691846996950103811148970852212 21],
   ↪  [376526935438515435737171105557755011801940408129953736699465114599900585692 96,
   ↪  522937085518996764545042919577545491740525425848351427065116780112188592346 19],
   ↪  [28, 28])
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

9. Аккаунт с адресом `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающей метод `applyCommits` с параметрами

   ```
   ("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34", 10000000000000000000,
   ↪  "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪  [830706558590161589730662201458725670327443718495145312185968671274853922254 61,
   ↪  677405335372902879989166085166180728839331561338556069687730293681359421463 98],
   ↪  [376526935438515435737171105557755011801940408129953736699465114599900585692 96,
   ↪  213374560407192363103990508039701397509155035764307634613628325930005648810 09],
   ↪  [28, 28])
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

US-019 Оракул: экономичная посылка подтверждений в левую сторону

Описание: Я, как пользователь моста, могу ожидать, что после переключения в режим экономичной пересылки оракулы продолжат переправлять подтверждения об пересылке токенов через мост. При этом количество транзакций от оракулов в левую сеть значительно уменьшится.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`

  - время выпуска блоков в среднем раз в 5 секунд
  - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
- Сеть `RIGHT`:
  - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
  - время выпуска блоков в среднем раз в 5 секунд
  - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`

2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

3. Установлена ликвидность моста в 2000 токенов.
4. На двух сторонах моста выполнена команда `enableRobustMode`.
5. Запущены три оракула. В `PRIVKEY` `.env`-файла каждого оракула указан свой собственный приватный ключ, который соответствует одному из адресов, перечисленных в `VALIDATORS` при регистрации контрактов моста. Порядок запуска оракулов не определен. Промежутки времени между запусками оракулов не определены.

Критерий оценивания (AC-019-01) : Пересылка токенов

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3`, отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
   В сети `RIGHT` в одном или нескольких блоках, выпущенном не позднее чем через 15 секунд после появления блока `B`, есть транзакции от двух или трех оракулов. При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `RIGHT` на момент завершения блока, в который включена последняя успешная транзакция от оракулов, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 0 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3`, отправляет `X` ($0.001 < X < 50$) нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6`. Транзакция `0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804` входит в блок 'B*' и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на `X` токенов. Баланс моста увеличивается на `X` токенов.
3. В сети `RIGHT` в одном из блоков 'B**', выпущенном не позднее через 15 секунд после того, как выпущен блок 'B*', есть транзакции от оракулов. В выписке одной из транзакции есть событие:

```
↪ commitsCollected("0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
↪ 2)
```

4. В сети `LEFT` в блоке, выпущенном не позднее чем через 15 секунд после появления блока ‘B**‘, есть транзакция от одного из оракулов и нет транзакций от других. При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `LEFT` на момент завершения блока, в который включена транзакция оракула, видно, что на `X` нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на `X` нативных токенов.
5. Пользователь, чей приватный ключ соответствует адресу `0x97bdb4071396b7f60b65e0eb62ce2` отправляет `Y` ($0.001 < Y < 50$) нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103` Транзакция входит в блок ‘B***‘ и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на `Y` токенов. Баланс моста увеличивается на `Y` токенов.
   В сети `RIGHT` в одном или нескольких блоках, выпущенном не позднее чем через 15 секунд после появления блока ‘B***‘, есть транзакции от двух или трех оракулов. При опросе баланса пользователя `0x97bdb4071396b7f60b65e0eb62ce212a699f4b08` в сети `RIGHT` на момент завершения блока, в который включена последняя успешная транзакция от оракулов, видно, что на `Y` нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на `Y` нативных токенов.

US-020 CLI: ручная пересылка пользователем собранных подтверждений в левую сторону

Описание: Я, как пользователь моста, при контрактах моста переключенных в режим экономичной пересылки могу с помощью специальной CLI комагды самостоятельно переслать в левую часть моста подтверждения от валидаторов, собранные в правой стороне. Такая возможность необходима на случай, когда валидаторы не имеют средств на оплату комиссий для транзакций блокчейн сети на левой стороне моста.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`
2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
VALIDATORS=0x226Dbd003697a5Ab7A501ED6fc02Bda096C66E70
↪ 0x130930e3E3D30bF8F975a729e948CdCc212ECFBB
↪ 0x61365C58E44A6Fc166897f4A30641dba82E606c0
THRESHOLD=2
```

3. Установлена ликвидность моста в 1000 токенов.
4. На двух сторонах моста выполнена команда `enableRobustMode`.
5. Запущены три оракула. В `PRIVKEY` `.env`-файла каждого оракула указан свой собственный приватный ключ, который соответствует одному из адресов, перечисленных в `VALIDATORS` при регистрации контрактов моста. Порядок запуска оракулов не определен. Промежутки времени между запусками оракулов не определены.
6. У всех валидаторов моста отсутствует средства на балансах в блокчейн сети `LEFT`.

Критерий оценивания (AC-020-01) : Пересылка подтверждения на левую сторону

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
   В сети `RIGHT` в одном или нескольких блоках, выпущенном не позднее чем через 15 секунд после появления блока `B`, есть транзакции от двух или трех оракулов. При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `RIGHT` на момент завершения блока, в который включена последняя успешная транзакция от оракулов, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 0 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу `0xF91296B9d7699d800c2Ba0a80e755` отправляет 10 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция `0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804` входит в блок ‘B*‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 10 токенов. Баланс моста увеличивается на 10 токенов.
   В сети `LEFT` ни в одном из блоков, выпущенном после появления блока ‘B*‘, нет транзакций от оракулов. Баланс пользователя `0xF91296B9d7699d800c2Ba0a80e755972a9DA7C` в сети `LEFT` не изменился.
3. В директории `/some/path` находится `.env` файл со следующим содержимым:

```
PRIVKEY=<private key corresponding to
↪  0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34>
LEFT_RPCURL=https://sokol.poa.network
LEFT_ADDRESS=0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441
LEFT_GASPRICE=5000000000
RIGHT_RPCURL=https://data-seed-prebsc-1-s1.binance.org:8545/
RIGHT_ADDRESS=0x159AE4d7012B18A0dE4e390714a6efa036487f0b
```

4. После запуска команды:

```
$ docker run -ti --rm --env-file /some/path/.env n2n-oracle \
    /tools/applyCommits.py \
    0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804
```

   в терминале выводится

```
0x930650054b50608dafb24c08b71f46bf8605de83d9ad5502cad0c6159dd074bf executed
```

При опросе баланса пользователя `0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34` в сети `LEFT` на момент завершения блока, в который включена транзакция `0x930650054b50608dafb24c08b71f46bf8605de83d9ad5502cad0c6159dd074bf`, видно, что на 10 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 10 нативных токенов.

US-021 Удалена из списка задач

US-022 Контракт моста: лимиты на пересылку

Описание: Я, как администратор моста, хочу определять ограничения на минимальное и максимальное количество токенов, которые могут быть переданы в одной операции. Минимальное количество за транзакцию обуславливается экономической целесообразностью операций (переданное значение больше комиссий, которые валидаторы заплатят за подтверждения). Максимальное количество за транзакцию обуславливается безопасностью (злоумышленник не сможет лишить мост ликвидности несколькими транзакциями до того, как администратор сможет заняться решением проблемы).

В данной US подразумевается, что ABI контрактов мостов изменяется. Добавляются следуюшие методы в контракты моста:

- `setMinPerTx(uint256 _min)`

  устанавливает ограничение на минимальное количество нативных токенов, которые могут пересланы на контракт моста пользователем. Если метод ни разу не вызван, то ограничение – 0 токенов.
- `setMaxPerTx(uint256 _max)`

  устанавливает ограничение на максимальное количество токенов, которые могут пересланы на контракт моста пользователем или которые могут быть подверждены валидатором. Если метод ни разу не вызван, то ограничения нет.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`

2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
VALIDATORS=0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc
THRESHOLD=1
```

3. Установлена ликвидность моста в 2000 токенов.

Критерий оценивания (AC-022-01) : Ограничение на минимальное количество

---

1. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 50 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция исполняется успешно. Баланс пользователя уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов.
2. Администратор моста отправляет в контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c44` транзакцию, вызывающую метод `setMinPerTx` с параметром

   `(49999999999999999999)`

   Транзакция исполняется успешно.
3. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 40 нативных токенов на адрес `0xfab1ed72a7236a6b34` Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
4. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 50 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция исполняется успешно. Баланс пользователя уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов.
5. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов. Через некоторое время в сети `RIGHT` видно, что баланс пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` увеличился на 1000 нативных токенов, а баланс моста уменьшился на эту же сумму.
6. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 50 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция исполняется успешно. Баланс пользователя уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов.
7. Администратор моста отправляет в контаркт `0x159AE4d7012B18A0dE4e390714a6efa036487f0` транзакцию, вызывающую метод `setMinPerTx` с параметром

   `(49999999999999999999)`

   Транзакция исполняется успешно.
8. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 40 нативных токенов на адрес `0x159AE4d7012B18A0dE` Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.

9. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 50 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция исполняется успешно. Баланс пользователя уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов.

Критерий оценивания (AC-022-02) : Ограничение на максимальное количество

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов. Через некоторое время в сети `RIGHT` видно, что баланс пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` увеличился на 1000 нативных токенов, а баланс моста уменьшился на эту же сумму.
2. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` отправляет транзакцию на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающую метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 10000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. У пользователя `0xf712a82DD8e2Ac923299193e9d6dAEda2d5` после исполнения транзакции становится на 100 нативных токенов больше. Баланс контракта моста меняется на 100 нативных токенов.
3. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 100 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция исполняется успешно. Баланс пользователя уменьшается на 100 токенов. Баланс моста увеличивается на 100 токенов.
4. Администратор моста отправляет в контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c44` транзакцию, вызывающую метод `setMaxPerTx` с параметром

   ```
   (99999999999999999999)
   ```

   Транзакция исполняется успешно.
5. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающей метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 10000000000000000000,
   ↪  "0x930650054b50608dafb24c08b71f46bf8605de83d9ad5502cad0c6159dd074bf")
   ```

   Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
6. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` отправляет транзакцию на контракт `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`, вызывающую метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 99000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. У пользователя `0xf712a82DD8e2Ac923299193e9d6dAEda2d5` после исполнения транзакции становится на 99 нативных токенов больше. Баланс контракта моста меняется на 99 нативных токенов.

7. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 100 нативных токенов на адрес `0xfab1ed72a7236a6b3`. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
8. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 99 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция исполняется успешно. Баланс пользователя уменьшается на 99 токенов. Баланс моста увеличивается на 99 токенов.
9. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` отправляет транзакцию на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающую метод `commit` с параметром

   ```
   ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 10000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. У пользователя `0xf712a82DD8e2Ac923299193e9d6dAEda2d5` после исполнения транзакции становится на 100 нативных токенов больше. Баланс контракта моста меняется на 100 нативных токенов.
10. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 100 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция исполняется успешно. Баланс пользователя уменьшается на 100 токенов. Баланс моста увеличивается на 100 токенов.
11. Администратор моста отправляет в контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0` транзакцию, вызывающую метод `setMaxPerTx` с параметром

    ```
    (99999999999999999999)
    ```

    Транзакция исполняется успешно.
12. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающей метод `commit` с параметром

    ```
    ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 10000000000000000000,
    ↪  "0x930650054b50608dafb24c08b71f46bf8605de83d9ad5502cad0c6159dd074bf")
    ```

    Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
13. Аккаунт с адресом `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc` отправляет транзакцию на контракт `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`, вызывающую метод `commit` с параметром

    ```
    ("0xf712a82DD8e2Ac923299193e9d6dAEda2d5a32fd", 99000000000000000000,
    ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
    ```

    Транзакция исполняется успешно. У пользователя `0xf712a82DD8e2Ac923299193e9d6dAEda2d5` после исполнения транзакции становится на 99 нативных токенов больше. Баланс контракта моста меняется на 99 нативных токенов.
14. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` вызывает JSON RPC метод `eth_estimateGas` для оценки потребления газа перед посылкой транзакции на пересылку 100 нативных токенов на адрес `0x159AE4d7012B18A0d`. Вызов `eth_estimateGas` возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда `revert`.
15. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 99 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`.

Транзакция исполняется успешно. Баланс пользователя уменьшается на 99 токенов. Баланс моста увеличивается на 99 токенов.

US-023 Контракт моста: отключение моста

Описание: Я, как администратор моста, хочу иметь возможность остановить все пользовательские опреации по передаче средств через мост. Например, это может быть использовано, если в контрактах обнаружена ошибка и необходимо обновление.

В данной US подразумевается, что ABI контрактов мостов изменяется. Добавляются следуюшие методы в контракты моста:

- `stopOperations()`

  переводит контракт моста в такое состояние при котором любые пользовательские транзакции на перевод средств контракту моста завершают свое исполнение операцией `revert`.
- `startOperations()`

  возвращает контракт в работоспособное состояние.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c`
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу `0x159AE4d7012B18A0dE4e390714a6efa036487`
2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
VALIDATORS=0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc
THRESHOLD=1
```

3. Установлена ликвидность моста в 2000 токенов.

Критерий оценивания (AC-023-01) : Отключение моста

---

1. Пользователь, чей приватный ключ соответствует адресу `0x79dD14623c4D33413c0c28fDAbC22` отправляет 50 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция исполняется успешно. Баланс пользователя уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов.

2. Администратор моста отправляет в контракт 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c44 транзакцию, вызывающую метод stopOperations. Транзакция исполняется успешно.
3. Пользователь, чей приватный ключ соответствует адресу 0x79dD14623c4D33413c0c28fDAbC22 вызывает JSON RPC метод eth_estimateGas для оценки потребления газа перед посылкой транзакции на пересылку 50 нативных токенов на адрес 0xfab1ed72a7236a6b34 Вызов eth_estimateGas возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда revert.
4. Аккаунт с адресом 0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc отправляет транзакцию на контракт 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441, вызывающую метод commit с параметром

   ```
   ("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e", 5000000000000000000,
   ↪  "0x1f4e6e8d2e4b34a3fd4df3b9ae91b3cf4cf129b089a2700e633cdeb3ea575c85")
   ```

   Транзакция исполняется успешно. У пользователя 0x79dD14623c4D33413c0c28fDAbC2285Fdb после исполнения транзакции становится на 5 нативных токенов больше. Баланс контракта моста меняется на 5 нативных токенов.
5. Администратор моста отправляет в контракт 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c44 транзакцию, вызывающую метод startOperations. Транзакция исполняется успешно.
6. Пользователь, чей приватный ключ соответствует адресу 0x79dD14623c4D33413c0c28fDAbC22 отправляет 50 нативных токенов на адрес 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441. Транзакция исполняется успешно. Баланс пользователя уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов.
7. Аккаунт с адресом 0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc отправляет транзакцию на контракт 0x159AE4d7012B18A0dE4e390714a6efa036487f0b, вызывающую метод commit с параметром

   ```
   ("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e", 5000000000000000000,
   ↪  "0xddfbb81ca5f813ff658bc60df9d23a464cec78966b90d98aa9c09ce4045bb285")
   ```

   Транзакция исполняется успешно. У пользователя 0x79dD14623c4D33413c0c28fDAbC2285Fdb после исполнения транзакции становится на 50 нативных токенов больше. Баланс контракта моста меняется до 50 нативных токенов.
8. Пользователь, чей приватный ключ соответствует адресу 0x79dD14623c4D33413c0c28fDAbC22 отправляет 5 нативных токенов на адрес 0x159AE4d7012B18A0dE4e390714a6efa036487f0b. Транзакция исполняется успешно. Баланс пользователя уменьшается на 5 токенов. Баланс моста увеличивается на 5 токенов.
9. Администратор моста отправляет в контракт 0x159AE4d7012B18A0dE4e390714a6efa036487f0 транзакцию, вызывающую метод stopOperations. Транзакция исполняется успешно.
10. Пользователь, чей приватный ключ соответствует адресу 0x79dD14623c4D33413c0c28fDAbC22 вызывает JSON RPC метод eth_estimateGas для оценки потребления газа перед посылкой транзакции на пересылку 5 нативных токенов на адрес 0x159AE4d7012B18A0dE4 Вызов eth_estimateGas возвращает ошибку, сообщающую о том, что в ходе вызова метода контракта выполнилась команда revert.
11. Аккаунт с адресом 0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc отправляет транзакцию на контракт 0x159AE4d7012B18A0dE4e390714a6efa036487f0b, вызывающую метод commit с параметром

    ```
    ("0x79dD14623c4D33413c0c28fDAbC2285Fdb1e572e", 5000000000000000000,
    ↪  "0x930650054b50608dafb24c08b71f46bf8605de83d9ad5502cad0c6159dd074bf")
    ```

Транзакция исполняется успешно. У пользователя 0x79dD14623c4D33413c0c28fDAbC2285Fdb
после исполнения транзакции становится на 50 нативных токенов больше. Баланс контракта моста меняется на 50 нативных токенов.

12. Администратор моста отправляет в контракт 0x159AE4d7012B18A0dE4e390714a6efa036487f0 транзакцию, вызывающую метод `startOperations`. Транзакция исполняется успешно.
13. Пользователь, чей приватный ключ соответствует адресу 0x79dD14623c4D33413c0c28fDAbC22 отправляет 5 нативных токенов на адрес 0x159AE4d7012B18A0dE4e390714a6efa036487f0b. Транзакция исполняется успешно. Баланс пользователя уменьшается на 5 токенов. Баланс моста увеличивается на 5 токенов.

US-024 Контракт моста: ожидание финализации цепочки блоков

Описание: Я, как пользователь, хочу быть уверенным, что мостом безопасно пользоваться при кратковременных разветвлениях цепочки блоков, возможных либо во время штатной работы сети (два блока с одним номером выпущены примерно в одинаковое время), либо во время атаки – транзакции включенных в блоки, которые позднее не войдут в основную цепочку, не будут переданы через мост.

В данной US подразумевается, что оракул начинает полагаться еще на одну переменную окружения `REQUIRED_CONFIRMATIONS`, которая устанавливает количество блоков, которые оракул должен пропустить, прежде чем расценивать цепочку с номерами блоков, меньше текущего на указанное число, как завершенную. Если переменная отсутствует, то подразумевается, что блоки не нужно пропускать (также как и при `REQUIRED_CONFIRMATIONS=0`). Данная настройка применяется к каждой сети, куда направлен оракул.

Пояснение: предположим, что `REQUIRED_CONFIRMATIONS=12` при запуске оракула. Это значит, что если в блоке 54698 появилось событие `bridgeActionInitiated`, то оракул ждет пока над блоком 54698 не появится еще 12 блоков, прежде чем передаст свое подтверждение об инициации передачи токенов через мост.

В тестах, перечисленных ниже, за исключением случаев, где это специально оговорено, считается, что

1. Существуют две блокчейн сети:
   - Сеть `LEFT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c
   - Сеть `RIGHT`:
     - максимальное количество газа, которое может быть потрачено транзакциями в блоке `12'500'000`
     - время выпуска блоков в среднем раз в 5 секунд
     - контракт моста находится по адресу 0x159AE4d7012B18A0dE4e390714a6efa036487
2. Контракты моста зарегистрированы в блокчейн сетях при следующем содержимом переменных окружения:

```
VALIDATORS=0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc
```

```
THRESHOLD=1
```

3. Установлена ликвидность моста в 2000 токенов.
4. Оракул запускается при содежимом файла `.env`:

```
PRIVKEY=<private key corresponding to
↪  0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc>
LEFT_RPCURL=https://sokol.poa.network
LEFT_ADDRESS=0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441
LEFT_START_BLOCK=17290597
LEFT_GASPRICE=5000000000
RIGHT_RPCURL=https://data-seed-prebsc-1-s1.binance.org:8545/
RIGHT_ADDRESS=0x159AE4d7012B18A0dE4e390714a6efa036487f0b
RIGHT_START_BLOCK=5847797
RIGHT_GASPRICE=10000000000
ORACLE_DATA=/some/path
REQUIRED_CONFIRMATIONS=12
```

Критерий оценивания (AC-024-01) : Отправка подтверждения в обычном режиме

---

1. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 1000 нативных токенов на адрес `0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441`. Транзакция входит в блок `B` и исполняется успешно. Баланс пользователя в сети `LEFT` уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
   В сети `RIGHT` в одном из блоков, выпущенном чуть позже после того, как в сети `LEFT` выпущен блок `B+12`, есть транзакция от `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc`. При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `RIGHT` на момент завершения блока, в который включена успешная транзакция от оракула, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 0 нативных токенов.
2. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 50 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция входит в блок ‘B*‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов.
   В сети `LEFT` в одном из блоков, выпущенном чуть позже после того, как в сети `RIGHT` выпущен блок `B*+12`, есть транзакция от `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3`. При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `LEFT` на момент завершения блока, в который включена успешная транзакция от оракула, видно, что на 50 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 50 нативных токенов.
3. Пользователь, чей приватный ключ соответствует адресу `0x5580ba66f8d6dc71adb0ca1d1c6b3` отправляет 30 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция входит в блок ‘B**‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 30 токенов. Баланс моста увеличивается на 30 токенов.

4. Пока еще в сети RIGHT не выпустилось 12 блоков, оракул останавливается, и его .env файл меняется так, что в RIGHT_RPCURL указан URL узла в сети RIGHT, который из-за ошибки синхронизации отстает от основной сети на 24 блока. Оракул запускается снова.
5. В сети LEFT в одном из блоков, выпущенном чуть позже после того, как в сети RIGHT выпущен блок B**+12, есть транзакция от 0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC
   При опросе баланса пользователя 0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca в сети LEFT на момент завершения блока, в который включена успешная транзакция от оракула, видно, что на 30 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 30 нативных токенов.

Критерий оценивания (AC-024-02) : Отправка подтверждения в режиме экономичной пересылки

---

1. На каждой стороне моста администратор посылает тарназкции вызывающие enableRobustMode в контрактах моста. Транзакции исполняются успешно.
2. Пользователь, чей приватный ключ соответствует адресу 0x5580ba66f8d6dc71adb0ca1d1c6b3
   отправляет 1000 нативных токенов на адрес 0xfab1ed72a7236a6b34f47ee7Ed103d6CD448c441
   Транзакция входит в блок B и исполняется успешно. Баланс пользователя в сети LEFT уменьшается на 1000 токенов. Баланс моста увеличивается до 1000 токенов.
   В сети RIGHT в одном из блоков, выпущенном чуть позже после того, как в сети LEFT выпущен блок B+12, есть транзакция от 0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc
   При опросе баланса пользователя 0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca в сети RIGHT на момент завершения блока, в который включена успешная транзакция от оракула, видно, что на 1000 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение до 0 нативных токенов.
3. Пользователь, чей приватный ключ соответствует адресу 0x5580ba66f8d6dc71adb0ca1d1c6b3
   отправляет 50 нативных токенов на адрес 0x159AE4d7012B18A0dE4e390714a6efa036487f0b.
   Транзакция 0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804 входит в блок ‘B*‘ и исполняется успешно. Баланс пользователя в сети RIGHT уменьшается на 50 токенов. Баланс моста увеличивается на 50 токенов.
   В сети RIGHT в одном из блоков C с номером выше B*+12 есть транзакция от 0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc. В выписке транзакции есть событие:

   ```
   ↪ commitsCollected("0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804",
   ↪ 1)
   ```

   В сети LEFT в одном из блоков, выпущенном чуть позже после того, как в сети RIGHT выпущен блок C+12, есть транзакция от 0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3F
   При опросе баланса пользователя 0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca в сети LEFT на момент завершения блока, в который включена транзакция оракула, видно, что на 50 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 50 нативных токенов.
4. Пользователь, чей приватный ключ соответствует адресу 0x5580ba66f8d6dc71adb0ca1d1c6b3

отправляет 30 нативных токенов на адрес `0x159AE4d7012B18A0dE4e390714a6efa036487f0b`. Транзакция `0xb7a8533489fc70301bee59e5f8760d8db830be076b7a6acf352f8fa6d9d804b5` входит в блок ‘B**‘ и исполняется успешно. Баланс пользователя в сети `RIGHT` уменьшается на 30 токенов. Баланс моста увеличивается на 30 токенов.
В сети `RIGHT` в одном из блоков ‘C*‘ с номером выше `B**+12` есть транзакция от `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3Fc`. В выписке транзакции есть событие:

```
↪ commitsCollected("0xb7a8533489fc70301bee59e5f8760d8db830be076b7a6acf352f8fa6d9d804b5",
↪ 1)
```

5. Пока еще в сети `RIGHT` не выпустилось 12 блоков после блока ‘C*‘, оракул останавливается, и его `.env` файл меняется так, что в `RIGHT_RPCURL` указан URL узла в сети `RIGHT`, который из-за ошибки синхронизации отстает от основной сети на 24 блока. Оракул запускается снова.
6. В сети `LEFT` в одном из блоков, выпущенном чуть позже после того, как в сети `RIGHT` выпущен блок `C*+12`, есть транзакция от `0xaF1E1D6C3cedD99DA7df3a91F5B956AB6d2eC3`
При опросе баланса пользователя `0x5580ba66f8d6dc71adb0ca1d1c6b3d142ff7aaca` в сети `LEFT` на момент завершения блока, в который включена успешная транзакция от оракула, видно, что на 30 нативных токенов стало больше. При аналогичном запросе для баланса моста видно его уменьшение на 30 нативных токенов.

Решение

blockchain.py

```
import logging
import os
import re
import time
from subprocess import check_output
from typing import Tuple

import requests
from eth_abi import encode_abi
from eth_account.signers.local import LocalAccount
from solcx import compile_files
from web3 import Web3, HTTPProvider, Account
from web3.exceptions import TransactionNotFound
from web3.middleware import geth_poa_middleware

from config import (
    LEFT_GASPRICE, RIGHT_GASPRICE,
    LEFT_RPCURL, RIGHT_RPCURL,
    PRIVKEY, DEFAULTGAS
)

if PRIVKEY:
    account = Account.privateKeyToAccount(PRIVKEY)
else:
    account = None


def get_prepared_web3(rpc_url: str) -> Web3:
    web3 = Web3(HTTPProvider(rpc_url))
    web3.middleware_onion.inject(geth_poa_middleware, layer=0)
    if account:
        web3.eth.defaultAccount = account.address
    return web3


web3_left = get_prepared_web3(LEFT_RPCURL)
web3_right = get_prepared_web3(RIGHT_RPCURL)

LEFT_CONNECTED = web3_left.isConnected()
RIGHT_CONNECTED = web3_right.isConnected()
web3_left.eth.custom_defaultGasPrice = LEFT_GASPRICE
web3_right.eth.custom_defaultGasPrice = RIGHT_GASPRICE

if PRIVKEY:
    account = web3_right.eth.account.from_key(PRIVKEY)
    web3_left.eth.defaultAccount = account.address
    web3_right.eth.defaultAccount = account.address
else:
    account = None

ZERO_ADDRESS = '0x' + '0' * 40
normal_addr = Web3.toChecksumAddress
keccak = Web3.keccak
```

```
class TransactionReverted(Exception):
    pass


class NotEnoughFunds(Exception):
    def __init__(self, address, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.address = address


def default_nonce(web3, account):
    return web3.eth.getTransactionCount(account.address)


def get_deployed_contract(web3, address, abi):
    return web3.eth.contract(address=address, abi=abi)


def balanceOf(web3, address):
    return web3.eth.getBalance(address)


def extract_solc_version() -> str:
    output = check_output(["solc", "--version"]).decode()
    result = re.search(r"Version: (.+)\.Linux\.g\+\+", output)
    assert result, output
    version: str = result.group(1)
    if not version.startswith('v'):
        version = 'v' + version

    return version


def compile_contract(contract_path, contract_name) -> Tuple[str, dict]:
    """
    Return bytecode and abi of
    `contract_name` in `contract_path` file
    """
    if not os.path.exists(contract_path):
        raise FileNotFoundError(contract_path)
    artifacts = compile_files(contract_path, optimize=True, solc_binary='solc',
    ↪  output_values=['abi', 'bin'])
    compiled = artifacts.get(f'{contract_path}:{contract_name}')

    return compiled['bin'], compiled['abi']


def deploy_contract(web3, path, name, account=account, gas=None, args=(),
↪  abi_of_args=None, verify=False):
    """
    Deploy contract with given bytecode and abi
    using `account`

    `args` will be passed to contract constructor

    If `verify` is true, then try to verify contract on blockscout

    Return transaction receipt of contract deployment
    """
    contract_bytecode, contract_abi = compile_contract(path, name)
```

```
    contract = web3.eth.contract(bytecode=contract_bytecode, abi=contract_abi)
    try:
        tx = contract.constructor(*args).buildTransaction()
    except ValueError as e:
        logging.error(f"Cannot evaluate gas for constructor with args {args}")
        logging.error(e)
        return

    data = tx['data']
    gas = gas or tx['gas']

    txReceipt = build_and_send_tx(web3, _from=account, to='', data=data, gas=gas)

    assert txReceipt['status']
    contractAddress = txReceipt['contractAddress']

    logging.info(f'Contract {name} deployed at {contractAddress}')

    if verify:
        with open(path) as file:
            source_code = file.read()

        abiEncodedArgs = encode_abi(abi_of_args, args).hex()
        contract_info = {
            "address": contractAddress,
            "source_code": source_code,
            "name": name,
            "compiler_version": extract_solc_version(),
            "constructor_arguments": abiEncodedArgs,
        }
        logging.debug(f"Contract info: {contract_info}")
        verified = verify_contract(
            **contract_info
        )

        if verified:
            logging.info(f'Contract {name} is verified')
        else:
            raise ValueError(f"Contract is not verified. address: {contractAddress}.")

    return txReceipt


def verify_contract(address, source_code, compiler_version, name,
↪   constructor_arguments, optimization=200) -> bool:
    """
    Try to verify contract
    """
    url = 'https://blockscout.com/poa/sokol/api?module=contract&action=verify'
    data = {
        "addressHash": address,
        "compilerVersion": compiler_version,
        "contractSourceCode": source_code,
        "name": name,
        "optimization": True if optimization else False,
        "optimizationRuns": optimization,
        "constructorArguments": constructor_arguments,
        "evmVersion": 'default',
    }
    length = len(source_code)
```

```
    logging.info(
        f"Start verifying contract {name}, address={address}, source code
        ↪  length={length}")

    MAX_REQUESTS = 10
    # in seconds
    DELAY_BETWEEN_REQUESTS = 10
    for _ in range(MAX_REQUESTS):
        response = requests.post(url, json=data, timeout=120)
        logging.info(f"Got response {response}")

        verified = response.ok

        if not verified:
            logging.info(f"Sleep for {DELAY_BETWEEN_REQUESTS} seconds")
            time.sleep(DELAY_BETWEEN_REQUESTS)
        else:
            break

    return verified


def build_and_send_tx(web3, _from, to, data='', value=0, gas=None, gasPrice=None,
↪   nonce=None):
    """
    Build, sign, send transaction and wait for its receipt
    """

    if not isinstance(_from, LocalAccount):
        _from = web3.eth.account.from_key(_from)

    nonce = nonce or default_nonce(web3, _from)
    tx = {
        'from': _from.address,
        'to': to,
        'nonce': nonce,
        'data': data or '',
        'value': value,
        'gasPrice': gasPrice or web3.eth.custom_defaultGasPrice,
        'gas': gas or DEFAULTGAS
    }

    return send_tx(web3, tx)


def send_tx(web3, tx, account=account):
    if 'nonce' not in tx:
        tx['nonce'] = default_nonce(web3, account)

    signed = account.signTransaction(tx)
    try:
        tx_hash = web3.eth.sendRawTransaction(signed.rawTransaction)
    except ValueError as e:
        if tx['gasPrice'] * tx['gas'] > web3.eth.getBalance(account.address):
            raise NotEnoughFunds(address=account.address)
        else:
            raise e
    return wait_tx_receipt(web3, tx_hash)
```

```
def wait_tx_receipt(web3, tx_hash, sleep_interval=0.5, max_tries=20):
    """
    Wait for transaction receipt with given sleep interval
    Stop after `max_tries` failed attempts
    """

    failed = 0
    while failed < max_tries:
        try:
            tx_receipt = web3.eth.getTransactionReceipt(tx_hash)
        except TransactionNotFound:
            tx_receipt = None
        if tx_receipt:
            return tx_receipt
        failed += 1
        time.sleep(sleep_interval)

    logging.error(f"Can not extract transaction")
```

config.py

```
import os

from dotenv import load_dotenv
from web3 import Web3

load_dotenv('.env')

PRIVKEY = os.environ.get('PRIVKEY')

LEFT_RPCURL = os.environ.get('LEFT_RPCURL')

LEFT_GASPRICE = int(os.environ.get('LEFT_GASPRICE', '-1'))

LEFT_ADDRESS = os.environ.get('LEFT_ADDRESS', '')
if LEFT_ADDRESS:
    LEFT_ADDRESS = Web3.toChecksumAddress(LEFT_ADDRESS)

LEFT_START_BLOCK = int(os.environ.get('LEFT_START_BLOCK', '-1'))

RIGHT_RPCURL = os.environ.get('RIGHT_RPCURL')

RIGHT_GASPRICE = int(os.environ.get('RIGHT_GASPRICE', '-1'))

RIGHT_ADDRESS = os.environ.get('RIGHT_ADDRESS', '')
if RIGHT_ADDRESS:
    RIGHT_ADDRESS = Web3.toChecksumAddress(RIGHT_ADDRESS)

RIGHT_START_BLOCK = int(os.environ.get('RIGHT_START_BLOCK', '-1'))

VALIDATORS = os.environ.get('VALIDATORS', '').split()

THRESHOLD = int(os.environ.get('THRESHOLD', '-1'))

LOCAL = os.environ.get('LOCAL', 'false')

DEFAULTGAS = 2000001

TIMEOUT_RECEIPT_CHECKER = 12000
```

```
REQUIRED_CONFIRMATIONS = int(os.environ.get('REQUIRED_CONFIRMATIONS', '0'))

CONTRACT_VALIDATORS = {
    # deployment directory
    'path': 'flats/BridgeValidators_flat.sol',

    'name': 'BridgeValidators',
    'abi_of_args': ['address[]', 'uint256', 'address'],
}

CONTRACT_LEFT = {
    # deployment directory
    'path': 'flats/LeftBridge_flat.sol',

    'name': 'LeftBridge',
    'abi_of_args': ['address', 'address'],
}

CONTRACT_RIGHT = {
    # deployment directory
    'path': 'flats/RightBridge_flat.sol',

    'name': 'RightBridge',
    'abi_of_args': ['address', 'address'],
}


REDIS_CONFIG = {
    'host': 'redis',
    'port': 5432,
}
```

crypto.py

```
from eth_account.messages import encode_defunct
from blockchain import account
from web3 import Web3
from web3 import Account
from eth_account.messages import (
    SignableMessage,
    _hash_eip191_message,
)
from eth_abi.packed import encode_single_packed, encode_abi_packed

def prepareSignature(account, recipient, value, txHash):
    """
    Signs message and returns r s v
    """
    msg = Web3.solidityKeccak(['address', 'uint256', 'bytes32'], [recipient, value,
    ↪  txHash])
    #msg = encode_abi_packed(['address', 'uint256', 'bytes32'], [recipient, value,
    ↪  txHash])
    wrapped_msg = encode_defunct(msg)
    signed = account.sign_message(wrapped_msg)
    v, r, s = signed.v, signed.r, signed.s
    return r, s, v

if __name__ == '__main__':
```

```
    recipient = Web3.toChecksumAddress("0xF91296B9d7699d800c2Ba0a80e755972a9DA7C34")
    value = Web3.toWei(10, 'ether')
    txhash = "0xac21bd9c034f02266f79ac064560ea1ed3e8b8ff6491f0f8e433e9122ae4e804"
    key = "0x930650054b50608dafb24c08b71f46bf8605de83d9ad5502cad0c6159dd074bf"

    r, s, v = prepareSignature(account, recipient, value, txhash)
    print(account.address)
    print(recipient, value, txhash)
    print(r, s, v)
```

deployment/contracts/Address.sol

```
pragma solidity 0.7.6;

import "./Sacrifice.sol";

/**
 * @title Address
 * @dev Helper methods for Address type.
 */
library Address {
    /**
    * @dev Try to send native tokens to the address. If it fails, it will force the
↪   transfer by creating a selfdestruct contract
    * @param _receiver address that will receive the native tokens
    * @param _value the amount of native tokens to send
    */
    function safeSendValue(address payable _receiver, uint256 _value) internal {
        if (!_receiver.send(_value)) {
            (new Sacrifice){value:_value}(_receiver);
        }
    }
}
```

deployment/contracts/AddressRecoveryTool.sol

```
pragma solidity ^0.7.6;

contract AddressRecoveryTool {
    function wrapped(address _a, uint256 _v, bytes32 _h) pure internal
    ↪   returns(bytes32)  {
        return keccak256(abi.encodePacked('\x19Ethereum Signed Message:\n32',
        ↪   getRobustModeMessage(_a, _v, _h)));
    }

    function getRobustModeMessage(address recipient, uint256 amount, bytes32 id) pure
    ↪   public returns (bytes32){
        return keccak256(abi.encodePacked(recipient, amount, id));
    }

    function recover(address _rcpt, uint256 _value, bytes32 _txhash, uint8 _v, bytes32
    ↪   _r, bytes32 _s) pure internal returns(address) {
        return ecrecover(wrapped(_rcpt, _value, _txhash), _v, _r, _s);
    }
}
```

deployment/contracts/BaseBridge.sol

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.7.6;

import "./Ownable.sol";
import "./Validable.sol";
import "./Address.sol";


contract BaseBridge is Ownable, Validable {
    struct Action{
        uint256 numMessagesSigned;
    }

    event bridgeActionInitiated(address recipient, uint256 amount);
    event SignedForUserRequest(address indexed signer, bytes32 messageHash);
    event MessageConfirmed(address recipient, uint256 value, bytes32 transactionHash);

    mapping (bytes32 => uint256) m_numMessagesSigned;
    mapping (bytes32 => bool) m_signed;



    uint256 m_balanceOnDiffBridge;

    bool m_robustMode = false;

    receive() payable external {
        require(msg.data.length == 0);
        nativeTransfer(msg.sender);
    }

    function isRobustMode() public view returns (bool) {
        return m_robustMode;
    }

    function _balanceOnDiffBridge() internal view returns (uint256){
        return m_balanceOnDiffBridge;
    }

    function nativeTransfer(address _receiver) internal checkLimit(msg.value)
    ↪  bridgeIsUp{
        uint256 valueToTransfer = msg.value;
        require(_balanceOnDiffBridge() >= valueToTransfer);
        m_balanceOnDiffBridge -= valueToTransfer;

        emit bridgeActionInitiated(_receiver, valueToTransfer);
    }

    function _numMessagesSigned(bytes32 _transactionId) public view returns (uint256){
        return m_numMessagesSigned[_transactionId];
    }

    function _accountHasSigned(bytes32 _transactionId, address _validator) public view
    ↪  returns (bool){
        return m_signed[keccak256(abi.encodePacked(_transactionId, _validator))];
    }
```

```
function _setAccountSigned(bytes32 _transactionId, address _validator, bool
↪  _value) internal{
    m_signed[keccak256(abi.encodePacked(_transactionId, _validator))] = _value;
}

function _isAlreadyProcessed(uint256 _number) internal pure returns (bool) {
    return _number & (2**255) == 2**255;
}

function _markAsProcessed(uint256 _v) internal pure returns (uint256) {
    return _v | (2**255);
}

function _commit(address payable _recipient, uint256 _amount, bytes32 _txHash)
↪  internal checkLimit(_amount){

    bytes32 transactionId = _txHash;

    uint256 signed = _numMessagesSigned(transactionId);

    require(!_isAlreadyProcessed(signed), "This transaction was already
    ↪  submitted");

    require(!_accountHasSigned(transactionId, msg.sender), "This account already
    ↪  submitted sign");

    _setAccountSigned(transactionId, msg.sender, true);
    signed += 1;
    emit SignedForUserRequest(msg.sender, transactionId);

    uint256 reqSigs = requiredSignatures();
    if (signed >= reqSigs){
        m_numMessagesSigned[transactionId] = _markAsProcessed(signed);
        m_balanceOnDiffBridge += _amount;
        emit MessageConfirmed(_recipient, _amount, _txHash);
        sendNativeTokens(_recipient, _amount);
        // TODO: remove all values in m_signed for every validator?
    } else {
        m_numMessagesSigned[transactionId] = signed;
    }
}

function sendNativeTokens(address payable _recipient, uint256 _amount) internal {
    Address.safeSendValue(_recipient, _amount);
}

function changeValidatorSet(address newvalidatorset) public onlyOwner {
    _setValidatorContract(newvalidatorset);
}

function enableRobustMode() public onlyOwner {
    m_robustMode = true;
}

function getMaxLimit() public view returns (uint256){
    return maxLimit;
}

function getMinLimit() public view returns (uint256){
    return minLimit;
```

```
    }

    function setMaxPerTx(uint256 limit) public onlyOwner{
        maxLimit = limit;
    }

    function setMinPerTx(uint256 limit) public onlyOwner{
        minLimit = limit;
    }

    function stopOperations() public onlyOwner{
        isBridgeUp = false;
    }

    function startOperations() public onlyOwner{
        isBridgeUp = true;
    }

}
```

deployment/contracts/BridgeValidators.sol

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.7.6;

import "./Ownable.sol";


contract BaseBridgeValidators is Ownable {
    address public constant F_ADDR = 0xFFfFfFffFFfffFFfFFfFFFFFffFFFffffFfFFFfF;


    event ValidatorAdded(address indexed validator);
    event ValidatorRemoved(address indexed validator);
    event RequiredSignaturesChanged(uint256 requiredSignatures);

    uint public m_validatorCount;
    uint m_required_signatures;

    mapping (bytes32 => address) validator_list;


    function getThreshold() public view returns (uint256){
        return m_required_signatures;
    }

    function changeThreshold(uint256 thresh) public onlyOwner {
        _changeThreshold(thresh);
        emit RequiredSignaturesChanged(thresh);
    }

    function _changeThreshold(uint256 thresh) internal {
        require(m_validatorCount >= thresh, "# of validators < than requested
        ↪  number");
        require(thresh != 0, "Zero _requiredSignatures");
        m_required_signatures = thresh;
    }
```

```
function isValidator(address _validator) public view returns (bool) {
    return _validator != F_ADDR && _getNextValidator(_validator) != address(0);
}

/*
    Set next validator for `_prevValidator` to `_validator`
*/
function _setNextValidator(address _prevValidator, address _validator) internal{
    validator_list[keccak256(abi.encodePacked("validatorList", _prevValidator))] =
    ↪  _validator;
}

/*
    Return next validator for `_validator`.
    If `_validator` is the last in the list,
    return 0xFF..FF
*/
function _getNextValidator(address _validator) internal view returns (address) {
    return validator_list[keccak256(abi.encodePacked("validatorList",
    ↪  _validator))];
}

/*
    Delete `_validator` from the storage
*/
function _deleteValidatorFromList(address _validator) internal {
    delete validator_list[keccak256(abi.encodePacked("validatorList",
    ↪  _validator))];
}

/*
    Return previous validator for `_validator` in the validator list
*/
function _getPrevValidator(address _validator) internal view returns (address) {
    address index = F_ADDR;
    address nextIndex = _getNextValidator(index);

    require(nextIndex != address(0), "there is no such validator in list");

    while (nextIndex != _validator) {

        index = nextIndex;
        nextIndex = _getNextValidator(index);

        require(nextIndex != F_ADDR && nextIndex != address(0), "there is no such
        ↪  validator in list");
    }
    return index;
}

/*
    Add `_validator` to the begging of the validator list
*/
function _addValidator(address _validator) internal {
    // check preconditions
    require(!isValidator(_validator), "There is already such validator");

    address firstValidator = _getNextValidator(F_ADDR);
    // there is no validators
```

```
        if (firstValidator == address(0)){
            firstValidator = F_ADDR;
        }
        _setNextValidator(_validator, firstValidator);
        _setNextValidator(F_ADDR, _validator);
        m_validatorCount++;

    }

    /*
        Remove `_validator` from the validator list
    */
    function _removeValidator(address _validator) internal {
        // check preconditions
        require(m_validatorCount > m_required_signatures, "invalid number of
        ↪  validators");
        require(isValidator(_validator), "No such validator");

        address prevValidator = _getPrevValidator(_validator);
        address nextValidator = _getNextValidator(_validator);

        _setNextValidator(prevValidator, nextValidator);
        _deleteValidatorFromList(_validator);
        m_validatorCount--;
    }

    /*
        Return list of all validators
    */
    function _validatorList() internal view returns (address[] memory) {
        address[] memory list = new address[](m_validatorCount);
        uint256 counter = 0;
        address nextValidator = _getNextValidator(F_ADDR);

        if (nextValidator == address(0)){
            return list;
        }

        while (nextValidator != F_ADDR) {
            list[counter] = nextValidator;
            nextValidator = _getNextValidator(nextValidator);
            counter++;

            require(nextValidator != address(0));
        }

        return list;
    }

}


contract BridgeValidators is BaseBridgeValidators{
    constructor (address[] memory _validators, uint256 _requiredSignatures, address
    ↪  _owner) {
        _setOwner(_owner);

        // add validator from validator list
        for (uint i = 0; i < _validators.length; ++i){
            _addValidator(_validators[i]);
```

```
            emit ValidatorAdded(_validators[i]);
        }

        // set threshold
        _changeThreshold(_requiredSignatures);
        emit RequiredSignaturesChanged(_requiredSignatures);
    }


    function addValidator(address newvalidator) public onlyOwner {
        _addValidator(newvalidator);
        emit ValidatorAdded(newvalidator);
    }

    function removeValidator(address validator) public onlyOwner {
        _removeValidator(validator);
        emit ValidatorRemoved(validator);
    }

    function getValidators() public view returns (address[] memory) {
        return _validatorList();
    }

}
```

deployment/contracts/IBridgeValidators.sol

```
pragma solidity ^0.7.6;

interface IBridgeValidators {
    function isValidator(address _validator) external view returns (bool);
    function getThreshold() external view returns (uint256);
    function owner() external view returns (address);
}
```

deployment/contracts/LeftBridge.sol

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.7.6;

import "./BaseBridge.sol";
import "./AddressRecoveryTool.sol";

contract LeftBridge is BaseBridge, AddressRecoveryTool {

    constructor (address validatorContract, address _owner) {
        _setValidatorContract(validatorContract);
        _setOwner(_owner);
        setMinPerTx(0 ether);
        setMaxPerTx(0 ether);
        startOperations();
    }

    function getLiquidityLimit() public view returns (uint256){
        return _balanceOnDiffBridge();
    }
```

```
    function updateLiquidityLimit(uint256 newlimit) public onlyOwner {
        require(newlimit >= 0);
        m_balanceOnDiffBridge = newlimit;
    }

    function commit(address payable _recipient, uint256 _amount, bytes32 _txHash)
    ↪  external onlyValidator {
        require(!m_robustMode, "robust mode");
        _commit(_recipient, _amount, _txHash);
    }

    function applyCommits(address recipient, uint256 amount, bytes32 id, uint256[]
    ↪  memory r, uint256[] memory s, uint8[] memory v) external {
        require(isRobustMode(), "Not robust mode");
        require((r.length == s.length) && (s.length == v.length));
        require(requiredSignatures() == r.length);



        uint256 signed;

        for (uint256 i = 0; i < r.length; i++) {
            address address_ = recover(recipient, amount, id, uint8(v[i]),
            ↪  bytes32(r[i]), bytes32(s[i]));

            require(isValidator(address_), string(abi.encodePacked(address_, " not a
            ↪  validator")));

            require(!_isAlreadyProcessed(signed), "This transaction was already
            ↪  submitted");

            require(!_accountHasSigned(id, address_), "This account already submitted
            ↪  sign");

            signed = _numMessagesSigned(id);

            _setAccountSigned(id, address_, true);

        }

        m_numMessagesSigned[id] = _markAsProcessed(signed);

        sendNativeTokens(payable(recipient), amount);
        emit MessageConfirmed(recipient, amount, id);
    }
}
```

deployment/contracts/Ownable.sol

```
pragma solidity ^0.7.6;


contract Ownable{
    address m_owner;
    event OwnershipTransferred(address previousOwner, address newOwner);


    modifier onlyOwner() {
```

```
        require(msg.sender == m_owner, "Not owner");
        /* solcov ignore next */
        _;
    }

    /**
    * @dev Allows the current owner to transfer control of the contract to a newOwner.
    * @param newOwner the address to transfer ownership to.
    */
    function transferOwnership(address newOwner) external onlyOwner {
        require(newOwner != m_owner);
        _setOwner(newOwner);
    }

    /**
    * @dev Sets a new owner address
    */
    function _setOwner(address newOwner) internal {
        require(newOwner != address(0), "Zero address for setOwner");
        emit OwnershipTransferred(m_owner, newOwner);
        m_owner = newOwner;
    }
}
```

deployment/contracts/RightBridge.sol

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.7.6;

import "./BaseBridge.sol";
import "./AddressRecoveryTool.sol";


contract RightBridge is BaseBridge, AddressRecoveryTool {
    struct RobustAction {
        address recipient;
        uint256 amount;
        uint256[] r;
        uint256[] s;
        uint8[] v;
        address assignee;
    }

    event CommitRegistered(address indexed signer, bytes32 messageHash);
    event commitsCollected(bytes32 id, uint8 commits);


    mapping(bytes32 => RobustAction) m_actions;

    function getTransferDetails(bytes32 txHash) public view returns (address
    ↪  recipient, uint256 amount) {
        recipient = m_actions[txHash].recipient;
        amount = m_actions[txHash].amount;
    }

    function getCommit(bytes32 txHash, uint8 index) public view returns (uint256 r,
    ↪  uint256 s, uint8 v) {
        r = m_actions[txHash].r[index];
```

```
        s = m_actions[txHash].s[index];
        v = m_actions[txHash].v[index];
    }

    constructor (address validatorContract, address _owner) {
        _setValidatorContract(validatorContract);
        _setOwner(_owner);
        setMinPerTx(0 ether);
        setMaxPerTx(0 ether);
        startOperations();
    }

    function addLiquidity() payable external onlyOwner {
        require(msg.value > 0, "Zero value");
    }

    function commit(address payable _recipient, uint256 _amount, bytes32 _txHash)
    ↪  external onlyValidator {
        _commit(_recipient, _amount, _txHash);
    }


    function registerCommit(address recipient, uint256 amount, bytes32 txHash, uint256
    ↪  r, uint256 s, uint8 v) external onlyValidator checkLimit(amount) {
        require(m_robustMode, "Not robust mode");
        require(recover(recipient, amount, txHash, v, bytes32(r), bytes32(s)) ==
        ↪  msg.sender);
        uint256 signed = _numMessagesSigned(txHash);

        require(!_isAlreadyProcessed(signed), "This transaction was already
        ↪  submitted");

        require(!_accountHasSigned(txHash, msg.sender), "This account already
        ↪  submitted sign");

        _setAccountSigned(txHash, msg.sender, true);
        uint256 reqSigs = requiredSignatures();

        if (signed == 0) {
            m_actions[txHash].r = new uint256[](reqSigs);
            m_actions[txHash].s = new uint256[](reqSigs);
            m_actions[txHash].v = new uint8[](reqSigs);
        }

        m_actions[txHash].r[signed] = r;
        m_actions[txHash].s[signed] = s;
        m_actions[txHash].v[signed] = v;
        m_actions[txHash].amount = amount;
        m_actions[txHash].recipient = recipient;

        signed += 1;
        emit CommitRegistered(msg.sender, txHash);

        if (signed >= reqSigs) {
            m_numMessagesSigned[txHash] = _markAsProcessed(signed);
            m_actions[txHash].assignee = msg.sender;
            emit commitsCollected(txHash, uint8(signed));
        }
        else {
            m_numMessagesSigned[txHash] = signed;
```

```
        }
    }

    function getAssignee(bytes32 txHash) public view returns(address assignee){
        assignee = m_actions[txHash].assignee;
    }

    function getLiquidityLimit() public view returns (uint256){
        return address(this).balance;
    }
}
```

deployment/contracts/Sacrifice.sol

```
pragma solidity 0.7.6;

contract Sacrifice {
    constructor(address payable _recipient) payable {
        selfdestruct(_recipient);
    }
}
```

deployment/contracts/Validable.sol

```
pragma solidity ^0.7.6;


import "./IBridgeValidators.sol";


contract Validable {
    address m_validatorContract;
    uint256 minLimit;
    uint256 maxLimit;
    bool isBridgeUp;

    function _setValidatorContract(address _validatorContract) internal {
        require(_validatorContract != m_validatorContract);
        m_validatorContract = _validatorContract;
    }

    function validatorContract() public view returns (IBridgeValidators) {
        return IBridgeValidators(m_validatorContract);
    }
    function isValidator(address validator) public view returns (bool){
        return validatorContract().isValidator(validator);
    }

    modifier onlyValidator() {
        require(isValidator(msg.sender));
        _;
    }

    modifier checkLimit(uint256 amount){

        if (maxLimit == 0){
            require(amount > minLimit);
```

```
        } else {
            require(amount > minLimit && amount < maxLimit);
        }
        _;
    }

    function requiredSignatures() public view returns (uint256) {
        return validatorContract().getThreshold();
    }

    modifier bridgeIsUp(){
        require(isBridgeUp);
        _;
    }
}
```

deployment/deploy.py

```
import argparse
import logging
from json import dump

from blockchain import (NotEnoughFunds, account, compile_contract,
                        deploy_contract, normal_addr, web3_right, web3_left,
                        LEFT_CONNECTED, RIGHT_CONNECTED)
from utils import exit_with_message

from config import (CONTRACT_RIGHT, CONTRACT_LEFT, CONTRACT_VALIDATORS,
                    THRESHOLD, VALIDATORS, LEFT_RPCURL, RIGHT_RPCURL)


def parse_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--verify', action='store_true',
        help='if true then verify the contracts',
    )

    return parser.parse_args()


if __name__ == '__main__':
    logging.basicConfig(level=logging.INFO)
    if not LEFT_CONNECTED:
        exit_with_message(f"Cannot connect to left rpc: {LEFT_RPCURL}")
    if not RIGHT_CONNECTED:
        exit_with_message(f"Cannot connect to right rpc: {RIGHT_RPCURL}")


    args = parse_arguments()

    OWNER = account.address
    VERIFY = args.verify

    VALIDATORS = tuple(map(normal_addr, VALIDATORS))

    validator_left_deploy_tx = deploy_contract(
        web3=web3_left,
        args=(VALIDATORS, THRESHOLD, OWNER),
```

```
        verify=VERIFY,
        **CONTRACT_VALIDATORS,
    )
    VALIDATOR_LEFT_ADDRESS = validator_left_deploy_tx['contractAddress']
    print(f"#1 [LEFT] Validators Set deployed at {VALIDATOR_LEFT_ADDRESS}")

    left_bridge_deploy_tx = deploy_contract(
        web3=web3_left,
        args=(VALIDATOR_LEFT_ADDRESS, OWNER,),
        verify=VERIFY,
        **CONTRACT_LEFT
    )

    LEFT_BRIDGE_ADDRESS = left_bridge_deploy_tx['contractAddress']
    print(f"#2 [LEFT] Bridge deployed at {LEFT_BRIDGE_ADDRESS}")

    validator_right_deploy_tx = deploy_contract(
        web3=web3_right,
        args=(VALIDATORS, THRESHOLD, OWNER),
        verify=VERIFY,
        **CONTRACT_VALIDATORS,
    )
    VALIDATOR_RIGHT_ADDRESS = validator_right_deploy_tx['contractAddress']
    print(f"#3 [RIGHT] Validators Set deployed at {VALIDATOR_RIGHT_ADDRESS}")

    right_bridge_deploy_tx = deploy_contract(
        web3=web3_right,
        args=(VALIDATOR_RIGHT_ADDRESS, OWNER),
        verify=VERIFY,
        **CONTRACT_RIGHT
    )
    RIGHT_BRIDGE_ADDRESS = right_bridge_deploy_tx['contractAddress']
    print(f"#4 [RIGHT] Bridge deployed at {RIGHT_BRIDGE_ADDRESS}")

    LEFT_BLOCK = left_bridge_deploy_tx['blockNumber']
    print(f"#5 [LEFT] Bridge deployed at block {LEFT_BLOCK}")

    RIGHT_BLOCK = right_bridge_deploy_tx['blockNumber']
    print(f"#6 [RIGHT] Bridge deployed at block {RIGHT_BLOCK}")
```

deployment/flatten.sh

```
if [ -d flats ]; then
  rm -rf flats
fi

FLATTENER=solu
VERSION=0.7.6

${FLATTENER} contracts/LeftBridge.sol ${VERSION} -o LeftBridge_flat.sol
${FLATTENER} contracts/RightBridge.sol ${VERSION} -o RightBridge_flat.sol
${FLATTENER} contracts/BridgeValidators.sol ${VERSION} -o BridgeValidators_flat.sol

mv out flats
```

deployment/run.sh

```
#!/bin/sh
# change working directory
cd ${WORKDIR}/deployment

# flat the contracts
./flatten.sh > /dev/null

if [ "$VERIFY" = "true" ];
then
    python deploy.py --verify #2> /dev/null
else
    python deploy.py
fi
```

docker-compose.yml

```
version: "3.9"

services:
  oracle:
    image: n2n-oracle
    command: sh /wait-for.sh redis:5432 -- python oracle.py
    working_dir: /oracle
    env_file: .env

  redis:
    image: redis:4.0.5-alpine
    command: ["redis-server", "--appendonly", "yes", "--port", "5432"]
    hostname: redis
    volumes:
        - ${ORACLE_DATA}/redis:/data
```

Dockerfile

```
FROM python:3.9-alpine

COPY --from=ethereum/solc:0.7.6-alpine  /usr/local/bin/solc /usr/local/bin/solc

RUN apk update && apk upgrade
# Iptables.
RUN apk add iptables
# GCC.
RUN apk add --no-cache --virtual .build-deps gcc musl-dev

RUN python -m pip install --upgrade pip
COPY requirements.txt .
RUN python -m pip install -r requirements.txt

# Remove gcc.
RUN apk del .build-deps
# Remove cache.
RUN python -m pip cache purge

# workdir in root directory
ENV WORKDIR "/"
```

```
ENV PYTHONPATH "${PYTHONPATH}:/"

COPY deployment/ deployment/
WORKDIR /deployment/
RUN /bin/sh ./flatten.sh

WORKDIR /
COPY . .
```

oracle/__init__.py

oracle/base_oracle.py

```
from blockchain import web3_left, web3_right
from config import LEFT_ADDRESS, RIGHT_ADDRESS, \
    RIGHT_START_BLOCK, LEFT_START_BLOCK
from contract_abi import RIGHT_ABI, LEFT_ABI




def init():
    left_contract = web3_left.eth.contract(address=LEFT_ADDRESS, abi=LEFT_ABI)
    right_contract = web3_right.eth.contract(address=RIGHT_ADDRESS, abi=RIGHT_ABI)
    left_STOPBLOCK = LEFT_START_BLOCK
    right_STOPBLOCK = RIGHT_START_BLOCK

    try:
        pass #read from database
    except Exception:
        pass
    left_filter =
    ↪  left_contract.events.bridgeActionInitiated.createFilter(fromBlock=left_STOPBLOCK,
    ↪  toBlock='latest')
    right_filter =
    ↪  right_contract.events.bridgeActionInitiated.createFilter(fromBlock=right_STOPBLOCK,
    ↪  toBlock='latest')
    return [left_filter, right_filter]


def output_compose(eventlist, side):
    data = []
    for txn in eventlist:
        recipient, txn_hash = None, None
        try:
            recipient, txn_hash = txn['args']['recipient'], txn['transactionHash']

        except Exception:
            pass
        data.append({'recipient': recipient, 'txn_hash': txn_hash, "side": side})
    return data


def reed_blockchain():
    left_filter, right_filter = init()
```

```
    final_data = []
    left_eventlist = left_filter.get_all_entries()
    right_eventlist = right_filter.get_all_entries()
    if len(left_eventlist):
        final_data += output_compose(left_eventlist, 'left')
    if len(right_eventlist):
        final_data += output_compose(right_eventlist, 'right')

    return final_data
```

oracle/constants.py

```
TO_BUILD_QUEUE = 'to_build'
TO_SIGN_QUEUE = 'to_sign'
```

oracle/contract_abi.py

```
from config import CONTRACT_LEFT, CONTRACT_RIGHT
from blockchain import compile_contract

CONTRACT_RIGHT['path'] = '../deployment/' + CONTRACT_RIGHT['path']
CONTRACT_LEFT['path'] = '../deployment/' + CONTRACT_LEFT['path']

_, RIGHT_ABI = compile_contract(CONTRACT_RIGHT['path'], CONTRACT_RIGHT['name'])
_, LEFT_ABI = compile_contract(CONTRACT_LEFT['path'], CONTRACT_LEFT['name'])
```

oracle/contract_functions.py

```
from typing import Tuple
import logging
from blockchain import send_tx

def call_function(contract, function_name, *args):
    logging.info(f"Call {function_name}{args}")
    return contract.functions[function_name](*args).call()

def isValidator(contract, address):
    """
    Returns true if address is validator in left of right side
    """
    return call_function(contract, 'isValidator', address)


def isRobustMode(contract) -> bool:
    return call_function(contract, 'isRobustMode')


def getTransferDetails(contract, txHash: bytes) -> Tuple[str, int]:
    """
    Return recipient and amount
    """
    return call_function(contract, 'getTransferDetails', txHash)

def getCommit(contract, txHash: bytes, index: int) -> Tuple[int, int, int]:
    """
```

```
    Return r, s, v
    """
    return call_function(contract, 'getCommit', txHash, index)


def custom_getCommitsAmount(contract, txHash):
    if txHash.startswith('0x'):
        txHash = txHash[2:]
    events = contract.events.commitsCollected().createFilter(
        fromBlock=0,
        argument_filters={
            'id': bytes.fromhex(txHash)
        }).get_all_entries()
    assert events, "Not such txHash"

    event = events[0]

    return event['args']['commits']


def transact_contract_function(web3, contract, account, function_name, *args):
    function = contract.functions[function_name](*args)

    tx = function.buildTransaction()

    return send_tx(web3, tx, account=account)

def applyCommits(web3, contract, account, recipient, amount, id, r, s, v):
    return transact_contract_function(web3, contract, account, 'applyCommits',
    ↪  recipient, amount, id, r, s, v)
```

oracle/oracle.py

```
from signer_worker import OracleManager
from config import RIGHT_RPCURL, RIGHT_ADDRESS , RIGHT_GASPRICE
from blockchain import web3_right, account, normal_addr, LEFT_CONNECTED,
↪  RIGHT_CONNECTED
from contract_abi import RIGHT_ABI
import time
import logging

import random
import os

if not LEFT_CONNECTED or not RIGHT_CONNECTED:
    print(f'ERROR: NOT CONNECTED TO RPC: {LEFT_CONNECTED, RIGHT_CONNECTED}')
    exit()

logging.basicConfig(level=logging.DEBUG)

with OracleManager(builders=2) as manager:
    manager.join()
```

oracle/signer_worker.py

```
from multiprocessing import Value, cpu_count
from queue import Queue
from threading import Thread
from contract_functions import isRobustMode
from contract_abi import LEFT_ABI, RIGHT_ABI
from workers import (
    event_watcher,
    tx_builder_worker,
    tx_sender_worker,
    receipt_checker_worker,
)

from config import (
    LEFT_RPCURL, LEFT_ADDRESS, LEFT_START_BLOCK, LEFT_GASPRICE,
    RIGHT_RPCURL, RIGHT_ADDRESS, RIGHT_START_BLOCK, RIGHT_GASPRICE,
)
from utils import set_left_block, set_right_block, get_right_block, get_left_block


def get_block(side):
    if side == 'left':
        database_block = get_left_block()
        return max(database_block, LEFT_START_BLOCK)
    else:
        database_block = get_right_block()
        return max(database_block, RIGHT_START_BLOCK)


class OracleManager:
    def __init__(self, builders=cpu_count()):
        self.web3_url_left = LEFT_RPCURL
        self.contract_address_left = LEFT_ADDRESS
        self.abi_left = LEFT_ABI

        self.web3_url_right = RIGHT_RPCURL
        self.contract_address_right = RIGHT_ADDRESS
        self.abi_right = RIGHT_ABI

        self.gasPrice_left = LEFT_GASPRICE
        self.gasPrice_right = RIGHT_GASPRICE
        self.start_block_left = get_block('left')
        self.start_block_right = get_block('right')

        self.set_left_block = set_left_block

        self.set_right_block = set_right_block

        # Loop flag.
        self.__is_sender_running = Value('i', True)
        self.__is_builders_running = Value('i', True)
        self.__is_watcher_running = Value('i', True)
        self.__is_receipt_checker_running = Value('i', True)

        # Queues.
        self.builders_input = Queue()

        self.sender_left_input = Queue()
        self.sender_right_input = Queue()
```

```
self.sender_left_output = Queue()
self.sender_right_output = Queue()

# Create left watcher
watcher_left_args = (
    self.web3_url_left,
    self.web3_url_right,
    self.contract_address_left,
    self.abi_left,
    self.contract_address_right,
    self.abi_right,
    self.__is_watcher_running,
    self.builders_input,
    self.start_block_left,
    self.set_left_block,
    'left'
)
self.__watcher_left = Thread(target=event_watcher, args=watcher_left_args,
↪  name='watcher-left')

# Create right watcher
watcher_right_args = (
    self.web3_url_right,
    self.web3_url_left,
    self.contract_address_left,
    self.abi_left,
    self.contract_address_right,
    self.abi_right,
    self.__is_watcher_running,
    self.builders_input,
    self.start_block_right,
    self.set_right_block,
    'right'
)
self.__watcher_right = Thread(target=event_watcher, args=watcher_right_args,
↪  name='watcher-right')

# Builders.
builders_args = (
    self.__is_builders_running,
    self.builders_input,
    self.sender_left_input,
    self.sender_right_input,
)

self.__builders = [
        Thread(target=tx_builder_worker, args=builders_args,
        ↪  name=f'builder-{i}')
        for i in range(builders)
]

# left sender
sender_left_args = (
    self.web3_url_left,
    self.gasPrice_left,
    self.__is_sender_running,
    self.sender_left_input,
    self.sender_left_output,
)
```

```
        self.__sender_left = Thread(target=tx_sender_worker, args=sender_left_args,
        ↪  name='sender-left')

        # right sender
        sender_right_args = (
            self.web3_url_right,
            self.gasPrice_right,
            self.__is_sender_running,
            self.sender_right_input,
            self.sender_right_output,
        )

        self.__sender_right = Thread(target=tx_sender_worker, args=sender_right_args,
        ↪  name='sender-right')


        # Receipt checker.
        receipt_checker_left_args = (
            self.web3_url_left,
            self.__is_receipt_checker_running,
            self.sender_left_output,
            self.sender_left_input,
        )

        self.__receipt_checker_left = Thread(target=receipt_checker_worker,
        ↪  args=receipt_checker_left_args,
                                             name='receipt_checker_left')

        # right Receipt checker.
        receipt_checker_right_args = (
            self.web3_url_right,
            self.__is_receipt_checker_running,
            self.sender_right_output,
            self.sender_right_input,
        )

        self.__receipt_checker_right = Thread(target=receipt_checker_worker,
        ↪  args=receipt_checker_right_args,
                                             name='receipt_checker_right')

    def start(self) -> None:
        self.__watcher_left.start()
        self.__watcher_right.start()

        self.__sender_left.start()
        self.__sender_right.start()


        self.__receipt_checker_left.start()
        self.__receipt_checker_right.start()

        for builder in self.__builders:
            builder.start()

    def stop(self) -> None:
        with self.__is_sender_running.get_lock():
            self.__is_sender_running.value = False

        with self.__is_builders_running.get_lock():
```

```
            self.__is_builders_running.value = False

        with self.__is_watcher_running.get_lock():
            self.__is_watcher_running.value = False

        with self.__is_receipt_checker_running.get_lock():
            self.__is_receipt_checker_running.value = False

    def join(self) -> None:
        for builder in self.__builders:
            builder.join()

        self.__watcher_left.join()
        self.__watcher_right.join()

        self.__sender_left.join()
        self.__sender_right.join()


        self.__receipt_checker_left.join()
        self.__receipt_checker_right.join()


    def __enter__(self):
        self.start()

        return self

    def __exit__(self, exc_type, exc_value, traceback) -> None:
        self.stop()
        self.join()
```

oracle/test.py

```
import json

from web3 import Web3, HTTPProvider
from web3.eth import Account

'''
addd upd liquidity
enable robust left right
commit roight 1 eth
'''


# Sends transaction. If `function` in kwargs, sends contract call.
# If `receipt` is `True` - returns receipt, else - transaction hash.
def send_transaction(web3: Web3, account: Account, receipt=True, revert=False, \
                     **kwargs) -> dict:
    tx = {
        'from': account.address,
        'nonce': web3.eth.getTransactionCount(account.address),

        'gasPrice': 20000000000
    }

    # Custom parameters.
    tx.update(kwargs)
```

```
    try:
        # Contract call.
        if 'function' in tx:
            # Geth.
            gas_price = tx.pop('gasPrice')

            tx = tx.pop('function').buildTransaction(tx)

            tx['gasPrice'] = gas_price

        else:
            tx['gas'] = web3.eth.estimateGas(tx)

    except Exception as exc:
        # Should revert.
        if revert:
            return None

        raise exc

    # If not reverted.
    assert not revert, \
        f'Transaction should be reverted:\n{json.dumps(tx, indent=4,
        ↪  sort_keys=True)}.'

    tx_hash = web3.eth.sendRawTransaction(account.sign_transaction(tx).rawTransaction)

    if receipt:
        return web3.eth.waitForTransactionReceipt(tx_hash)

    return tx_hash


# Sends transaction and checks that status == `status`,
# If `receipt` is `True` - returns receipt, else - transaction hash.
def send_transaction_and_check(web3: Web3, account: Account, status=1, receipt=True, \
                               revert=False, **kwargs) -> dict:
    try:
        tx_receipt = send_transaction(web3, account, receipt, revert, **kwargs)

    except Exception as exc:
        raise Exception(f'Cannot execute transaction: {exc}')

    if revert:
        assert tx_receipt is None, \
            'Transaction should be reverted: {tx_reseipt}.'

        return None

    if receipt:
        # Checks only of we wait receipt.
        assert int(tx_receipt.status) == status, \
            f'Transaction status mismatch: {tx_receipt.status}'

    return tx_receipt


def setLiquidity(bridge_left, bridge_right, owner_account, valueInWei):
    """
```

```
    Set the liquidity on `bridge_left` and `bridge_bridge`
    to `value` (in ether) using `owner_account`
    """
    web3_left = bridge_left.web3
    web3_right = bridge_right.web3

    send_transaction_and_check(web3_right, owner_account,
                               function=bridge_right.functions.addLiquidity(),
                               value=valueInWei)

    assert bridge_right.functions.getLiquidityLimit().call() >= valueInWei

    send_transaction_and_check(web3_left, owner_account,

                               ↪  function=bridge_left.functions.updateLiquidityLimit(valueInWei),
                               value=0)

    assert bridge_left.functions.getLiquidityLimit().call() >= valueInWei


url = "http://sokol.poa.network"
from config import PRIVKEY as priv_key
from json import loads
from contract_abi import LEFT_ABI, RIGHT_ABI

web3 = Web3(HTTPProvider(url))
acc = web3.eth.account.from_key(priv_key)

left = web3.eth.contract(address='0xeaBd5e26e8a2C97b090680B2d2dbd033ADf17629',
↪  abi=LEFT_ABI)
right = web3.eth.contract(address='0x9391AE4CfcC4Fb5B27C3b816D633a05Ff12935a7',
↪  abi=RIGHT_ABI)

print('setLiquidity')
setLiquidity(left, right, acc, web3.toWei('10', 'ether'))

print('enable')
send_transaction_and_check(web3, acc,
                           function=left.functions.enableRobustMode(),
                           value=0)
send_transaction_and_check(web3, acc,
                           function=right.functions.enableRobustMode(),
                           value=0)

print('sent to left')
send_transaction(web3, acc, to=left.address, value=web3.toWei('0.1', 'ether'))
import time
print('sleep 10 sec')
time.sleep(10)
print('send to right')
send_transaction(web3, acc, to=right.address, value=web3.toWei('0.1', 'ether'))
```

oracle/utils/__init__.py

```
from .database import set_left_block, set_right_block, get_left_block, get_right_block
```

oracle/utils/database.py

```
from redis import Redis
from config import REDIS_CONFIG
from typing import Union


redis_db = Redis(**REDIS_CONFIG)
redis_db.ping()

LEFT_BLOCK_NUMBER = 'oracle::left_block_number'
RIGHT_BLOCK_NUMBER = 'oracle::right_block_number'

def _set(key: Union[int, str, bytes], value: Union[int, str, bytes]):
    redis_db.set(key, value)

def _get(key: Union[int, str, bytes]) -> bytes:
    return redis_db.get(key)

def get_left_block() -> int:
    """
    Restore block number value from database
    """
    return int(_get(LEFT_BLOCK_NUMBER) or 0)

def get_right_block() -> int:
    """
    Restore block number value from database
    """
    return int(_get(RIGHT_BLOCK_NUMBER) or 0)


def set_left_block(block_numer):
    """
    Set block_number to database
    """
    _set(LEFT_BLOCK_NUMBER, block_numer)


def set_right_block(block_numer):
    """
    Set block_number to database
    """
    _set(RIGHT_BLOCK_NUMBER, block_numer)
```

oracle/utils/exceptions.py

```
class AccountNotValidator(Exception):
    def __init__(self, address, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.address = address

class EventWasAlreadySent(Exception):
    pass
```

oracle/workers/__init__.py

```
from .builder import tx_builder_worker
from .sender import tx_sender_worker
from .watcher import event_watcher
from .receipt_checker import receipt_checker_worker
```

oracle/workers/builder.py

```
import time
from multiprocessing import Value
from queue import Empty as EmptyException

from eth_typing import Address

from blockchain import get_prepared_web3, get_deployed_contract
from .constants import builder_logger, Queue
from contract_functions import isValidator
from utils.exceptions import AccountNotValidator, EventWasAlreadySent
from web3 import Web3


def buildCommitTransaction(function, args):
    """
    Check validator account and build the transaction
    """
    web3: Web3 = function.web3
    contract_address = function.address
    contract_abi = function.contract_abi

    contract = get_deployed_contract(web3, contract_address, contract_abi)

    validator_address = web3.eth.defaultAccount
    if not isValidator(contract, validator_address):
        raise AccountNotValidator(validator_address)

    try:
        tx = function(*args).buildTransaction()
    except ValueError as e:
        builder_logger.info(f"{args}")
        raise EventWasAlreadySent(*e.args)
    tx['gas'] *= 4
    return tx


def tx_builder_worker(is_running: Value, input_queue: Queue, left_output_queue:
↪  Queue,\
     right_output_queue: Queue):
    # Processing.
    while is_running.value:
        try:
            job = input_queue.get_nowait()
            builder_logger.info(f"Got job {job}")
            side, function, args = job['side'], job['function'], job['args']

        # To prevend blocking when is_running is False.
        except EmptyException:
            # Wait 100 ms.
```

```
            time.sleep(0.1)
            # Again.
            continue

        try:
            # Build transaction.
            tx = buildCommitTransaction(function, args)
        except AccountNotValidator as e:
            builder_logger.error(f'Account is not validator! {e.address}')
            # put job back
            input_queue.put(job)
            time.sleep(1)
            continue

        except EventWasAlreadySent as e:
            builder_logger.error(f"Transaction was already sent")
            continue

        builder_logger.info(f'Put builded transaction {tx}')

        if side == 'left':
            left_output_queue.put({'tx': tx})

        if side == 'right':
            right_output_queue.put({'tx': tx})
```

oracle/workers/constants.py

```
import logging

from queue import Queue

LOG_FORMAT = ('%(levelname) -10s %(asctime)s %(name) -15s %(funcName) -20s:
↪  %(message)s')
logging.basicConfig(format=LOG_FORMAT)

level = logging.INFO

def getLogger(name):
    logger =  logging.getLogger(name)
    logger.setLevel(level)
    return logger

watcher_logger = logging.getLogger('watcher')
watcher_logger.setLevel(level)

builder_logger = logging.getLogger('builder')
builder_logger.setLevel(level)

sender_logger = logging.getLogger('sender')
sender_logger.setLevel(level)

receipt_checker_logger = logging.getLogger('receipt_checker')
receipt_checker_logger.setLevel(level)
```

oracle/workers/receipt_checker.py

```
from multiprocessing import Value
from time import sleep, time

from web3.exceptions import TransactionNotFound

from blockchain import get_prepared_web3
from config import TIMEOUT_RECEIPT_CHECKER as TIMEOUT
from .constants import Queue, receipt_checker_logger as logger

def receipt_checker_worker(rpc_url: str, is_running: Value, input_queue: Queue,
↪  sender_queue: Queue) -> None:
    """
    Watches for event forever
    """
    web3 = get_prepared_web3(rpc_url)

    while is_running.value:

        if not input_queue.empty():
            job = input_queue.get()
            tx, tx_hash, tx_time = job
            try:
                tx_from_node = web3.eth.getTransaction(tx_hash)
                logger.info(f"check transaction for block:
                ↪  {tx_from_node['blockNumber']}")
                if not tx_from_node['blockNumber']:
                    if time() - tx_time >= TIMEOUT:
                        logger.info('Put transaction back to sender input')
                        sender_queue.put(tx)
                    else:
                        input_queue.put(job)
                else:
                    txReceipt = web3.eth.getTransactionReceipt(tx_hash)
                    logger.info(f"Transaction is done, status: {txReceipt['status']}")
            except TransactionNotFound:
                logger.info('Transaction not found. Put back to sender input')
                sender_queue.put({'tx' : tx})
        else:
            sleep(0.1)
```

oracle/workers/sender.py

```
from queue import Empty as EmptyException
from multiprocessing import Value
from .constants import sender_logger as logger, Queue
from blockchain import get_prepared_web3, account

import time
class LowBalance(Exception):
    pass

def tx_sender_worker(rpc_url: str, gasPrice: int, is_running: Value,
                     input_queue: Queue, output_queue: Queue):
    web3 = get_prepared_web3(rpc_url)
    nonce = web3.eth.getTransactionCount(account.address)
    while is_running.value:
```

```
        try:
            job = input_queue.get_nowait()
            tx = job['tx']
            logger.info(f'Got transaction. {tx}')

        # To prevend blocking when is_running is False.
        except EmptyException:
            # Wait 100 ms.
            time.sleep(0.1)

            # Again.
            continue

        try:
            # Success.
            tx['gasPrice'] = gasPrice
            while True:
                try:
                    logger.info(f"Send transaction with nonce={nonce}")
                    tx['nonce'] = nonce
                    raw_signed = account.sign_transaction(tx).rawTransaction
                    txHash = web3.eth.sendRawTransaction(raw_signed)
                except ValueError as e:
                    if web3.eth.getBalance(account.address) < tx['gas'] *
                    ↪  tx['gasPrice']:
                        raise LowBalance
                    else:
                        nonce = web3.eth.getTransactionCount(account.address)
                        continue
                break

            current_time = time.time()
            logger.info(f"Sent at {txHash}")
            output_queue.put((tx, txHash, current_time))
            nonce += 1

        except Exception as e:
            logger.error('Low balance!')
            input_queue.put(job)
            time.sleep(1)
            continue
```

oracle/workers/watcher.py

```
from multiprocessing import Value
from time import sleep

from eth_typing import Address
from contract_functions import isRobustMode
from blockchain import get_prepared_web3, account
from .constants import getLogger, Queue
from crypto import prepareSignature
from config import REQUIRED_CONFIRMATIONS


def event_watcher(rpc_url_home: str, rpc_url_foreign: str, contract_address_left:
↪  Address, abi_left: dict,
                  contract_address_right: Address, abi_right: dict, is_running: Value,
```

```
                output_queue: Queue,
                start_block: int, write_to_bd: callable, side: str) -> None:
    """
    Watches for event forever
    """
    logger = getLogger(f"{rpc_url_home}-wathcer")

    web3_home = get_prepared_web3(rpc_url_home)
    web3_foreign = get_prepared_web3(rpc_url_foreign)

    if side == 'right':
        contract_left = web3_foreign.eth.contract(address=contract_address_left,
        ↪  abi=abi_left)
        contract_right = web3_home.eth.contract(address=contract_address_right,
        ↪  abi=abi_right)
        contract = contract_right
    else:
        contract_left = web3_home.eth.contract(address=contract_address_left,
        ↪  abi=abi_left)
        contract_right = web3_foreign.eth.contract(address=contract_address_right,
        ↪  abi=abi_right)
        contract = contract_left
    latest_block = web3_home.eth.blockNumber

    logger.info(f"{REQUIRED_CONFIRMATIONS} -- REQUIRED_CONFIRMATIONS")
    logger.info(f"{start_block} -> {latest_block}")

    while is_running.value:
        latest_block = web3_home.eth.blockNumber
        while start_block > latest_block:
            latest_block = web3_home.eth.blockNumber
            sleep(0.1)
        if side == 'right' and isRobustMode(contract_left):
            filter_ =
            ↪  contract.events.commitsCollected.createFilter(fromBlock=start_block,
            ↪  toBlock=latest_block)
            events = filter_.get_all_entries()
            if events:
                logger.info(f'Found {len(events)} event(s) commits collected')

            for event in events:
                function = contract_left.functions.applyCommits
                id_, commits = '0x' + event['args']['id'].hex(),
                ↪  event['args']['commits']
                logger.info(f'{id_} {commits}: commits collected')
                try:
                    oracle = contract_right.functions.getAssignee(id_).call()
                except Exception as e:
                    logger.error(f"{contract_right.functions.getAssignee(id_)},
                    ↪  {contract_right.address}")
                    raise e

                if account.address != oracle:
                    logger.info(f'Needed oracle: {oracle}, I\'am: {account.address}')
                    continue
                recipient, amount =
                ↪  contract_right.functions.getTransferDetails(id_).call()
                r = []
                s = []
                v = []
```

```
                for i in range(commits):
                    r_, s_, v_ = contract_right.functions.getCommit(id_, i).call()
                    r.append(r_)
                    s.append(s_)
                    v.append(v_)
                args = recipient, amount, id_, r, s, v
                logger.info(f'Args of event: {args}: commits collected')
                dict_for_queue = {'args': args, 'side': 'left', 'function': function}
                output_queue.put(dict_for_queue)
                logger.info(f"Found event {event}.")

        logger.info(f'Found block new block: {latest_block}')
        to_check_block = latest_block - REQUIRED_CONFIRMATIONS
        if to_check_block < start_block:
            continue
        logger.info(f"{start_block} -> {to_check_block}")
        filter_ =
        ↪  contract.events.bridgeActionInitiated.createFilter(fromBlock=start_block,
        ↪  toBlock=to_check_block)
        events = filter_.get_all_entries()
        if events:
            logger.info(f'Found {len(events)} event(s)')

        for event in events:
            if side == 'right' and isRobustMode(contract_left):
                function = contract_right.functions.registerCommit
                args = event['args']['recipient'], event['args']['amount'],
                ↪  event['transactionHash'], *prepareSignature(
                    account, event['args']['recipient'], event['args']['amount'],
                    ↪  event['transactionHash'])
                dict_for_queue = {'args': args, 'side': 'right', 'function': function}
            else:
                if side == 'right':
                    function = contract_left.functions.commit
                    side_ = 'left'
                else:
                    function = contract_right.functions.commit
                    side_ = 'right'
                args = event['args']['recipient'], event['args']['amount'],
                ↪  event['transactionHash']
                dict_for_queue = {'args': args, 'side': side_, 'function': function}
            output_queue.put(dict_for_queue)
            logger.info(f"Found event {event}.")


        write_to_bd(to_check_block + 1)
        start_block = to_check_block + 1


'''
    filter_ =
↪  contract.events.bridgeActionInitiated.createFilter(fromBlock=start_block,
↪  toBlock=latest_block)
    events = filter_.get_all_entries()
    if events:
        logger.info(f'Found {len(events)} event(s)')

    for event in events:
        args = event['args']['id'], event['args']['commits']
```

```
        output_queue.put(
            {'args': args}
        )
        logger.info(f"Found event {event}.")
    write_to_bd(latest_block + 1)
    start_block = latest_block + 1
'''
```

requirements.txt

```
attrs==20.3.0
base58==2.1.0
bitarray==1.2.2
certifi==2020.12.5
chardet==4.0.0
cytoolz==0.11.0
eth-abi==2.1.1
eth-account==0.5.4
eth-hash==0.3.1
eth-keyfile==0.5.1
eth-keys==0.3.3
eth-rlp==0.2.1
eth-typing==2.2.2
eth-utils==1.10.0
hexbytes==0.2.1
idna==2.10
ipfshttpclient==0.7.0a1
jsonschema==3.2.0
lru-dict==1.1.7
multiaddr==0.0.9
netaddr==0.8.0
parsimonious==0.8.1
protobuf==3.15.6
pycryptodome==3.10.1
pyrsistent==0.17.3
requests==2.25.1
rlp==2.0.1
six==1.15.0
toolz==0.11.1
urllib3==1.26.4
varint==1.0.2
web3==5.17.0
websockets==8.1
python-dotenv==0.15.0
py-solc-x==1.1.0
solidity-unfolder==1.0.1
redis==3.5.3
```

tools/applyCommits.py

```
#!/usr/local/bin/python
import sys
from blockchain import account, web3_left, web3_right, get_deployed_contract,
↪  ZERO_ADDRESS
from config import LEFT_ADDRESS, RIGHT_ADDRESS
from oracle.contract_abi import LEFT_ABI, RIGHT_ABI
from oracle.contract_functions import (
```

```
    isRobustMode, getTransferDetails,
    getCommit, custom_getCommitsAmount,
    applyCommits
    )
import logging

if __name__ == '__main__':
    logging.basicConfig(level=logging.INFO)

    contract_left = get_deployed_contract(web3_left, LEFT_ADDRESS, LEFT_ABI)
    contract_right = get_deployed_contract(web3_right, RIGHT_ADDRESS, RIGHT_ABI)

    assert isRobustMode(contract_left), "Robust mode on left is off"
    assert isRobustMode(contract_right), "Robust mode on right is off"

    txHash = sys.argv[1]

    receiver, amount = getTransferDetails(contract_right, txHash)
    logging.info(f"receiver, amount: {receiver, amount}")
    assert receiver != ZERO_ADDRESS and amount != 0, f"There is no such action with
    ↪  id={txHash}"

    commits = custom_getCommitsAmount(contract_right, txHash)
    logging.info(f"Found {commits} commits")
    rs, ss, vs = [], [], []

    for index in range(commits):
        r, s, v = getCommit(contract_right, txHash, index)
        logging.info(f"r, s, v: {r, s, v}")
        rs.append(r)
        ss.append(s)
        vs.append(v)

    tx = applyCommits(web3_left, contract_left, account, receiver, amount, txHash, rs,
    ↪  ss, vs)
    print(f"{tx['transactionHash'].hex()} executed")
```

utils.py

```
def exit_with_message(message, exit_code=44, done=True):
    message = message.strip()
    if not done:
        message += ' Nothing to do.'
    print(message)
    exit(exit_code)
```

wait-for.sh

```
#!/bin/sh

# The MIT License (MIT)
#
# Copyright (c) 2017 Eficode Oy
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
```

```
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.

set -- "$@" -- "$TIMEOUT" "$QUIET" "$HOST" "$PORT" "$result"
TIMEOUT=15
QUIET=0

echoerr() {
  if [ "$QUIET" -ne 1 ]; then printf "%s\n" "$*" 1>&2; fi
}

usage() {
  exitcode="$1"
  cat << USAGE >&2
Usage:
  $cmdname host:port [-t timeout] [-- command args]
  -q | --quiet                        Do not output any status messages
  -t TIMEOUT | --timeout=timeout      Timeout in seconds, zero for no timeout
  -- COMMAND ARGS                     Execute command with args after the test
  ↪  finishes
USAGE
  exit "$exitcode"
}

wait_for() {
 if ! command -v nc >/dev/null; then
    echoerr 'nc command is missing!'
    exit 1
  fi

  while :; do
    nc -z "$HOST" "$PORT" > /dev/null 2>&1

    result=$?
    if [ $result -eq 0 ] ; then
      if [ $# -gt 6 ] ; then
        for result in $(seq $(($# - 6))); do
          result=$1
          shift
          set -- "$@" "$result"
        done

        TIMEOUT=$2 QUIET=$3 HOST=$4 PORT=$5 result=$6
        shift 6
        exec "$@"
      fi
      exit 0
    fi
```

```
    if [ "$TIMEOUT" -le 0 ]; then
      break
    fi
    TIMEOUT=$((TIMEOUT - 1))

    sleep 1
  done
  echo "Operation timed out" >&2
  exit 1
}

while :; do
  case "$1" in
    *:* )
    HOST=$(printf "%s\n" "$1"| cut -d : -f 1)
    PORT=$(printf "%s\n" "$1"| cut -d : -f 2)
    shift 1
    ;;
    -q | --quiet)
    QUIET=1
    shift 1
    ;;
    -q-*)
    QUIET=0
    echoerr "Unknown option: $1"
    usage 1
    ;;
    -q*)
    QUIET=1
    result=$1
    shift 1
    set -- -"${result#-q}" "$@"
    ;;
    -t | --timeout)
    TIMEOUT="$2"
    shift 2
    ;;
    -t*)
    TIMEOUT="${1#-t}"
    shift 1
    ;;
    --timeout=*)
    TIMEOUT="${1#*=}"
    shift 1
    ;;
    --)
    shift
    break
    ;;
    --help)
    usage 0
    ;;
    -*)
    QUIET=0
    echoerr "Unknown option: $1"
    usage 1
    ;;
    *)
    QUIET=0
```

```
    echoerr "Unknown argument: $1"
    usage 1
    ;;
  esac
done

if ! [ "$TIMEOUT" -ge 0 ] 2>/dev/null; then
  echoerr "Error: invalid timeout '$TIMEOUT'"
  usage 3
fi

if [ "$HOST" = "" -o "$PORT" = "" ]; then
  echoerr "Error: you need to provide a host and port to test."
  usage 2
fi

wait_for "$@"
```