Трёхсторонние контракты в банковской сфере

Sergey Prilutskiy, Chief of Research @ MixBytes.io & Smartz.io

Трёхсторонние контракты в банковской сфере

- План доклада:
  - схема публикации смарт-контрактов в современных блокчейнах
  - долгосрочные и краткосрочные смарт-контракты
  - трёхсторонние контракты, контракт "multisig"
  - схемы использования трёхсторонних контрактов в банковской сфере

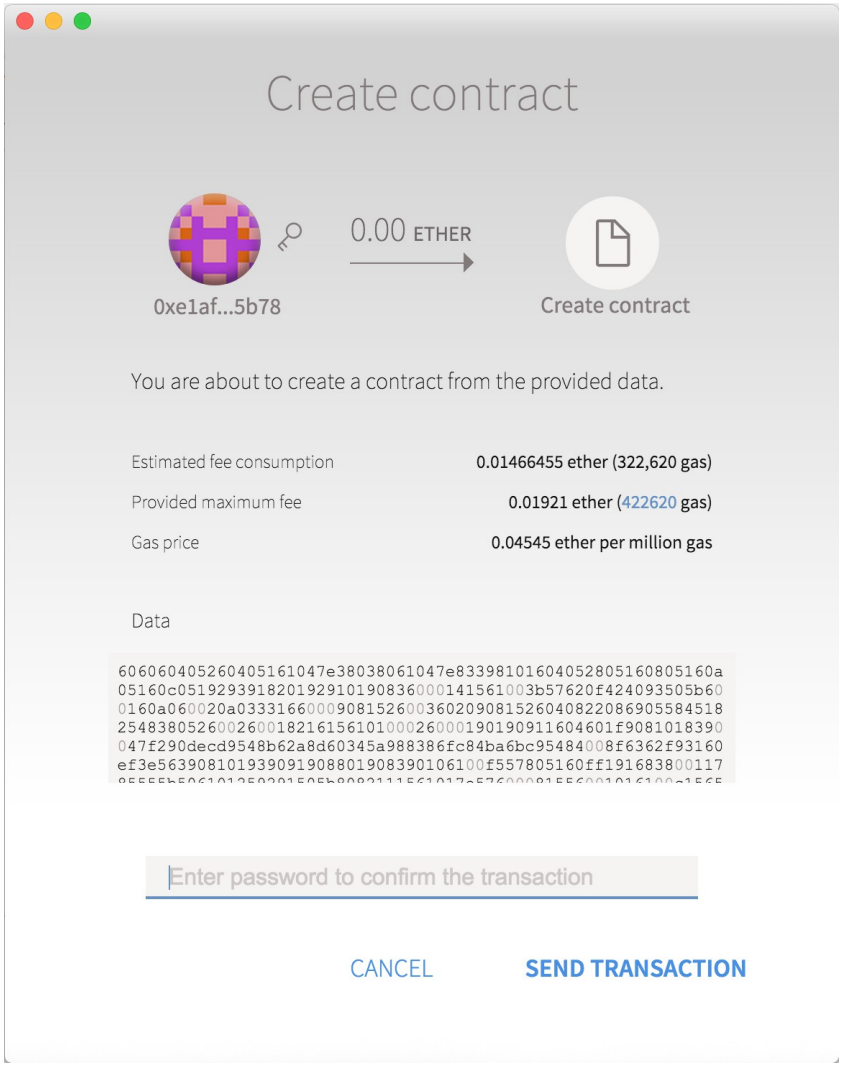
Общая схема работы со смарт-контрактами в современных сетях, tech view

- публикация контракта
  - клиент сети готовит специальную транзакцию типа „create_contract“
  - клиент помещает в неё код контракта, и параметры конструктора, с которыми контракт будет создан (например внутренний курс, время жизни контракта и т.п.)
  - клиент подписывает транзакцию (попросту добавляя к ней саму подпись), используя секретный ключ, хранящийся в клиентском софте (браузере, мобильном приложении, внешней программе)
  - клиент отправляет транзакцию на любую из публичных нод Ethereum
  - нода, получившая транзакцию проверяет её, и, рассылает её через p2p сеть соседним нодам, которые в свою очередь распространяют ее дальше по сети, пока она не достигнет block-producer-а
  - block-producer, также проверяет транзакцию, валидирует код контракта в ней и помещает код контракта в блок под некоторым адресом, таким же как у любого участника сети
  - block-producer „закрывает“ блок согласно алгоритму консенсуса, например посчитав "proof-of-work"
  - block-producer публикует созданный им блок в p2p сети, и, как и с транзакциями, новый блок тиражируется на все ноды сети
  - любая получившая новый блок нода процессит его, и, встретив код контракта запоминает по какому адресу в сети находится код контракта
  - „любая получившая новый блок нода“ теперь может „передавать“ транзакции, адресованные новому контракту прямо в его код

- вызов кода контракта (с отправкой средств)
  - клиент (точнее его нода) видит код нужного контракта по определённому адресу, и хочет отправить в него транзакцию
  - клиент создаёт транзакцию от своего адреса на адрес контракта, возможно, вкладывает в неё необходимое количество средств, а также другие нужные контракту параметры
  - клиент подписывает траназкцию, используя свой секретный ключ и отправляет транзакцию в p2p сеть (в любую ноду)
  - block-producer видит новую транзакцию, валидирует её, далее, действия block-producer-а можно по аналогии представить так:
    - block-producer смотрит какая фкнция вызывается транзакцией клиента, и вызывает эту функцию в контракте, передавая в контракт все данные транзакции
    - теперь контракт „знает“ кто ему прислал средства, и сколько. Контракт меняет своё внутреннее состояние (начисляет отправителю токены, или наоборот в ответ на запрос высылает средства куда либо).
    - множество block-producer-ов проделывают эту процедуру, получая строго одни и те же изменения в состоянии контракта.
  - block-producer, "исполнив" таким образом код контракта записывает в создаваемый блок транзакцию клиента и вызванные ей изменения (баланс контракта изменился, внутренний storage контракта тоже изменился)
  - изменения в блоке тиражируются между нодами согласно адгоритму консенсуса сети как во всех остальных случаях

Общая схема работы со смарт-контрактами в современных сетях, simplify view

- можно представить публикацию контракта как размещение сайта в сети Интернет с получением нового, уникального адреса
- этот „сайт“ обладает такими свойствами:
  - код этого „сайта“ не может быть изменён (на деле это можно делать при помощи контрактов-контроллеров)
  - каждый „запрос“ к такому сайту, который как то меняет его состояние – это неотвергаемая транзакция, навсегда записанная в блокчейн
  - каждый „запрос“ к такому сайту может нести в качестве нагрузки любое число средств
  - каждый „запрос“ к такому сайту виден всем участникам сети, в том числе и неудачные и ошибочные вызовы
- можно представить выложенный в сеть контракт, как автоматизированный „кошелёк“, в который можно послать средства, он как то внутри обрабатывает эту информацию, и умеет пересылать средства другим адресам
- чтобы обратиться к контракту в сети, надо знать его адрес и создать транзакцию с некоторым количеством средств, адресованную контракту



Долгосрочные и краткосрочные смарт-контракты

- долгосрочные контракты
  - содержат данные о большом количестве сущностей
  - долгое или неограниченное время жизни
  - один контракт на большое количество взаимодейтсвий с пользователями

- примеры долгосрочных контрактов
  - token
  - multisig
  - equity

- долгосрочные контракты выкладываются в сеть один раз, и затем используются долгое время
- адреса долгосрочных контрактов надолго сохраняются в различных источниках, например на криптобиржах

- краткосрочные контракты
  - содержат минимальное количество данных, пару конкретных чисел
  - имеют короткое время жизни (часы, дни)
  - один контракт на одну операцию, либо на одного пользователя

- примеры краткосрочных контрактов
  - invoice-paid
  - commit-reveal
  - one-time multisig

- краткосрочные смарт-контракты выкладываются в сеть по требованию, большим партиями, отражая операции со множеством объектов
- адреса краткосрочных смарт-контрактов живут недолго, в идеале вообще используются один раз, и, после проведения операции, контракт необходим только для получения информации из блокчейна

Multisig

- универсальный тип контрактов, позволящий хранение и вывод средств только по достижении кворума участников, требующий N из M (N <= M) электронных подписей участников для совершения действия (обычно – вывода средств)

- схема работы простейшего multisig 2/3 в сети Ethereum:
  - есть три участника, муж, жена и банк
  - муж создает multisig контракт:
    - прописывает в нем адреса себя, жены и банка
    - прописывает, что когда в контракт придёт запрос (от него, жены, или банка) на вывод средств, он должен перейти в состояние „ожидаю дополнительных подтверждений“
    - любые входящие платежи кладутся на баланс контракта
    - для того, чтобы запросить вывод жена отправляет в контракт запрос на вывод средств
    - контракт переходит в состояние „ожидаю 2 из 3 подтверждений, первая подпись уже есть (жены)“
    - муж потерял телефон и не может потвердить вывод
    - муж идёт в банк, подтверждает свою личность, и банк отправляет подтверждение на вывод в контракт
    - в тот момент, когда в контракт придёт вторая транзакция (а он требует 2 из 3 подписей), он высылает средства на указанный женой адрес
    - если в течение определенного времени кворум не достигнут, попытка вывода считаеется несостоявшейся
- таким образом взломанный компьютер мужа, телефон жены, или сервер банка не позволят вывести средства с адреса, атакующему необходимо контролировать 2 из 3 участников
- реализации и внутренние алгоритмы различных multisig могут сильно отличаться. Это касается порядка запросов и способов вывода средств

Multisig для банков

- код multisig, является простым в применении и конфигурировании и крайне надёжным, ибо является одним из наиболее ответственных “блоков“ для построения сложных систем доверия и передачи цифровых прав. По факту, простая транзакция перевода криптовалюты является вариантом multisig 1/1
- решения с использованием multisig просты, не содержат shared secrets и не требуют защищённых каналов связи – это огромное архитектурное преимущество
- если электронная подпись банка является юридически значимой, следовательно транзакции, созданные с использованием сертифицированных пар ключей по идее должны являться юридически значимыми
- лучшими кандидатами на звание доверенных участников, которым можно доверить третий ключ от multisig 2/3 – это банки
- в схеме multisig банк может являться гарантом восстановления средств в случае взлома или иных проблем у одного из участников
- никто из участников не раскрывает никаких секретных данных и не может контролировать критичные для остальных ресурсы без их согласия
- с использованием дополнительного внутреннего функционала (времени ожидания подтверждений, комиссий, и т.п.) multisig контракты позволяют автоматизировать множество типов сделок, учитывающих сроки и характер платежей
- оптимизирует банковскую инфраструктуру, позволяя пользователям самим процессить свои платежи, а банку вмешиваться только тогда, когда возникает необходимость в доверенном участнике. Также, банк имеет доступ к информации о платежах в „своих“ контрактах
- схема multisig в неявном виде уже присутствует в алгоритмах репликации высокодоступных баз данных, т.к. запись в БД также требует кворума нескольких работающих нод, так что эти алгоритмы уже давно и успешно работают
- multisig реализован в большом количестве блокчейн движков, схема работы банка легко тиражируется на несколько различных блокчейнов. А в большинстве блокчейнов также совместимы ключевые пары (ECS), что позволяет удобно управлять ключами

Multisig для банковской сферы, примеры использования

- аккредитив
  - покупатель создаёт контракт, multisig 2/3, с участием себя, продавца, и банка, и помещает средства на него
  - продавец создаёт транзакцию на вывод средств, и ожидает пока либо банк, либо покупатель разрешат вывод
  - далее, либо банк проверяет документы и отправляет своё согласие в контракт, либо покупатель делает это по доброй воле
  - контракт может легко дорабатываться в сторону ужесточения правил сделки, к примеру требуя обязательной фиксации сделки в виде пришедшего подтверждения от Госреестра, или обязательного участия банка при выводе (что превращает эту схему в multisig 2/2)

Multisig для банковской сферы, примеры использования

- факторинг
  - клиент, получающий оплату создаёт контракт-invoice
  - в контракте предусмотрено возможное участие банка (или нескольких), чей адрес фиксируется в контракте
  - после истечения срока погашения, клиент отправляет транзакцию в контракт, и он преходит в режим ожидания факторинга
  - банк (или один из банков) принимает решение поддержать invoice и присылет средства в контракт
  - клиент получает свои средства, информация о прошедшей факторинговой сделке зафиксирована в блокчейне
  - контракт также легко дорабатывается с учётом сроков, комиссий и других факторов

Multisig для банковской сферы, примеры использования

- commit-reveal cross-blockchain swap
  - client1 и client2 заявляют банку желание обменять 5 ETH у одного на 7 EOS у другого
  - банк генерирует **secret**, и сообщает его **хеш**(не сам secret) обоим клиентам
  - client1 создаёт **контракт в Ethereum**, позволяющий **client2** забрать **5 ETH**, если он представит **secret**. Для этого он помещает в контрат полученный от банка хеш от secret
  - client2 создаёт второй **контракт в EOS**, позволяющий **client1** забрать **7 EOS**, если он представит **secret**. Для этого он также помещает в контрат полученный от банка хеш от secret
  - банк опубликовывает secret
  - client1 и client2, получив secret забирают свои средства, отправляя транзакции withdraw, с приложенным secret
  - одобрение банка на сделку, история её прохождения и уникальный идентификатор в обоих блокчейнах прилагаются
  - *{{секретным шрифтом}} если при этом вместо ETH и EOS будут строчки „1.21 USD“ и „1.01 EUR“, то в смысле контракта ничего не поменяется. Клиентская транзакция сама по себе является одноразовым доказательством права client1 и client2 получить свои USD и EUR.*

Предвосхищая вопросы

- зачем тут блокчейн?
  - прозрачная и железобетоннная(местами даже слишком), бизнес-логика. Ни банк, ни клиенты не могут обманывать. Доверие к банку по прежнему важно для клиентов, но сделки гораздо проще и безопасней в реализации.
  - во всех вышеприведённых схемах ни банк, ни клиенты ни разу не отправили по незащищённым каналам информацию, перехват и расшифровка которой позволила бы злоумышленнику как то сломать логику контракта
  - подключение к этой „системе переводов“ выполняется посредством использования полностью свободного софта, который не может вдруг стать платным – иначе сеть потеряет смысл. За счёт открытого софта, также, появляется совместимость между банковскими системами.
  - весь банковский софт для работы с такими сервисами может быть с точки зрения безопасности абстрагирован от собственно процедуры подписи, которая может выполняться отдельным типом сервисов, с высочайшими уровнями безопасности

EOF

- ну а почему тут еще не блокчейн, раз всё так здорово?
  - мы очень ценим эту технологию
    - но всё таки она beta :)
      - и немного страшно...

Sergey Prilutskiy, Chief of Research @ MixBytes.io & Smartz.io