[Russian] White Paper

Edit New Page

snordenstorm edited this page on 17 Dec 2014 · 177 revisions

[Russian] White Paper — Манифест проекта Эфириум. Ссылка на англоязычный оригинал.

Зачем мне это нужно?

У всех сервисов, которыми вы пользуетесь в повседневности, есть нечто общее: они централизованы. Когда вы размещаете деньги в банке, вам приходится доверять ему — доверять его честности, защищённости от хакеров и государственного давления; верить в соблюдение им банковской тайны, верить в то, что проверка его деятельности проводится независимыми аудиторами. (Наконец, верить в беспристрастность судов в вашей местности, если по каким-то причинам так вышло, что вы судитесь с банком.) Абсолютно та же история и с размещением в Фейсбуке фотографий, и с важным документом, выложенным на Dropbox. Эта модель ущербна, но до недавних пор децентрализованных альтернатив не существовало.

А теперь есть Эфириум.

Приложения, построенные поверх Эфириум, не требуют от пользователя доверять вообще никому, а исходный код самого Эфириума открыт и может быть проанализирован каждым. Эфириум не только решает описанные выше проблемы, он также открывает дверь для всех типов децентрализованных приложений, в том числе тех, которые мы ранее не могли даже помыслить.

Обозначения

Биткойн в значении "сеть, система" мы будем писать с большой буквы. В значении валюты мы будем писать с маленькой — "переслал 1 биткойн", "переслал 2.71828 биткойнов". (Как Земля и земля.)

BTC — стандартное (на данный момент) обозначение для валюты биткойн.

Слова "узел", "нод" и "нода" (*от англ.* node — узел) в этом тексте обозначают абсолютно одно и то же.

(ссылочки посмотреть и подправить)

Платформа следующего поколения для смарт-контрактов и децентрализованных приложений

Когда Сатоши Накамото запустил блокчейн Биткойна в январе 2009-го, на самом деле он одновременно запустил также две радикальные и совершенно не тестировавшиеся до того момента концепции. Первая — это "биткойн", пиринговая децентрализованная валюта без так называемой внутренней стоимости или центрального эмитента. В качестве валюты Биткойн привлекал и привлекает некоторое внимание публики — как в политическом аспекте как валюта без центрального банка, так и как валюта с очень высокой волатильностью. Однако также есть другая, не менее важная часть большого эксперимента Сатоши: концепция основанного на proof-of-work блокчейна, с помощью которого и реализуется публичное децентрализованное соглашение о порядке транзакций. В Биткойн действует правило "кто первый встал, того и тапки": если кто-то имеет 50 BTC и одновременно пересылает 50 BTC пользователям A и B, то только транзакция, подтверждённая первой, пройдёт. Нет никакого способа узнать, какая из двух транзакций была "отправлена" раньше, и именно это десятилетиями заводило в тупик развитие децентрализованных цифровых валют. Блокчейн Сатоши был первым заслуживающим доверия децентрализованным решением. Сейчас внимание сообщества переключается на эту другую часть Биткойн-технологии — на то, каким образом идея блокчейна может быть использована для чего-то помимо денег.

Clone this wiki locally

https://github.com/snordens

Clone in Desktop

В этом контексте обычно говорят о блокчейн-цифровых активах, представляющих некоторые валюты и финансовые инструменты ("цветные монеты"), блокчейн-подтверждениях владения каким-либо устройством ("умная собственность"), незаменяемых активах, таких как доменные имена ("Неймкойн"), а также более продвинутых приложениях, таких как децентрализованные биржи, финансовые деривативы, пиринговые азартные игры и интегрированные в блокчейн системы идентификации и репутации. Другая интересная важная область — "смарт-контракты": представляющие собой программный код контракты, которые автоматически производят криптовалютные переводы и переводы других цифровых активов в соответствии с тем, что прописал автор контракта. К примеру, контракт может иметь форму "A может забрать вплоть до X единиц валюты в день, B — до Y единиц валюты в день, A и B по договорённости могут забрать всё, и A может отключить возможность B забирать валюту"; поскольку Эфириум предлагает Тьюринг-полный язык для написания таких контрактов, действие контракта натурально ограничено только фантазией автора (и количеством вычислительных операций, конечно — чтобы особо весёлые авторы не вешали сеть). Логичным расширением этой идеи являются децентрализованные автономные организации (ДАО) — долгосрочные смарт-контракты, хранящие активы и программно реализующие правила работы целой организации. Идея Эфириума — предоставить блокчейн со встроенным полностью самостоятельным Тьюринг-полным языком программирования, который может быть использован для создания любых мыслимых "контрактов", позволяя пользователям создавать любую из описанных выше систем,— равно как и такие, которые человечество ещё не смогло вообразить — просто записывая логику работы такой системы в несколько строчек кода.

Оглавление

Введение в Биткойн и существующие концепции

История

Концепция децентрализованной цифровой валюты, а также альтернативных приложений, таких как реестры собственности, витала в воздухе десятилетиями. Протоколы "электронного кэша" 80-х и 90-х, по большей части опиравшиеся на широко известное в криптографии понятие слепой подписи, предлагали валюту высокой степени анонимности, но не получили никакого распространения, поскольку принципиально зависели от централизованного посредника. Представленная в 1998 Вэй Даем (Wei Dai) концепция b-money стала первой, в которой говорилось как про идею децентрализованного соглашения между участниками, так и про идею создания денег посредством решения вычислительных задач. Однако его предложение было скудным на детали того, как конкретно можно было бы реализовать это децентрализованное соглашение. В 2005 Хал Финни (Hal Finney) предложил криптовалюту с концепцией "reusable proofs of work", которая использует идеи b-money вместе с Hashcash (прототипом proof-of-work системы Биткойн) Адама Бэка (Adam Back) — к сожалению, его идея также включала в себя посредника, которому пользователи должны были доверять.

Поскольку речь идёт о валюте, порядок проведения транзакций часто критически важен; в децентрализованной валюте должно быть реализовано децентрализованное соглашение о порядке транзакций между участниками. Основное препятствие, с которым сталкивались все протоколы валют до Биткойна, в том, что, опираясь на разные вариации задачи византийских генералов, они решали лишь часть проблемы. Эти протоколы предполагали, что все участники системы известны друг другу, и их авторы, комментируя вопросы безопасности, делали утверждения вида "наша система с N участниками безопасна, если не более N/4 участников — злоумышленники". Что никак не позволяет запустить на таком протоколе электронную валюту, т.к. в условиях анонимности атака Сивиллы — атака, при которой симулируются тысячи узлов на сервере или ботнете (что можно сделать с лёгкостью) и атакующий получает сколько нужно процентов "пользователей" всей сети — радостно передаёт нам привет.

Инновация, внедрённая Сатоши, заключалась в том, что он скомбинировал протокол децентрализованного соглашения (очень простой, основанный на узлах, объединяющих транзакции в "блоки" каждые 10 минут) и концепцию proof-of-work. Согласно концепции proof-of-work для добычи монет нужно тратить вычислительные мощности, так что сделать хардфорк блокчейна и перерисовать историю транзакций как вздумается задёшево не получится *(стоимость "атаки 51%" на ноябрь 2014 года составляет не менее 150 million USD - прим.перев.)*. Влияние каждого узла в Биткойн пропорционально его вычислительной мощности; организовать вычислительную мощность больше, чем мощность всей сети, многократно сложнее, чем просто симулировать миллион узлов. Несмотря на грубую простоту блокчейна Биткойн, он доказал свою работоспособность, и стал колыбелью более чем 200 криптовалют, выпущенных на основе его кода за 5 лет работы системы.

Биткойн как система изменения состояний

Функцией изменения состояния называется та последовательность действий, которая осуществляется протоколом валюты (в данном случае Биткойн-протоколом) при попытке произвести транзакцию. Функция изменения состояния Биткойн-протокола, вы не поверите, производит Биткойн-транзакцию. Но бывают и нетривиальные функции изменения состояния (см. про метакойны ниже).

О блокчейне Биткойна можно думать как о системе изменения состояний. "Состояние" описывает то, как распределены все существующие биткойны между различными адресами. "Функция изменения состояния" — функция, по состоянию и транзакции возвращающая новое состояние. В стандартной банковской системе, для примера, состояние — список балансов каждого из клиентов, транзакция — запрос о переводе $X от A к B, и функция изменения состояния уменьшает значение на счёте A на $X и увеличивает значение на счёте B на $X. Если на счёте A меньшая сумма, чем $X, функция изменения состояния возвращает ошибку. Формально это выглядит так:

```
APPLY(S,TX) -> S' or ERROR
```

В банковской системе, описанной выше

```
APPLY({ Alice: $50, Bob: $50 },"send $20 from Alice to Bob") = { Alice: $30, Bob: $70 }
```

Но:

```
APPLY({ Alice: $50, Bob: $50 },"send $70 from Alice to Bob") = ERROR
```

"Состояние" системы Биткойн — множество всех добытых непотраченных монет (технически, "непотраченные результаты входящих транзакций", НРВТ) такое, что у каждого НРВТ указаны достоинство и владелец. Владелец — это просто номер кошелька, 20-байтовый адрес, называемый *публичным ключом*[1]. У каждой транзакции есть один или больше "входов", и каждый из входов содержит указание на пересылаемые НРВТ и предоставляемую криптографическую подпись; выходов тоже может быть один или несколько, и каждый выход содержит новые НРВТ. Нет никаких ограничений по количеству кошельков, которые могут быть "на входе" и "на выходе" *(да?)*.

Каждая транзакция полностью декларируется отправителем — и то, с каких кошельков и сколько валюты подаётся "на вход" транзакции, и то, сколько и кому должно достаться "на выходе". Только потом проверяется валидность транзакций (и делают это майнеры *(да?)*). Принцип работы функции изменения состояния `APPLY(S,TX) -> S'` выглядит так:

1. Для каждой транзакции в `TX`:
   - Если пересылаемый НРВТ отсутствует в состоянии `S`, вернуть ошибку.
   - Если предоставленная цифровая подпись не совпадает с подписью владельца НРВТ, вернуть ошибку.
2. Если сумма всех денег "на входе" меньше, чем сумма всех денег "на выходе", вернуть ошибку.
3. Вернуть `S`, у которого отправленные на вход транзакции НРВТ удалены, и к которому добавлены все НРВТ, являющиеся "выходами" транзакции.

Первая часть первого пункта не позволяет отправителям тратить несуществующие монеты, вторая часть не позволяет тратить монеты других людей. Второй пункт нужен понятно зачем; третий же собственно реализует изменение балансов. Напоследок разберём простенький пример. Пусть Алиса хочет переслать 11.7 BTC Бобу. Для этого сначала Алиса должна найти набор НРВТ, обладание которыми она может подтвердить и сумма средств которых равна как минимум 11.7 BTC (на деле всем этим, конечно, занимается не Алиса, а её Биткойн-клиент — программа, позволяющая работать с биткойнами). В большинстве ситуаций Алисе не удастся найти у себя несколько НРВТ, сумма средств на которых в точности бы равнялась 11.7 BTC; вместо этого ей, например, пересылали 6, 4 и 2 BTC, и наименьшее число, которое она может собрать из имеющихся у неё НРВТ, равно 6+4+2=12. Поэтому она (её Биткойн-клиент) создаёт транзакцию с тремя входами и двумя выходами. Первым выходом будет адрес Боба, куда отправится 11.7 BTC, вторым — один из кошельков Алисы, куда отправится 0.3 BTC "сдачи".

Майнинг

Если бы речь шла о "заслуживающем доверия" централизованном сервисе (PayPal, WebMoney), всё было бы просто: транзакции проходят через центральные сервера и их история хранится на них же. Однако, поскольку хочется иметь по-настоящему децентрализованную валюту, необходимо скомбинировать систему проведения транзакций с системой децентрализованного соглашения, чтобы соглашение о порядке транзакций у каждого пользователя сети было одно и то же. В системе Биткойн это достигается путём постоянного запаковывания новых транзакций в "блоки". Протокол Биткойн-сети устроен так, что майнеры (miners; люди, занимающиеся майнингом (*от англ.* to mine — добывать, по аналогии с to mine gold — добывать золото)) производят новый блок примерно раз в десять минут; каждый блок содержит таймштамп, значение nonce (сокращение от number used once), хэш предыдущего блока (т.е. указание на него) и список всех транзакций, которые в наш блок попали (обычно это все транзакции, отправленные в сеть с начала майнинга предыдущего блока). Так и возникает блокчейн (*букв. перев.* "цепочка блоков"). Изменений в блокчейне не происходит, он лишь постоянно растёт; блокчейн хранит в себе историю всех транзакций и совершенно децентрализован.

Алгоритм проверки валидности блока в этой парадигме таков:

1. Проверить, существует ли и валиден ли предыдущий блок (тот, на который рассматриваемый блок указывает как на предыдущий).
2. Проверить, что таймштамп рассматриваемого блока больше, чем таймштамп предыдущего[2] (принцип причинности), но не более, чем на 2 часа.
3. Проверить валидность proof-of-work рассматриваемого блока.
4. Пусть `S[0]` — состояние после добавления в блокчейн предыдущего блока.
5. Пусть `TX` — список транзакций в рассматриваемом блоке, а всего этих транзакций `n`. Для всех `i` из набора `0..n-1` зададим `S[i+1] = APPLY(S[i],TX[i])`. Если любая такая apply-процедура выдаёт ошибку, выйти из цикла и выдать ошибку.
6. Код выполнен успешно, и `S[n]` — состояние после обсчёта транзакций рассматриваемого блока.

Заметим, что состояние (счёта) не включено в блок ни в коей мере; узлы, занимающиеся валидацией, вычисляют его (для проверки того, что деньги у пациента вообще есть), проходя весь путь по блокчейну с самого начала. Также видно, что порядок, в котором майнер включает транзакции в блок, имеет значение: если в данном блоке есть две транзакции A→B и B→C, такие, что B тратит средства (НРВТ), предоставленные A, то угадайте, дети, в каком порядке эти транзакции должны идти, чтобы всё получилось?..

Нетривиальная часть представленного алгоритма валидации блока — так называемая концепция "proof-of-work": условие того, что SHA256-хэш каждого блока как 256-битное число должен быть меньше динамически изменяющейся величины под названием таргет, равной на момент написания этих строк примерно 2^{192}. (Текущее значение таргета в 16-ричной форме.) Эта динамически изменяющаяся величина явно вычисляется по так называемой "сложности"; сложность (а, значит, и вычисляемый по ней таргет) перенастраиваются каждые 2016 блоков так, чтобы, какой бы суммарная вычислительная мощность сети ни была, в среднем каждый новый блок обнаруживался примерно раз в 10 минут. Сделано это для того, чтобы сделать нахождение блоков вычислительно сложным (протокол Биткойн признаёт истинной самую длинную цепочку блокчейна; для того, чтобы иметь возможность атаковать сеть, "перерисовывая" блокчейн как вздумается, необходимо контролировать 51% вычислительной мощности сети, что намного сложнее, чем, например, симулировать 51% узлов). Поскольку SHA-256 — совершенно непредсказуемая псевдослучайная хэш-функция, единственный способ найти подходящий (т.е. меньший, чем таргет) хэш — простой перебор значений nonce. (Если интересно, что такое nonce, от чего конкретно берётся хэш-функция — словом, если интересна матчасть, то вот, пожалуйста, специальная статья, посвящённая матчасти майнинга. Для подбора хэша, который как число меньше, чем 2^{192}, нужно потратить, в среднем, 2^{64} попытки. В чём смысл людям заниматься майнингом? В том, что нашедшему подходящий хэш майнеру выплачивается вознаграждение, в данный момент равное 25 BTC (примерно 12000$ на сентябрь 2014). Это единственный способ эмиссии (выпуска) биткойнов. Стоит также отметить, что если сумма на входе больше, чем сумма на выходе (отправитель установил "комиссию"), майнер, нашедший блок, в который попала эта транзакция, забирает "комиссию" себе. (В настоящее время комиссия является почти обязательной; рекомендуемое значение для средней транзакции равно 0.0001 BTC — это примерно 2 рубля. За деталями сюда.) Принято считать, что, когда награда за нахождение блока существенно уменьшится, комиссии станут основным доходом майнеров, и формироваться они будут исключительно свободным рынком.

Для лучшего понимания смысла майнинга (выше описано нечто довольно хитроумное, согласитесь) обсудим, что произойдёт при попытке атаки. Поскольку криптография, лежащая в основе Биткойн (SHA-256 алгоритм хеширования), проверена временем и на данный момент считается надёжной (и, в частности, активно используется банками), целью злоумышленника будет часть системы, напрямую не связанная с шифрованием — порядок транзакций.

Стратегия злоумышленника могла бы быть такой:

1. Переслать продавцу 100 BTC за некоторый продукт (лучше всего, цифровой продукт быстрого получения)
2. Дождаться получения продукта
3. Произвести другую транзакцию, переслав 100 BTC на другой свой адрес
4. Попытаться убедить систему, что транзакция самому себе была произведена первой.

Как только произошёл шаг (1), спустя несколько минут некоторый майнер включит эту транзакцию в блок — например, блок 270000. Примерно за час ещё пять блоков будут добавлены в блокчейн, и каждый из новых блоков будет косвенно указывать на транзакцию (1), таким образом подтверждая её. В этот момент продавец решит, что оплата завершена, и перешлёт продукт; поскольку мы рассматриваем случай цифрового продукта, доставка моментальна. Теперь злоумышленник создаёт другую транзакцию, пересылая 100 BTC себе. Если он просто отправит её в сеть как обычно, транзакция не произойдёт; майнеры, пытаясь провести эту транзакцию, запустят `APPLY(S,TX)` и получат ошибку, поскольку `TX` (транзакция) пытается потратить HPBT, которого уже нет. Вместо этого злоумышленник мог бы "сделать форк" блокчейна: начать майнить другую версию блока 270000, отмечающую как предыдущий блок 269999, но с новой (когда он пересылает 100 BTC себе) транзакцией вместо предыдущей. Запустить параллельный "отросток" блокчейна и попытаться обогнать основную цепочку (ещё раз подчеркнём, что, согласно протоколу Биткойн, настоящей является самая длинная цепочка блокчейна). Поскольку информация в этом блоке другая, для своей версии блока 270000 ему придётся заново искать хэш-победитель (=переделывать proof-of-work). Более того, новая версия блока 270000 имеет другой хэш, и блоки 270001-270005 ссылаются не на него; таким образом, блокчейн всей сети и блокчейн злоумышленника совершенно различны. Поскольку Биткойн-протокол считает самую длинную цепочку блокчейна истинной, "честные" майнеры будут продолжать майнинг цепи блокчейна длиной 270005, а злоумышленник будет в одиночку майнить свою. Злоумышленник может сделать свою цепь самой длинной, только если его вычислительные мощности больше суммарной вычислительной мощности всей остальной сети (так называемая "атака 51%").

Деревья Мёркла

Слева: лишь малое число вершин в дереве Мёркла необходимо для доказательства валидности ветви.

Справа: любая попытка изменить любую часть дерева Мёркла приведёт к несогласованности.

Важным для масштабируемости свойством протокола Биткойн является то, что каждый блок хранится в блокчейне как многоуровневая структура информации. Упомянутый в прошлом подразделе хэш блока — на самом деле хэш заголовка блока, который, в свою очередь, является ~200-байтовым куском информации, включающим в себя таймштамп, nonce блока, хэш предыдущего блока и хэш корня структуры под названием дерево Мёркла, которая и хранит всю информацию о транзакциях блока. Дерево Мёркла — бинарное дерево, состоящее из 1) большого количества листовых вершин внизу, которые и содержат всю информацию 2) вершин в середине дерева, каждая из которых является хэшем двух своих детей 3) "корня дерева" (самого верхнего хэша), также являющегося хэшем двух своих детей. Смысл дерева Мёркла в быстрой проверке валидности нужной информации: узел может загрузить заголовок блока из некоторого источника, а затем малую часть дерева с нужными ему транзакциями из другого источника, и быстро удостовериться, что вся информация корректна. Причина, почему это работает, в том, что хэши вычисляются вверх: если злоумышленник попытается "подсунуть" фейковую транзакцию вниз дерева Мёркла, все вершины сверху, с которыми она связана, также изменят свои хэши, и хэш заголовка блока тоже изменится, что приведёт к тому, что протокол будет воспринимать такой блок как совершенно другой (proof-of-work которого будет практически наверняка не валиден (см. раздел "Майнинг" о том, что хэш является валидным, если он как 256-битовое число меньше некоторого динамически подстраивающегося числа (таргета); валидный хэш очень трудно найти)).

Деревья Мёркла, по-видимому, необходимы для жизнеспособности системы Биткойн в долгосрочной перспективе. На апрель 2014 года полная нода Биткойн-сети (полной нодой называется узел, который хранит и непрерывно обновляет всю копию блокчейна, в т.ч. занимаясь валидацией новых блоков) занимает 15 Гб дискового пространства и растёт более чем на 1 Гб в месяц. На данный момент хранить полную копию блокчейна могут позволить себе владельцы ПК (но не владельцы смартфонов); в будущем, по-видимому, только бизнесы и отдельно взятые фанаты будут держать полные ноды. Протокол упрощённой верификации платежей (УВП) допускает существование также "лёгких нод" (лёгких кошельков), которые загружают лишь заголовки блоков, проверяют их proof-of-work, и затем загружают ветви дерева Мёркла лишь с необходимыми держателю такой ноды транзакциями. Это позволяет держателям лёгких нод надёжно определить статус любой Биткойн-транзакции, а также собственный текущий баланс, скачав крайне малую часть всего блокчейна.

Альтернативные приложения блокчейна

Идея нефинансового применения блокчейн-технологии также имеет долгую историю. В 2005 году Ник Сабо (Nick Szabo) изложил концепцию безопасного хранения права на собственность, рассказав, как распределённые копии баз данных (блокчейн-система, по сути) могут использоваться для хранения реестра земельной собственности — что возможно выработать фреймворк, способный описывать, в частности, самообеспечение, "недружественное владение" землёй и налог, выплачиваемый при уменьшении стоимости земельной собственности (при недружественном владении). Однако, к сожалению, никакой эффективной реализации распределённых копий баз данных на тот момент не было, и протокол не имел практического применения. С 2009, однако, с развитием системы децентрализованного соглашения Биткойн число альтернативных приложений блокчейна стало быстро увеличиваться (благо исходный код Биткойн-протокола открыт).

- **Неймкойн** (Namecoin) — созданная в 2010 децентрализованная система регистрации и хранения произвольных комбинаций вида "имя-значение", наиболее известным применением которой является система альтернативных корневых DNS-серверов. Такие децентрализованные протоколы, как Tor, Bitcoin и BitMessage, решают проблему идентификации аккаунтов присваиванием идентификатора в виде псевдослучайного хэша вида `1LW79wp5ZBqaHW1jL5TCiBCrhQYtHagUWy` . При этом было бы здорово иметь аккаунты с именами (идентификаторами) наподобие "george". Проблема, очевидно, в том, что если какой-то пользователь может создать аккаунт "george", то (поскольку речь идёт о децентрализованной системе) кто-то другой сможет таким же способом зарегистрировать аккаунт с тем же именем. Единственное решение здесь — парадигма «кто раньше зарегистрировал пользователя "george", того и тапки», которая как раз прекрасно сочетается с Биткойн-протоколом. Неймкойн — первый и самый успешный пример такой системы регистрации имён.

- **Цветные монеты** — протокол, позволяющий людям создавать собственные цифровые валюты или жетоны на блокчейне Биткойна. Цветные(=окрашенные) монеты появляются в результате того, что кто-то из пользователей решает публично покрасить некоторые из своих Биткойн-НРВТ (упрощая, какую-то часть своих биткойнов) — скажем, в цвет "Audi X6"; в результате транзакции окрашенные монеты останутся окрашенными. Удобны и, по-видимому, в скором времени войдут в оборот "монохромные" кошельки (кошельки с монетами одного какого-то цвета), позволяющие наглядно осуществлять трансконтинентальную торговлю тем или иным товаром в промышленных масштабах. После окрашивания биткойны перестают быть деньгами и становятся жетонами, доказывающими владение каким-то конкретным активом.

  Таким образом, идея цветных монет — в том, чтобы использовать блокчейн-технологию для представления других вещей. Монеты в блокчейне сети Биткойн могут использоваться, скажем, для представления акций компании, слитков золота или документов на владение домом. При помощи цветных монет кто угодно может выпустить свои акции или торговать товарами, используя инфраструктуру сети Биткойн. Это предоставляет биткойнам огромное преимущество по сравнению с обычными методами передачи активов.

Сейчас, если вы хотите передать золото, акции компании, или документы на дом, вам потребуется осуществить сложный процесс, в который включены нотариусы, регистрации в госархивах и бумажные документы. Всё это для того, чтобы удостовериться, что переход права собственности на определённый объект точно состоялся.

Криптографическая технология Биткойн построена по-другому. Вы точно знаете, что некоторое количество биткойнов покинули один адрес и пришли на другой. Вы можете быть уверены, что это произойдёт всего лишь за несколько минут. Если биткойны могут представлять другие активы, то права на эти активы можно переслать быстро и легко, кому угодно, из любой точки земного шара. Факт пересылки, как вы понимаете (блокчейн-технология же), совершенно публичен.

Про статус разработки кошельков и бирж, связанных с цветными монетами, на момент начала июня 2014 написано здесь.

- **Метакойны** — их идея в том, что можно использовать протокол, действующий поверх Биткойн-сети, использующий биткойн-транзакции для хранения метакойн-транзакций, но имеющий другую функцию изменения состояния `APPLY'`. Это означает, что некоторые из биткойн-транзакций метакойн-протокол может интерпретировать как метакойн-транзакции (такие транзакции называются метакойн-валидными), при этом функция изменения состояния этой транзакции как метакойн-транзакции может быть абсолютно непохожа на функцию изменения состояния этой транзакции как биткойн-транзакции. Поскольку метакойн-протокол не может предотвратить появление метакойн-невалидных транзакций в Биткойн-блокчейне (естественно, далеко не все биткойн-транзакции понимаются метакойн-протоколом как "свои"), в протокол добавляется логичное правило: если `APPLY'(S, TX)` (действие этой новой функции `APPLY'`) возвращает ошибку, применить `APPLY'(S, TX) = S` (метакойн-балансы не изменились). Метакойны — лёгкий механизм создания произвольного криптовалютного протокола "малой кровью", т.к. "паразитируя" на Биткойн-блокчейне, ведя бурную жизнь поверх него, не нужно заниматься майнингом и распространением собственного блокчейна. Наиболее известная монета, относящаяся к классу метакойнов — Мастеркойн. Пример разбора Мастеркойн-транзакции.

Таким образом, есть два подхода к достижению децентрализованного соглашения: построить независимую сеть, или же построить протокол поверх сети Биткойн. Первый подход, при всей его успешности в приложениях вроде Неймкойн, трудноосуществим; для реализации каждой конкретной идеи необходимо создавать свой блокчейн, а также создать и протестировать весь необходимый код. Кроме того, 1) децентрализованных приложений со временем станет настолько много, что каждому из них создавать для своих нужд отдельный блокчейн было бы совершенно нелепо 2) есть целый класс децентрализованных приложений (в т.ч. децентрализованных автономных организаций), которые нуждаются во взаимодействии друг с другом.

Опирающийся на блокчейн Биткойна подход, с другой стороны, имеет такой недостаток: он не наследует систему упрощённой верификации платежей (УВП), возможную в Биткойн (вместе со всеми прелестями вроде "лёгких кошельков"). УВП возможна в Биткойн потому, что глубина Биткойн-блокчейна является как бы гарантией валидности: если предшественники транзакции находятся достаточно глубоко в блокчейне, можно с уверенностью сказать, что эти предшественники легитимны (а, значит, легитимен и баланс на момент начала транзакции). Основанные на Биткойн мета-протоколы, однако, не могут требовать от Биткойн-блокчейна не включать транзакции, невалидные с точки зрения этого самого мета-протокола (т.е. попросту не относящиеся к мета-протоколу). Следовательно, внедрение полностью безопасного УВП-мета-протокола потребовало бы пробегать Биткойн-блокчейн вплоть до начала для проверки валидности той или иной транзакции. *(Здесь рассказать своими словами о том, почему для метакойнов нельзя сделать лёгкие кошельки тупо по аналогии с лёгкими кошельками для биткойнов.)* На данный момент все реализации УВП для мета-протоколов полагаются на "заслуживающий доверия сервер", что особенно неприемлемо в свете того, что одно из главных преимуществ криптовалют — отсутствие необходимости кому-либо доверять.

Написание сценариев (скриптов)

Даже в отсутствие каких-либо расширений протокол Биткойна поддерживает примитивную версию "смарт-контрактов". Владение НРВТ в Биткойн можно подтверждать не только публичным ключом, но и скриптом, написанным на простом stack-based языке программирования. Транзакция, тратящая такие НРВТ, должна предоставить данные, которые удовлетворят такой скрипт. На самом деле даже стандартный (использующий публичный ключ) способ владения аккаунтами реализован как скрипт: "на входе" он берёт цифровую подпись ECDSA, verifies it against the transaction and the address that owns the UTXO, и возвращает 1 в случае успешной верификации и 0 в случае неуспешной. Возможны более замысловатые скрипты. К примеру, возможен скрипт, согласно которому для получения возможности тратить средства необходимо предоставить 2 из 3 цифровых подписей "владельцев" средств (так называемая мультиподпись). Это может быть полезно для корпоративных аккаунтов, аккаунтов безопасного хранения и торговли с участием гаранта (=эскроу). Скрипты также могут быть использованы для автоматизации выплат наград за решение важных вычислительных задач; можно даже создать скрипт, говорящий что-то вроде "эта Биткойн-НРВТ Ваша, если предоставлено доказательство того, что Вы переслали столько-то догекойнов мне" — и это уже почти децентрализованная биржа криптовалют, не правда ли?

Однако встроенный в Биткойн язык сценариев имеет ряд серьёзных ограничений:

- **Отсутствие Тьюринг-полноты** — этот язык позволяет реализовать далеко не все виды вычислений. В частности, запрещены все петли (циклы). Это сделано, чтобы избежать длинных циклов во время верификации транзакций; теоретически это преодолимое препятствие для авторов скриптов, поскольку любой небесконечный цикл может быть симулирован достаточно большим количеством `if` 'ов — но это, конечно, приводит к "раздутию" такого скрипта, увеличению количества строк в нём во много раз и, как следствие, сравнительно большой комиссии за обработку такой транзакции (комиссии пропорциональны количеству передаваемых байтов). К примеру, для того, чтобы написать с нуля альтернативный алгоритм цифровой подписи эллиптической кривой, по-видимому, придётся вручную прописать 256 повторяющихся операций умножения.

- **Финансовая слепота** — скрипт не видит, сколько денег может быть снято со счёта. Рассмотрим хеджирующий контракт, в который A и B закладывают эквивалент $1000 в биткойнах, и спустя 30 дней скрипт пересылает стороне A эквивалент $1000 в биткойнах (по новому курсу BTC/USD), а остальное пересылает стороне B. Безусловно, остаётся та трудность, что придётся как-то научить такой контракт узнавать текущую стоимость 1 BTC в долларах, но даже при этом такой контракт был бы существенным прогрессом по сравнению с существующими централизованными решениями (прежде всего, по параметрам "необходимость доверия" и "количество необходимой инфраструктуры"). Однако поскольку нельзя взять часть НРВТ, единственный способ реализовать такой контракт — иметь НРВТ для каждой из возможных денежных сумм (0.26936152 BTC, 11.9132312 BTC), что, ну, невозможно. (НРВТ — неделимые куски, текущее состояние счёта есть сумма всех имеющихся у тебя НРВТ, при попытке перевести кому-то деньги пользователь берёт сколько нужно НРВТ и подаёт на "вход" транзакции, они исчезают, но у него появляется новая, равная "сдаче"; см. разъяснение в конце этого подраздела). *(Приведённое здесь рассуждение выглядит так: с помощью публичного ключа можно тратить любые суммы, а с помощью языка сценариев Биткойн — лишь "целые куски" нашего электронного золота (НРВТ). Но контролирование с помощью публичного ключа также реализовано как сценарий, и здесь моя логика заходит в тупик.)*

- **Отсутствие промежуточных состояний** — НРВТ может быть только потраченным или непотраченным, промежуточных вариантов нет. Нет никакой возможности написать контракт из нескольких стадий, который мог бы содержать в себе переход к некоторому промежуточному состоянию. Биткойн-сценарии не приспособлены для многоуровневых контрактов с опциями, децентрализованных обменников и двустадийных протоколов криптографического соглашения (необходимых для безопасности наград за решение той или иной вычислительной задачи). Сказанное означает, что НРВТ могут быть использованы только для простых моментально исполняющихся контрактов, но никак не для более замысловатых контрактов с опциями, промежуточными стадиями и состояниями. По этой причине децентрализованные организации не могут быть реализованы на этом языке, по ней же трудно внедрить мета-протоколы. Бинарность состояния в сочетании с финансовой слепотой также приводит к тому, что на языке сценариев Биткойн задать предельное значение суммы, которую можно снять с кошелька, не получится.

- **Блокчейн-слепота** — НРВТ не видят информацию из блокчейна (такую как nonce и предыдущий хэш). Это сильно ограничивает возможные приложения в азартных играх и некоторых других категориях, лишая язык сценариев естественного источника случайных чисел.

Таким образом, мы видим три подхода к разработке нетривиальных приложений с участием криптовалюты: создать новый блокчейн, использовать язык сценариев Биткойн, и создать мета-протокол поверх Биткойн. Создание нового блокчейна приводит к неограниченной свободе творчества, за которую придётся заплатить временем разработки и нервами, потраченными на начальную настройку, на фикс багов. Язык сценариев Биткойн прост в использовании и стандартизации, но крайне ограничен в возможностях. Мета-протоколы также просты, но трудномасштабируемы из-за невозможности нормально реализовать упрощённую верификацию платежей, а вместе с ней и "лёгкие" клиенты. Разрабатывая Эфириум, мы надеемся создать наиболее общий фреймворк, включающий в себя плюсы всех трёх парадигм.

Эфириум

Цель Эфириума — объединив и доработав концепции скриптинга, альткойнов и мета-протоколов, позволить всем желающим создавать произвольные децентрализованные приложения, обладающие одновременно свойствами масштабируемости, стандартизации, feature-полноты, лёгкости разработки и совместимости. Эфириум добивается этого построением предельно базовой платформы: блокчейна со встроенным Тьюринг-полным языком программирования, позволяющим любому желающему писать смарт-контракты и децентрализованные приложения, где каждый сможет создавать собственные произвольные правила владения, форматы транзакций и произвольные функции изменения состояния. Реализация идеи известной криптовалюты Неймкойн (Namecoin) на этом языке занимает (это не шутка) две строки кода, а такие протоколы, как валюты и системы репутации, можно реализовать менее чем в двадцать строк. Смарт-контракты, криптографические "коробки", содержащие значение и открывающиеся только при определённых условиях, также могут быть построены поверх платформы Эфириум, причём со значительно большей функциональностью, чем та, что предложена в языке сценариев Биткойн, по причине Тьюринг-полноты языка Эфириум, его финансовой зрячести, блокчейн-зрячести и наличия промежуточных состояний (Эфириум лишён всех недостатков, описанных в предыдущем параграфе).

Эфириум-аккаунты

В Эфириум состояние сделано из "аккаунтов"; каждому аккаунту соответствует его 20-байтовый адрес. Функции изменения состояния — переводы валюты или информации между аккаунтами. Эфириум-аккаунт содержит четыре поля:

- **nonce** — счётчик, необходимый для того, чтобы каждая транзакция произошла ровно один раз
- текущий **баланс** аккаунта (в ETH)
- **код контракта**, связанного с аккаунтом (если контракт есть)
- **хранилище** аккаунта (пустое по умолчанию)

"Эфир" (ether, ETH) — главное внутреннее "крипто-топливо" Эфириума, он же используется для уплаты комиссий. Возможны два способа ведения аккаунтов: контролируемый владельцем аккаунт (externally owned account) и аккаунт, контролируемый кодом связанного с ним контракта (контракт-аккаунт). Первый способ — это ровно то, что было в системе Биткойн: аккаунт, единственной опцией которого является пересылка денег (создание исходящей транзакции); путём создания и подписания транзакций с такого аккаунта можно пересылать сообщения. В случае контракт-аккаунта каждый раз, когда аккаунт получает входящее сообщение, активируется код контракта, который может открывать сообщению возможность писать в хранилище и считывать оттуда информацию, посылать другие сообщения и создавать другие контракты. (Слово "контракт", которое часто употребляется далее в этом тексте, является просто короткой формой для "контракт-аккаунт".)

Сообщения и транзакции

"Сообщения" в Эфириум — нечто похожее на транзакции в Биткойн, но с тремя существенными отличиями. Во-первых, сообщение в Эфириум может быть создано как внешним источником (=человеком), так и контрактом, в то время как Биткойн-транзакция может быть создана только человеком. Second, there is an explicit option for Ethereum messages to contain data. В-третьих, получатель Эфириум-сообщения, если это контракт-аккаунт, имеет возможность "ответить" автору на него; этим Эфириум-сообщения похожи на обычные функции.

Термин "транзакция" в Эфириум означает подписанный пакет данных, содержащий сообщение, отправляемое контролируемым владельцем аккаунтом. Транзакции включают в себя получателя, электронную подпись отправителя, количество пересылаемого эфира и пересылаемую информацию, а также значения двух переменных `STARTGAS` и `GASPRICE`. Чтобы предотвратить экспоненциальное раздутие количества вычислительных операций кода и бесконечные циклы, отправитель каждой транзакции устанавливает предел количества вычислительных шагов, которые допускаются для исходного + всех порождённых сообщений. `STARTGAS` — это именно этот предел, а `GASPRICE` — комиссия, которая платится майнеру за каждый вычислительный шаг (по сути, предлагаемый отправителем курс GAS/ETH). Если при выполнении транзакции "газ заканчивается", изменения откатываются и происходит возврат к начальному состоянию системы, с одной, правда, оговоркой: комиссии, выплаченные майнерам за их вычисления, остаются у майнеров. Если выполнение транзакции прерывается при ненулевом количестве остающегося газа, этот газ пересылается обратно отправителю транзакции. Таким образом, авторы контрактов обеспечивают их выполнение количеством газа, вложенным ими в контракт (очень похоже на пополнение баланса мобильного телефона). Также есть специальный тип транзакций и специальный тип сообщений для создания контрактов; адрес контракта вычисляется на основе хэша nonce аккаунта и содержащейся в транзакции информации.

Важным следствием механизма сообщений является принцип равноправия Эфириум-аккаунтов, управляемых контрактом и человеком — контракт-аккаунты имеют весь тот же функционал, что доступен аккаунтам, управляемым человеком, включая возможность пересылать сообщения и создавать другие контракты. Это приводит к тому, что контракты могут даже, как люди, играть в сообществе разные роли одновременно: к примеру, членом децентрализованной организации (контракт) может оказаться гарант-аккаунт (второй контракт) между параноиком, использующим специально изготовленные стойкие к квантовому компьютеру подписи Лэмпорта (третий контракт) и лицом, использующим аккаунт с пятью ключами для безопасности (четвёртый контракт). Сила платформы Эфириум в том, что децентрализованной организации, равно как и гарант-контракту, не нужно заботиться о типе аккаунта того или иного агента, о том, человек это или контракт.

Функция изменения состояния в Эфириум

Эфириум-функция изменения состояния `APPLY(S,TX) -> S'` работает так:

1. Проверить корректность оформления транзакции (т.е. что все нужные значения указаны), валидность цифровой подписи, и соответствие nonce транзакции nonce аккаунта отправителя. Вернуть ошибку, если что-то не так.

2. Вычислить комиссию по формуле `STARTGAS * GASPRICE` и определить адрес отправителя по цифровой подписи. Вычесть величину комиссии из баланса отправителя и увеличить на единицу nonce аккаунта отправителя. Если баланс отправителя меньше значения взимаемой комиссии, вернуть ошибку.
3. Присвоить `GAS = STARTGAS`, и взимать определённое количество газа за каждый обработанный майнером байт транзакции.
4. Переслать сумму транзакции (в ETH) с аккаунта отправителя на аккаунт получателя. Если аккаунта получателя не существует, создать его. Если аккаунт получателя — контракт-аккаунт, запустить код контракта (и тут может быть два исхода — либо контракт будет вычислен до конца, либо во время вычисления закончится газ).
5. Если денежный перевод не удался по причине того, что у отправителя не было суммы перевода, или при вычислении контракта закончился газ, откатить все изменения, кроме выплаты комиссий за вычисления майнеру.
6. В случае успешного выполнения контракта пересчитать по курсу ETH/GAS, указанному в переменной `GASPRICE`, в единицы эфира оставшееся количество газа, и вернуть этот эфир отправителю, а весь потраченный переслать майнеру.

Рассмотрим пример. Пусть код контракта таков:

```
if !contract.storage[msg.data[0]]:
    contract.storage[msg.data[0]] = msg.data[1]
```

Заметим, что в реальности контракт пишется на низкоуровневом языке ("EVM-код", EVM = Ethereum Virtual Machine); этот пример написан для ясности на Serpent, нашем высокоуровневом языке, который может быть скомпилирован до EVM-кода. Предположим, что до транзакции хранилище нашего контракт-аккаунта пусто, а в результате транзакции пересылается 10 единиц эфира (ETH), контракт обеспечен 2000 единицами газа (`STARTGAS = 2000`), `GASPRICE = 0.001`, и дополнительно пересылается информация `[2, 'CHARLIE']`[3]. Для этого примера Эфириум-функция изменения состояния сработает так:

1. Проверить, что транзакция валидна и корректно оформлена.
2. Проверить, что у отправителя есть как минимум 2000*0.001 = 2 единицы эфира. После успешной проверки вычесть 2 ETH из баланса аккаунта отправителя.
3. Присвоить переменной `GAS` значение 2000; в предположении, что транзакция занимает 170 байт и комиссия за байт равна 5, вычтем 850; таким образом, останется 1150 единиц газа.
4. Вычесть ещё 10 ETH из баланса аккаунта отправителя и добавить их к балансу получателя (в рассматриваемом случае — контракт-аккаунта, на который такую транзакцию отослали).
5. Запустить выполнение кода. В нашем примере это совсем просто: код проверяет, есть ли в хранилище контракт-аккаунта информация по адресу `2`, замечает, что её там нет и прописывает туда значение `CHARLIE`. Предположим, это стоит 187 единиц газа, и газа остаётся 1150 - 187 = 963.
6. Вернуть 963*0.001 = 0.963 единиц эфира отправителю.

И это всё! Если аккаунт получателя не является контракт-аккаунтом, шаг с выполнением кода контракта (шаг номер 5 в разобранном примере) просто будет пропущен. Комиссия в таком случае будет определяться лишь длиной транзакции (шаг номер 3), а прописывания информации (вроде значения `CHARLIE`) и вместе с тем успешной её пересылки происходить не будет. Дополнительно, заметим, что контракт-аккаунты, в точности как и контролируемые владельцами аккаунты, могут сами указывать `GASLIMIT` (это то же, что и `STARTGAS`, но название `GASLIMIT` кажется более удачным; в англоязычной документации используется название `STARTGAS`) для порождаемых ими сообщений, и если во время такого подвычисления отведённый ему газ закончится, откат (к предыдущему состоянию) происходит к моменту вызова этого сообщения в контракте, но не к моменту начала всего контракта. Таким образом, подобно транзакциям, контракты могут задавать предельно допустимое количество шагов в каждом порождаемом собой подвычислении. *(Только вот зачем? Всё равно же за все порождаемые сообщения платит автор первого сообщения, т.е. не контракт-аккаунт.)*

Выполнение кода

Код, понимаемый контрактами Эфириум, пишется на низкоуровневом stack-based байткод-языке, который мы назвали "код виртуальной машины Эфириум" (Ethereum Virtual Machine code) или "EVM-код". (На самом деле, по факту, пишется на одном из высокоуровневых языков и затем компилируется до EVM-кода, но мы сейчас обсуждаем последний.) Код состоит из последовательности байтов, и каждый байт представляет какую-то операцию. В общем случае, выполнение кода — это бесконечный цикл, состоящий из выполнения операции текущего байта и последующего увеличения номера обрабатываемого байта на 1; прерваться этот процесс может при достижении конца кода, ошибке в коде, или выполнении встретившейся в коде инструкции `STOP` / `RETURN`. Операции могут обращаться к трём типам мест для размещения информации:

- **Стек**, last-in-first-out контейнер, куда можно записывать и откуда можно извлекать значения
- **Память**, байтовый массив произвольного размера
- **Хранилище** контракта, информация в котором хранится парами "ключ-значение". В отличие от стека и памяти, которые очищаются после выполнения кода, в хранилище параметры могут находиться долговременно.

Код также имеет доступ к денежной сумме, отправителю и информации входящего сообщения, а также к информации в заголовке блока; код может вернуть в том числе байтовый массив на выходе.

Принцип выполнения нашего EVM-кода на удивление прост. Во время вычисления всё текущее состояние содержится в наборе `(block_state, transaction, message, code, memory, stack, pc, gas)`, где `block_state` — полное состояние, содержащее информацию о балансах и состояниях хранилищ всех аккаунтов. При каждом раунде выполнения берётся `pc`-ый байт `code` и выполняется инструкция, соответствующая этому байту (напомним, что EVM-код — байткод); каждая из инструкций по-своему действует на наш набор. Для примера, `ADD` достаёт два элемента из стека и возвращает туда их сумму, после чего уменьшает `gas` на 1 и увеличивает `pc` на 1; а, скажем, `SSTORE` берёт верхние два элемента стека и помещает второй элемент в хранилище контракта под адресом, равным первому элементу (нечто похожее было проделано в предыдущем разделе с информацией `[ 2, 'CHARLIE' ]`). Хотя существует масса способов оптимизировать Эфириум посредством динамической компиляции, простейшая реализация Эфириума занимает лишь пару сотен строк кода.

Блокчейн и майнинг

Блокчейн Эфириума во многом похож на блокчейн Биткойна, но не во всём. :) Главное различие архитектур блокчейна в том, что, в отличие от Биткойн, блоки Эфириума содержат не только список транзакций, но и последнее состояние системы (балансы и хранилища каждого из пользователей). Помимо этого, номер блока и его сложность также размещаются в блоке. Алгоритм валидации блока в Эфириум таков:

1. Проверить, что блок, на который наш ссылается как на предыдущий, существует и валиден.
2. Проверить, что таймштамп нашего блока больше, чем предыдущего, но не более чем на 15 минут.
3. Проверить валидность номера блока, его сложности, transaction root, uncle root and gas limit (различные низкоуровневые специфичные для Эфириум концепции).
4. Проверить валидность proof-of-work блока.
5. Пусть `S[0]` — начальное состояние балансов и хранилищ перед обсчётом транзакций блока.
6. Пусть `TX` — список транзакций в блоке, а всего транзакций `n`. Для всех `0... n-1` положим `S[i+1] = APPLY(S[i],TX[i])` (состояние `i+1` есть состояние `i`, к которому применили транзакцию `i`; именно эта команда применяет все транзакции в блоке). Если при применении некоторых транзакций возникает ошибка, или если количество потреблённого газа при вычислении (соответствующих транзакциям) контрактов в блоке превысило `GASLIMIT`, вернуть ошибку.
7. Самая последняя транзакция в блоке — выплата майнеру награды за нахождение блока. Только после этого происходит приравнивание `S_FINAL = S[n]`.

8. Проверить, совпадает ли `S_FINAL` со `STATE_ROOT`. Если да, блок валиден; в противном случае — нет.

Этот подход может показаться крайне неэффективным на первый взгляд, поскольку он предлагает в дополнение к списку транзакций хранить текущие состояния всех аккаунтов в каждом блоке, но в реальности эффективность должна быть сравнима с эффективностью Биткойна. И вот почему. Состояния хранятся в древовидной структуре, и после нахождения нового блока (с новыми транзакциями) лишь малая часть дерева должна быть изменена. Thus, in general, between two adjacent blocks the vast majority of the tree should be the same, and therefore the data can be stored once and referenced twice using pointers (ie. hashes of subtrees). Для этого используется специальный тип дерева — "дерево Патрисия", including a modification to the Merkle tree concept that allows for nodes to be inserted and deleted, and not just changed, efficiently. Скажем, наконец, о том, ради чего это затевалось: т.к. вся информация о состояниях содержится в последнем блоке, нет необходимости хранить всю историю блокчейна — если бы это удалось встроить в Биткойн, каждая полная нода занимала бы в 10-20 раз меньше места на жёстком диске.

Приложения

Поверх Эфириум, по-видимому, возможны три типа приложений. Первая категория — финансовые приложения, предоставляющие пользователям мощные способы управления контрактами и заключения контрактов на свои активы. Примеры таких приложений — подвалюты, финансовые деривативы, хеджирующие контракты, сберегательные кошельки, завещания, и даже некоторые виды контрактов трудоустройства. Вторая категория — полуфинансовые приложения, приложения с участием также немонетарных материальных благ; идеальный пример здесь — самоисполняемые награды за решение важных вычислительных задач. Третья категория приложений — нефинансовые приложения, такие как онлайн-голосование и децентрализованное управление.

Системы жетонов

Системы on-blockchain жетонов имеют самые разные приложения — от подвалют, представляющих активы (такие как доллары, золото) до акций компаний, жетонов, представляющих умную собственность, безопасных неподделываемых купонов, и даже жетонов без привязки к какой-либо стоимости (выступающих в данном случае как плюсики на форумах с кармой). Системы жетонов на удивление просто встраиваются в Эфириум. Дело в том, что и валюта, и система жетонов являются базой данных с одной операцией — отнять X единиц от состояния счёта A и прибавить X единиц к состоянию счёта B, если выполнены следующие пререквизиты: 1) A имеет как минимум X единиц перед началом транзакции 2) транзакция заявлена стороной A. Всё, что нужно для реализации системы жетонов — реализовать эту логику в контракте.

Простейший Serpent-код, осуществляющий систему жетонов, выглядит так:

```
from = msg.sender
to = msg.data[0]
value = msg.data[1]

if contract.storage[from] >= value:
    contract.storage[from] = contract.storage[from] - value
    contract.storage[to] = contract.storage[to] + value
```

Это буквально функция изменения состояния обычной банковской системы (уже описанная в этом документе чуть выше). Необходимо, конечно, в начало добавить ещё несколько строк кода, описывающих начальное распределение валюты и учесть ещё несколько моментов — в идеале, неплохо бы также добавить функцию, позволяющую сторонним контрактам запрашивать информацию о балансе произвольного адреса. Но это, в общем-то, и всё. Основанные на Эфириум системы жетонов, используемые как подвалюты, обладают опцией, которая недоступна метакойнам: комиссии можно платить напрямую в такой подвалюте. Это может быть реализовано контрактом, который осуществляет непрерывный аукцион, на котором происходит докупка эфира на израсходованную сумму за подвалюту. (В Биткойн нет возможности написать такой контракт — см. "финансовая слепота" здесь.)

Финансовые деривативы и хеджирование рисков

Финансовые деривативы — наиболее распространённое приложение "смарт-контрактов", и одно из простейших с точки зрения написания кода. Основная трудность при реализации финансовых контрактов в том, что большинство из них предполагает отсылку к курсу валюты (эфира), который должен браться из внешних источников; к примеру, более чем желанным был бы смарт-контракт, хеджирующий риски волатильности эфира (или другой криптовалюты) по отношению к доллару США, но для работы такой контракт должен в режиме реального времени узнавать курс ETH/USD. Простейший способ реализовать это — через контракт на "поток информации" от условного NASDAQ, который должен обладать такими (очевидно нужными) свойствами:

- этот внешний источник (условный NASDAQ) имеет право обновлять контракт при необходимости
- этот внешний источник (а с точки зрения Эфириум, это обычный контракт) предоставляет интерфейс, позволяющий другим контрактам переслать сообщение этому контракту и получить назад значение курса ETH/USD.

Теперь, когда у нас есть этот ключевой ингредиент, хеджирующий контракт мог бы выглядеть так:

1. Дождаться, пока A "вложит в контракт" 1000 единиц эфира.
2. Дождаться, пока B "вложит в контракт" 1000 единиц эфира.
3. Спросить у внешнего источника (контракта NASDAQ) курс ETH/USD, записать стоимость 1000 единиц эфира в долларах США в хранилище нашего контракта — пусть это $x.
4. Спустя 30 дней разрешить A или B "реактивировать" контракт: переслать эфира текущей (т.е. уже при новом курсе ETH/USD) стоимостью $x стороне A, и остальное — стороне B.

Такой контракт имел бы заметный успех в криптокоммерции. Одной из наиболее обсуждаемых проблем криптовалют является их волатильность — при всей безопасности и всём удобстве работы с цифровыми активами вряд ли кому-то из физических лиц и продавцов могла бы понравиться перспектива потерять 23% своих сбережений за день. До эпохи смарт-контрактов чаще всего предлагалось осуществлять хеджирование с помощью обеспечения криптоактивов привычными активами (отличие контракта хеджирования от цветных монет в том, что в парадигме цветных монет можно покрасить хоть 0.0001 BTC в цвет "Audi X6", т.е. цена не обязана быть рыночной; цветная монета — просто жетон, подтверждающий право владения активом). Мысль была такова: каждое "заслуживающее доверия" лицо может выпустить некоторое количество своей криптовалюты и предлагать остальным (оффлайново) одну единицу этой валюты в обмен на одну единицу общепринятого актива (золота, доллара) с обещанием при пересылке такой валюты обратно (т.е. по требованию) выдать общепринятые активы их исходному владельцу.

На практике же, однако, обещать — не значит жениться, и эмитентам доверять не приходится, а банковская инфраструктура также недостаточно развита (слишком враждебна), чтобы предоставлять такие хеджирующие сервисы. Альтернативой являются финансовые деривативы (такие как приведённый выше хеджирующий контракт). Здесь вместо эмитента, предоставляющего свою валюту в обмен на актив, децентрализованный рынок спекулянтов, ставящих (на примере нашего хеджирующего контракта) на повышение курса ETH/USD. В отличие от эмитентов, спекулянты не могут развести ручками и сказать, что мы не будем возвращать ваш актив — их средства хранит контракт, и на него они уже не имеют влияния. При обоих подходах происходит то, что нужно — про волатильность можно забыть, но при первом подходе (эмиссии монет некоторой конторой) приходится забыть ещё и про безопасность. Заметим, что второй подход всё же не полностью децентрализован, поскольку нуждается во внешнем источнике, поставляющем информацию о курсе ETH/USD. Но даже такой подход гораздо лучше подхода с эмитентами — он требует меньше инфраструктуры (в отличие от денег, эмиссия информации о курсе валюты не требует лицензий и является просто формой свободы слова) и уменьшает возможность мошенничества.

Системы идентификации и репутации

Первая альтернативная криптовалюта (=альткойн) в истории, Namecoin (Неймкойн), была попыткой использовать Биткойн-подобный блокчейн для создания децентрализованной системы регистрации имён (на самом деле была решена более общая задача децентрализованного создания и хранения пар "ключ-значение"). Наиболее обсуждаемый пример того, зачем это могло бы быть нужно — децентрализованная DNS-система как альтернатива существующей. (DNS-серверы отображают доменные имена наподобие "kitten.com" — или, в случае Неймкойн, "kitten.bit" — в IP-адрес сервера сайта; корневые DNS-серверы являются базовым элементом инфраструктуры Интернета.) Другие возможные применения — email-аутентификация и продвинутые системы репутации. Вот пример простого контракта, который осуществляет Неймкойн-подобную систему регистрации имён в Эфириум:

```
if !contract.storage[tx.data[0]]:
    contract.storage[tx.data[0]] = tx.data[1]
```

Этот контракт крайне прост: всё, что в нём говорится — что информация в сеть Эфириум может быть добавлена, но не может быть изменена или удалена. Каждый может зарегистрировать некоторое имя, и ему будет присвоено некоторое значение, а эта регистрация будет длиться вечно. Более замысловатый контракт регистрации имени включал бы в себя функционал, разрешающий другим контрактам обращаться к нему с запросами, а также интерфейс управления контрактом для "владельца" (т.е. первого субъекта, зарегистрировавшего имя, скажем, "george"), с помощью которого он мог бы изменить данные или передать контракт другому лицу. Более того, поверх вполне можно добавить функционал репутаций и нечто наподобие web of trust ("у нас есть общие долгосрочные контакты, значит, мне можно доверять").

Децентрализованное хранение файлов

В последние годы появилось немало стартапов, связанных с онлайн-хранением файлов. Наиболее известным таким стартапом является Dropbox, предлагающий своим пользователям загружать файлы в специальную папочку, которая после этого радостно синхронизируется с калифорнийскими серверами. Однако беглый взгляд на различные существующие решения показывает, что в данный момент рынок хранения файлов весьма неэффективен; на 20—200-гигабайтовом уровне уже не предлагаются бесплатные аккаунты, но ещё не действуют скидки для предприятий, что приводит к стоимости месячного хранения файлов, сравнимой со стоимостью всего жёсткого диска. Контракты Эфириум могут породить настоящий бум в экосистеме децентрализованного хранения файлов, где все желающие смогут зарабатывать небольшие суммы сдачей в аренду собственных жёстких дисков, а изобилие неиспользуемого сейчас дискового пространства будет приводить лишь к снижению стоимости хранения файлов в долгосрочной перспективе.

Ключевым ингредиентом здесь является то, что мы назвали "децентрализованный Dropbox-контракт". Он мог бы быть устроен так:

1. Информация разбивается на блоки, каждый блок шифруется, по каждому блоку строится дерево Мёркла.
2. Создаётся новый контракт, который делает вот что: каждые N блоков он берёт случайный индекс в дереве Мёркла (используя хэш предыдущего блока как источник случайных чисел) и переводит X эфира первому, кто предъявит УВП-подобное доказательство владения блоком под тем индексом.

Когда пользователь хочет обратно заполучить свой файл, он может использовать протокол микроплатежей (выплачивая, к примеру, 1 сабо за каждые 32 килобайта); наиболее выгодной тактикой для платящего причём будет запрашивать по первой цене не весь файл, а публиковать транзакции с тем же nonce каждые 32 килобайта, предлагая каждый раз чуть меньшую цену.

Несмотря на то, что кажется, что владельцу файла приходится полагаться на большое количество случайных хранящих его файл узлов, риск может быть сведён практически до нуля с помощью технологии разделения секрета. Из структуры описанного выше контракта видно, что если выплаты по контракту происходят, это и является доказательством того, что рассматриваемый узел действительно хранит отведённую ему информацию.

Децентрализованные автономные организации

Децентрализованной автономной организацией (ДАО) называется виртуальная организация, состоящая из конкретных членов или акционеров, обладающая таким свойством: при согласии определённой доли членов (например, 67%) решения (например, о трате средств из фонда организации или о модификации её программного кода) могут приниматься от лица всей организации. Члены коллективно решают, как организация должна распределять свои средства. Методы распределения могут быть самые разные: от зарплат и премий до много более экзотических механизмов наподобие выплат во внутренней валюте. Звучит похоже на обычные организации, но правоприменение здесь происходит только посредством блокчейн-технологии. К настоящему моменту в основном обсуждалась "капиталистическая" модель ДАО — "децентрализованная автономная корпорация" (ДАК) с выплачиваемыми акционерам дивидендами и свободно торгующимися акциями; одна из альтернатив, которую, возможно, следовало бы назвать "децентрализованное автономное сообщество", такова: голоса всех членов равноправны и для добавления/исключения члена 67% текущих членов должны с этим согласиться. В последнем случае, правда, необходимо как-то добиться того, чтобы один человек мог иметь только одно членство.

Обсудим то, как мог бы выглядеть код децентрализованной автономной организации. Первое, что приходит в голову — код, обладающий свойством самоизменяться (принимать поправки), если 2/3 членов с ними согласились. Тот факт, что код однажды созданного контракта нельзя изменить, на самом деле очень легко обойти: самоизменяемость де-факто достигается хранением кусков кода в разных контрактах, и хранением информации о том, какие куски вызывать, в (по определению) изменяющемся хранилище контракта. В простейшей реализации такого ДАО-контракта ("код ДАО") есть три типа транзакций, различаемых поданной на вход информацией:

- `[0,i,K,V]` — регистрирует предложение номер `i`, которое таково: заменить находящееся в хранилище под индексом `K` значение на величину `V`
- `[0,i]` — регистрирует голос в поддержку предложения номер `i`
- `[2,i]` — принимает предложение `i` (вносит изменение), если необходимая доля голосов набрана

Помимо реализации вышеупомянутых трёх пунктов, такой контракт также, по-видимому, должен

- вести публичный учёт всех изменений в хранилище контракта (т.е. в коде)
- список всех голосовавших за то или иное изменение
- список всех участников организации.

Более проработанная структура имеет возможность не только голосованием изменять код, но и голосовать за отправление транзакции или добавление/исключение члена, и может даже реализовывать делегирование голосов, подобно тому как это происходит в облачной демократии (а происходит там это так: А может передоверить своё право голоса В, и эта операция транзитивна, т.е. если А передоверил свой голос В, а В, в свою очередь, передоверил тот голос С, то С проголосует по голосу участника А). Делегирование голосов позволило бы "спихнуть" работу по добавлению/удалению членов на "специалистов", в случае недовольства действиями кого-то из которых каждый может передоверить свой голос другому члену организации. *(Против этого, впрочем, есть такой аргумент: если кто-то завладевает большинством голосов, и в какой-то момент рядовые члены становятся им недовольны и начинают передоверять свои голоса другому, то такой "специалист", пока у него есть большинство голосов, может удалять "отвернувшихся" от него членов организации. В общем, в точности всё, как внутренняя политика российских правящих кругов на сентябрь 2014.)*

Мы ещё не обсудили случай "децентрализованной корпорации", в котором у каждого аккаунта есть ноль или больше акций, и 2/3 акций необходимы для принятия решения. Полная её структура включала бы в себя функционал управления активами, возможность сделать предложение покупки/продажи акций, и возможность принимать эти предложения (а также следить за порядком их поступления). Делегирование голосов в стиле облачной демократии тоже реализуемо и привело бы к прозрачно и естественно формируемому "совету директоров".

Дальнейшие приложения

1. Сберегательные кошельки. Предположим, Алиса хочет сохранить свои сбережения, но опасается того, что потеряет или кто-либо украдёт её приватный ключ. Она заключает контракт с Бобом (который заменяет банк), в который помещает свой эфир. Контракт может иметь такой вид:

- В одиночку Алиса может забрать не более 1% средств в день.
- В одиночку Боб может забрать не более 1% средств в день, но Алиса может отправить транзакцию, отключающую эту возможность.
- Алиса и Боб по договорённости могут забрать любую часть средств.

Обычно 1% в день — это более чем достаточно для Алисы, и если Алиса захочет забрать больше, по договорённости с Бобом она сможет воспользоваться третьим пунктом. Если приватный ключ Алисы будет украден, она обратится к Бобу за перемещением средств на новый контракт.(*как Боб узнает, что к нему обращается уже не злоумышленник?*) Если она потеряет свой приватный ключ, в конечном итоге Боб получит все средства. Если Боб окажется подозрительным лицом, она отменит пункт, позволяющий ему забирать средства.

2. Страхование урожая. Каждый может создать контракт типа "финансовые деривативы", использующий сведения о погоде, а не о динамике стоимости какого-либо актива. Если краснодарский фермер покупает дериватив, выплаты по которому производятся обратно пропорционально осадкам в его регионе, тогда и в случае засухи фермер не будет страдать, и в случае достаточного количества дождей, очевидно, тоже.

3. Децентрализованный поток информации. Используя протокол SchellingCoin, можно даже создать децентрализованный источник верной информации, что особенно необходимо для контрактов на разницу. SchellingCoin работает так: N сторон предоставляют значения некоторой величины (например, курс ETH/USD), эти значения сортируются и образуют список, и авторы значений, оказавшихся в середине этого списка (25%-75%), получают выплаты. Таким образом, выгодно отвечать так же, как и большинство, откуда следует, что каждой из сторон выгодно (в отсутствие массового сговора) предоставить правильное значение нужной величины. Это даёт децентрализованный протокол, предоставляющий верные значения: курс ETH/USD, температуру в Берлине или даже результат конкретного трудного вычисления.

4. Сделки с гарантом с мультиподписью. Биткойн позволяет заключать контракты с мультиподписью, в случае которых средства можно расходовать только если предъявлено, к примеру, не меньше трёх из пяти подписавших контракт ключей. Эфириум допускает более замысловатые контракты — для примера, "4 из 5 могут потратить все средства, 3 из 5 могут тратить до 10% в день, 2 из 5 могут тратить до 0.5% в день". В дополнение к сказанному, мультиподпись Эфириума асинхронна — стороны могут подписывать транзакцию в разное время, и последняя подпись автоматически отправит её в сеть.

5. Облачные вычисления. Виртуальная машина Эфириум также позволяет создать вычислительную среду, пользователи которой смогут просить друг друга проделывать вычисления с одновременным предоставлением доказательств того, что в некоторых случайных точках вычисление производится корректно (вычисляющий "не срезает углы"). Это создаёт целый рынок облачных вычислений, участие в котором может принять каждый владелец ПК или выделенного сервера — а для того, чтобы вычисляющие узлы не жульничали, как уже было сказано, может использоваться проверка корректности в произвольных местах наряду с системой залогов. Описанная среда подходит не для всех типов заданий; задания, для которых необходимо интенсивное межузловое взаимодействие, к примеру, трудно выполнить на большом количестве анонимных узлов. Другие задания, впрочем, намного легче распараллелизовать; проекты наподобие SETI@home, folding@home и генетические алгоритмы с лёгкостью могут быть реализованы на такой платформе.

6. Пиринговые азартные игры. Любое количество протоколов пиринговых азартных игр, таких как Cyberdice, может быть встроено в блокчейн Эфириума. Простейший такой протокол — просто контракт на разницу с хэшем следующего блока, и могут быть созданы более продвинутые протоколы сервисов азартных игр с практически нулевыми комиссиями; при этом в силу объективности информации блокчейна в таких играх нет никакой возможности жульничества.

7. Рынки предсказаний. Имея SchellingCoin или другой способ получения правильных ответов, рынки предсказаний также достаточно легко внедрить. Снабжённые SchellingCoin рынки предсказаний могут стать первым мейнстримным применением футархии как протокола управления децентрализованными организациями.

8. Децентрализованные рынки на основе блокчейна, в основе которых лежат описанные нами в соответствующем разделе системы идентификации и репутации.

Размышления и разное

Реализация modified GHOST

Протокол GHOST ("Greedy Heaviest Observed Subtree") был предложен в декабре 2013. Мотивация, стоящая за GHOST: блокчейны с быстрым временем подтверждения транзакции в данный момент страдают от недостаточной безопасности ввиду большого количества "прокисших" блоков. Поясним, что имеется в виду. Так как блокам нужно сколько-то времени для того, чтобы распространиться по сети, если майнер A находит блок, а затем майнер B находит другой блок с таким же порядковым номером до того, как к нему распространится информация о находке майнера A, блок майнера B остаётся ненужным, не войдёт в блокчейн, и вычисления майнера B никак не помогут сообществу *(ну а в обычном случае майнер-неудачник просто будет вычислять хэши всё это время, пока к нему будет распространяться информация об обнаружении кем-то блока, и что?)*. Такие блоки мы и будем называть "прокисшими" (*англ. термин* - stale). Кроме того, этот эффект заметно усиливает тенденцию к централизации: если майнер A — это майнинг-пул, суммарная мощность которого составляет 30% мощности всей сети, а у майнера (майнинг-пула) B эта цифра составляет 10% мощности всей сети, A будет иметь риск найти прокисший блок 70% времени (т.к. лишь в 30% случаев A будет находить последний блок), а B, соответственно, будет иметь риск найти прокисший блок 90% времени. Таким образом, если интервал между блоками достаточно короткий, прокисших блоков будет много и они будут заметно влиять на майнинг, и майнер B в нашем примере будет многократно проигрывать майнеру A даже не только за счёт меньшей мощности, но также и из-за того, что будет находить гораздо больше прокисших блоков. Майнеры будут переходить на крупные пулы, и всё это приведёт к тому, что блокчейны с малым временем генерации блока будут майниться на 1-2 крупных пулах, которые будут полностью контролировать майнинг, после чего атака 51% неизбежна.

Сомполински и Зохар описали GHOST, который решает первую из этих проблем включением прокисших блоков в вычисление, определяющее самый длинный (с точки зрения количества proof-of-work) отросток блокчейна. Для того, чтобы решить проблему централизации, мы выходим за рамки GHOST-протокола и вводим выплаты за прокисшие блоки: прокисший блок получает 87.5% от того, что бы получил, если б был принят в блокчейн,

Это было сделано по нескольким причинам. Во-первых, полная версия протокола GHOST сильно бы усложнила вычисление того, какие дяди данного блока валидны. Во-вторых, полная версия GHOST при предлагаемой Эфириум системе компенсации за прокисшие блоки приводит к отсутствию стимула майнить главную цепь блокчейна; как известно ...., а не цепь возможного злоумышленника. Кроме того, одноуровневый GHOST приносит 80% пользы от полного GHOST-протокола, и процент прокисших блоков в нём при block time (среднего времени нахождения нового блока) 40 секунд таков же, как у Лайткойна (как известно, block time Лайткойна — 2.5 минуты). Однако мы будем чуть более консервативны при выборе block time и положим его равным 60 секунд (как у Праймкойна), поскольку верификация некоторых блоков занимает чуть больше обычного.

Комиссии

Представим себе, что выплат из ниоткуда (как это обычно происходит при майнинге) за нахождение блока нет. Происходящее в этом подразделе — ответ на вопрос "как обеспечить нормальную работу сети в такой ситуации". *(нужен ли этот абзац?)*

Поскольку для каждой попавшей в блокчейн транзакции участники сети (более точно — люди, пожелавшие хранить полные ноды*(только ли они?)*) загружают её на свой жёсткий диск и проверяют валидность, должен быть некоторый регуляторный механизм во избежание злоупотреблений, и типичный подход здесь — комиссии. В Биткойн комиссия устанавливается отправителем и может быть какой угодно, в том числе равной нулю, так что минимально допустимое значение комиссии оказывается динамическим параметром, который задаётся майнерами. (Заработок майнера, нашедшего последний (впрочем, и вообще любой) блок, складывается из награды за нахождение блока и комиссий всех транзакций этого блока. Предполагается, что когда награды за нахождение блока станут крайне низкими и затраты на (необходимый для проведения транзакций) майнинг не будут окупаться только посредством наград за нахождение блока, основную долю заработка майнера будут составлять именно комиссии, и транзакции с меньшими комиссиями будут включаться в блокчейн в последнюю очередь, что приведёт к тому, что какие-то комиссии люди всё-таки будут платить, устанавливая тем самым динамическую сумму предполагаемой комиссии.) (*может, просто сложность упадёт, потому что ИНВЕСТОРАМ это будет не нужно?*) Это свойство Биткойн-протокола вызвало широкое одобрение Биткойн-сообщества как "рыночное", позволяющее спросу и предложению на услуги майнеров определять цену. Проблема такого подхода в том, что проведение транзакций не слишком похоже на рынок, пусть это и привлекательно — представлять проведение транзакций как рынок майнинг-услуг; в реальности каждая транзакция, включённая майнером в блок, проводится каждой полной нодой сети, так что практически вся часть проведения транзакции производится людьми, не получающими от неё никакой прибыли. Это почти буквально т.н. трагедия общин — у держателей полных нод нет никакого экономического стимула этим заниматься.

Однако этот недостаток (при некоторых допущениях) магически исчезает. Аргумент тут такой. Предположим, что:

1. Пусть транзакция состоит из k операций и предлагает комиссию в kR майнеру, который включит её в блокчейн, где R задаётся отправителем; k и R более-менее известны майнеру заранее.
2. Себестоимость проведения операций для каждой ноды равна C.
3. Вычислительные мощности всех майнящих нод одинаковы; обозначим число майнящих нод за N. (Тогда вычислительная мощность каждой равна $1/N$ от полного хэшрэйта сети.)
4. Нет полных нод, которые не заняты в майнинге.

Майнер захочет включить в блок только те транзакции, полученная комиссия с которых превысит себестоимость проведения. Средняя сумма, которую майнер будет получать за блок, составит kR/N, т.к. вероятность нахождения блока с рассматриваемой транзакцией (да какого угодно блока) в нашей модели равна $1/N$. (Окей, более точно — матожидание награды равно kR/N.) Себестоимость же проведения транзакции равна просто kC. Таким образом, майнеры будут включать такие транзакции в блок, для которых $kR/N > kC \rightarrow R > NC$. Заметим, что R — устанавливаемая отправителем комиссия за одну операцию транзакции, и потому R — нижняя граница "пользы" от этой транзакции для отправителя. А NC — себестоимость проведения операции для всей сети. Полученное (при некоторых предположениях) неравенство $R > NC$ можно расшифровать так: при таких предположениях майнерам выгодно включать в блок только такие транзакции, польза от которых для автора (отправителя) больше, чем себестоимость её проведения для всей сети.

Однако наша модель всё же не очень похожа на реальность. И вот почему:

1. Себестоимость проведения транзакций для майнера, нашедшего блок и включившего её в него, всё-таки выше, чем себестоимость для всех остальных нод, которые занимаются её верификацией. Дело в том, что время, необходимое на верификацию, замедляет распространение блока в сети, что увеличивает вероятность того, что блок станет "прокисшим" и не принесёт никакой прибыли. (*понять смысл*)
2. Существуют (тысячи их!) полные ноды, которые не занимаются майнингом.
3. Вычислительные мощности майнеров распределены крайне неравномерно, и тенденция здесь, увы, в сторону дальнейшего увеличения "коэффициента Джини".
4. Спекулянты, боящиеся конкуренции политики и разного рода фанатики, стремясь нанести вред сети, вполне могут создать замысловатые контракты, себестоимость которых для авторов будет гораздо ниже, чем для остальных нод.

По причине (1) майнерам выгоднее включать в блок поменьше транзакций, причина (2) увеличивает `NC`; так что эти два эффекта хотя бы частично компенсируют друг друга. Пункты (3) и (4) являются известной проблемой; для того, чтобы пофиксить её, мы вводим плавающий лимит: ни один блок не может иметь больше операций, чем `BLK_LIMIT_FACTOR` умножить на долгосрочное экспоненциально изменяющееся среднее значение. Более детально, `blk.oplimit = floor((blk.parent.oplimit * (EMAFACTOR - 1) + floor(parent.opcount * BLK_LIMIT_FACTOR)) / EMA_FACTOR)`. `BLK_LIMIT_FACTOR` и `EMA_FACTOR` — константы, которые пока что предлагается положить равными 65536 и 1.5 соответственно, но эти значения, скорее всего, изменятся после более глубокого анализа.

Вычисление и Тьюринг-полнота

Крайне важно, что EVM Тьюринг-полна; это означает, что EVM-код может реализовать любое мыслимое вычисление, включая бесконечные циклы. EVM допускает два способа написать бесконечный цикл. Во-первых, можно использовать инструкцию `JUMP` (или `JUMPI 1 0` (`JUMP` при выполнении некоторого условия)), которая позволяет вернуться в одно из предыдущих мест кода. Во-вторых, контракты могут обращаться к другим контрактам, что потенциально может привести к рекурсии. Возникает естественный вопрос: могут ли недобросовестные пользователи парализовать работу майнеров и полные ноды, вводя их в бесконечный цикл? В computer science это известно как проблема остановки: Тьюринг в своё время доказал, что не существует общего алгоритма, который позволял бы определить, закончится ли когда-либо выполнение той или иной программы.

Как описано в этом разделе, наше решение — в том, чтобы автор транзакции сразу задавал максимально допустимое число шагов в вычислении (прикреплял к контракту некоторое количество эфира(?)), и если этот лимит превышен (газ кончился), вычисление прерывается, произведённые изменения откатываются, но комиссии остаются уплаченными (=весь потраченный газ уходит майнеру). С сообщениями та же история. Чтобы пояснить причины такого решения, рассмотрим следующие примеры:

- Злоумышленник создаёт контракт с бесконечным циклом, и затем пересылает транзакцию, активирующую этот контракт, майнеру. Майнер, проводя транзакцию, попадает в бесконечный цикл, и в какой-то момент газ, прикреплённый к этому контракту, заканчивается. Выполнение контракта на этом останавливается, но транзакция, активировавшая этот контракт, остаётся валидной и майнер радостно взимает со злоумышленника комиссию за каждый вычислительный шаг (более точно, весь потраченный на вычисление газ в Эфириум всегда идёт майнеру — в данном примере потрачен весь прикреплённый к контракту газ).
- Злоумышленник создаёт цикл такой длины, чтобы за время, которое майнер будет вычислять его до конца, было найдено ещё несколько блоков, и майнер не мог включить транзакцию в один из следующих блоков, чтобы получить комиссию. Однако злоумышленник должен при отправлении транзакции задать значение переменной `STARTGAS`, ограничивающей количество вычислительных шагов, так что в Эфириум майнер заранее знает, какой контракт может содержать сумасшедшее количество вычислительных шагов.
- Злоумышленник, увидев контракт наподобие `send(A,contract.storage[A]); contract.storage[A] = 0`, пересылает по такому адресу транзакцию с количеством газа, достаточным только для выполнения первого шага, но не для второго (т.е. забирает деньги, не давая балансу стать равным нулю). Автор контракта не должен волноваться о возможности таких атак, ведь если вычисление прерывается, произведённые изменения откатываются.
- Для работы финансовый контракт должен усреднять внешние данные (скажем, о курсе ETH/USD) по, скажем, девяти потокам от частных источников информации (для минимизации риска). An attacker takes over one of the data feeds, which is designed to be modifiable via the variable-address-call mechanism described in the section on DAOs, and converts it to run an infinite loop, thereby attempting to force any attempts to claim funds from the financial contract to run out of gas. Для защиты от подобного можно просто указать gas limit сообщения.

Альтернатива Тьюринг-полного языка — Тьюринг-неполный, где нет инструкций `JUMP` и `JUMPI`, и только одна копия каждого контракта может существовать в стеке в каждый момент времени. В такой ситуации описанная система комиссий и паранойя по поводу возможности рекурсии теряют актуальность, поскольку в отсутствие возможности устроить бесконечный цикл стоимость выполнения контракта будет ограничена сверху его размером (количество вычислительных шагов будет прямо пропорционально размеру контракта). Дополнительно, Тьюринг-неполнота — не то чтобы такое уж сильное ограничение; из всех рассмотренных нами контрактов пока что только один содержал "петлю" (инструкцию типа `JUMP`), и даже этой петли можно было избежать (хорошо звучит), 26 раз повторив одну строчку. Возникает вопрос — есть ли какая-то серьёзная необходимость вводить в Эфириум Тьюринг-полный язык, если он причиняет нам столько проблем? Оказывается, что Тьюринг-неполнота языка не приводит к решению всех этих проблем. Рассмотрим такой контракт:

```
C0: call(C1); call(C1);
C1: call(C2); call(C2);
C2: call(C3); call(C3);
...
C49: call(C50); call(C50);
C50: (run one step of a program and record the change in storage)
```

Перешлём кому-нибудь транзакцию с таким контрактом. Тогда для 51 транзакции наш контракт занимает 2^{50} вычислительных шагов. Майнеры могли бы пресекать такие логические бомбы, устанавливая максимально допустимое число вычислительных шагов для каждого контракта и не останавливая счётчик шагов в том числе когда в теле одного контракта выполняется другой контракт. Но что, по-видимому, майнерам всё-таки пришлось бы при таком подходе запретить — контракты, не вызывающие, а создающие другие контракты. Другая проблема здесь в том, что поле "адресат сообщения" — переменная, так что в общем случае, видимо, нельзя предсказать, какие другие контракты будет вызывать данный контракт. Так что в итоге имеем нечто поистине удивительное: Тьюринг-полный язык прост в использовании; отсутствие Тьюринг-полноты языка не приводит к тому, что возможность атаки бесконечным циклом на сеть отпадает сама; механизмы защиты от бесконечных циклов в случае Тьюринг-неполного языка таковы же, как в случае Тьюринг-полного. Поэтому почему бы не позволить себе радость пользоваться Тьюринг-полным языком? (:

Валюта и выпуск

Сеть Эфириум включает в себя свою собственную встроенную валюту *эфир*, введённую с двумя целями: 1) предоставить по-настоящему ликвидную валюту для обмена цифровыми активами 2) предоставить механизм оплаты стоимости транзакций. Для удобства и во избежание будущих споров и дискуссий (как это сейчас происходит с Биткойном при обсуждениях названий для миллибиткойна, микробиткойна и мельчайшей частички Биткойна) номиналы валют предлагается обозначить:

- 1: wei
- 10^3: lovelace
- 10^6: babbage
- 10^9: shannon
- 10^{12}: szabo
- 10^{15}: finney
- 10^{18}: ether

(*Полагаю, стоит объяснить названия.*)

Это расширенная версия долларов/центов, рублей/копеек, биткойн/сатоши. Мы ожидаем, что "ether" будет использоваться для обычных транзакций, "finney" — для микротранзакций, "szabo" и "wei" — для технических дискуссий о протоколе; оставшиеся номиналы могут понадобиться позже и не должны быть пока что включены в стандартный Эфириум-клиент.

Модель выпуска может быть следующей:

- Эфир будет выставлен на продажу по цене 1000-2000 единиц эфира за 1 BTC; на вырученные деньги будет профинансирована проделанная и дальнейшая разработка Эфириума, как это имело место с Мастеркойн и NXT. Для первых покупателей будут действовать скидки. Биткойны, полученные с продажи, будут полностью использоваться для выплаты зарплат и премий разработчикам, а также инвестированы в различные коммерческие и некоммерческие проекты в экосистеме Эфириум.
- 30% от полного количества проданного эфира будет оставлено организации, чтобы поощрить ранних участников, выплатить ETH-расходы перед первым блоком и в качестве долгосрочного резерва. Это количество распределено в соответствии с законом экспоненциального затухания; каждый месяц до 5.6% оставшегося фонда может быть распределено между разработчиками и другими лицами, принимавшими участие в развитии проекта; распределение этих средств начнётся в декабре.
- 30% от полного количества проданного эфира будет добываться майнерами в год, и так будет всегда.

Group	At launch	After 1 year	After 5 years
Currency units	1.3X	1.6X	2.8X
Purchasers	76.9%	62.5%	35.7%
Reserve spent pre-sale	5.77%	4.69%	2.68%
Reserve used post-sale	17.3%	14.1%	8.04%
Miners	0%	23.1%	53.6%

Как и в Биткойн, несмотря на линейный во времени выпуск валюты, показатель роста количества валюты (в процентах) со временем стремится к нулю

Можно выделить две главных особенности нашей модели выпуска:

1. наличие фонда, принадлежащего Эфириум-организации
2. существование вечно растущего количества денег (в отличие от ограниченного количества в Биткойн).

Как нам кажется, фонд необходим, и вот почему. Если его не создать (при этом уменьшив эмиссию до 0.231x для сохранения того же процента инфляции), полное количество эфира будет на 23% меньше, и каждая единица будет обладать на 30% большей стоимостью. Следовательно, при продаже будет раскуплено на 30% больше эфира, так что каждая единица станет иметь абсолютно ту же стоимость, что и раньше. Организация при этом получит в 1.3 больше BTC, нежели при модели с фондом. Таким образом, ситуация с отсутствием фонда *приводит в точности к тому же*, что и ситуация с фондом, с одной лишь разницей: организация получит только BTC, и будет слегка менее заинтересована в дальнейшем развитии валюты.

"Вечный" линейный рост валюты уменьшает риск инстамайна — сверхприбылей первых майнеров, и больших (по отношению к последующему поколению) прибылей текущих майнеров. Такая модель даёт последующим поколениям майнеров те же возможности, что есть у живущих сейчас — что, наверное, справедливо. В то же время, поскольку процент, на который растёт количество валюты (=процент инфляции), со временем стремится к нулю, резон хранить в ней свои сбережения будет. Также поскольку полное число монет, доступных владельцам, будет уменьшаться по невнимательности (выброшенный жёсткий диск, забытый пароль от зашифрованного кошелька, ..), и если предположить, что в среднем теряется 1% монет в год, количество монет в обращении стабилизируется на значении, равном годовой эмиссии, поделённой на процент потерь (к примеру, если теряется 1% всех монет в год,)

Централизация майнинга

Алгоритм майнинга в Биткойн устроен примерно так: майнеры вычисляют SHA256-хэши от числа (заголовок блока + nonce (number used once), nonce — это просто индекс перебора) миллионы и миллионы раз, пока кто-нибудь не обнаружит хэш, который будет как 16-ричное число меньше так называемого таргета (текущее значение таргета в 16-ричной форме). Однако такой алгоритм майнинга уязвим по отношению к двум формам централизации. Во-первых, в экосистеме майнинга доминируют специализированные устройства — АСИКи (ASIC: application-specific integrated circuits). Они разработаны специально для майнинга биткойнов и имеют в сотни раз бóльшие вычислительные мощности, нежели остальные устройства. Таким образом, майнинг биткойнов более не является абсолютно децентрализованной гонкой людей с равными возможностями, и требует миллионов долларов инвестиций в спецоборудование, если заниматься этим серьёзно. Во-вторых, биткойн-майнеры в большинстве своём не производят валидацию блока самостоятельно; они надеются на свои майнинг-пулы, которые предоставляют им заголовки блоков. Возможно, дело обстоит даже хуже: на апрель 2014 три самых крупных майнинг-пула косвенно контролируют примерно 50% вычислительных мощностей (сейчас всё ещё веселее, мощность пула ghash.io уже несколько раз превысила 51% мощности сети), что, правда, смягчается тем фактом, что майнеры могут переключиться на другие майнинг-пулы если пул или несколько пулов попробуют предпринять атаку 51%.

На данный момент мы собираемся решить эту проблему с помощью алгоритма, при котором майнеры получают случайный кусок данных о состоянии, обсчитывают несколько случайно выбранных транзакций из последних N блоков и возвращают хэш результата. У этого есть два важных плюса. Во-первых, контракты Эфириума могут включать в себя вычисления любого сорта, так что Эфириум-АСИК должен с необходимостью быть АСИКом для любого вида вычислений — т.е. более мощным процессором. Во-вторых, для майнинга необходим доступ ко всему блокчейну, что приводит к необходимости для майнеров хранить весь блокчейн и быть способными как минимум проверить каждую транзакцию. Это делает ненужным существование майнинг-пулов в их современном понимании; хотя они могут продолжать работу, пиринговые пулы, которые по определению децентрализованы, способны выполнять задачу обычных пулов ничем не хуже.

Эта модель не оттестирована, и сложности с использованием контрактов как майнинг-алгоритмов действительно могут быть. Одна интересная особенность этого алгоритма заключается в том, что она разрешает кому угодно "отравить колодец", вводя такие контрактов в блокчейн, которые способны сделать непригодными для их вычисления тот или иной АСИК. Производителям АСИКов было бы выгодно использовать эту возможность для атаки друг друга. Таким образом, решение, которое мы разрабатываем — скорее адаптивное экономико-социальное, нежели сугубо техническое.

Масштабируемость

Как и Биткойн, Эфириум страдает от того недостатка, что каждая транзакция должна быть проведена каждым узлом сети. Текущий размер биткойн-блокчейна — 15 Гб, и он растёт примерно на 1 Мб в час. Если бы сеть Биткойн производила 2000 транзакций в секунду, как Visa, то блокчейн бы рос на 1 Мб каждые три секунды (1 Гб в час, 8 Тб в год). Эфириум, по-видимому, будет испытывать те же проблемы, отягощённые наличием огромного количества приложений, действующих поверх блокчейна Эфириума, но на самом деле не всё так плохо, потому что в Эфириум полным нодам не нужно хранить историю блокчейна, достаточно лишь помнить текущее состояние (баланс и состояние контрактов(*мб что-то ещё?*)) каждого из пользователей.

Проблема большого размера блокчейна, конечно, в риске централизации. Если размер блокчейна возрастает до 100 Тб, лишь малое количество бизнесов будут держателями полных нод, а простые пользователи будут держать "лёгкие" УВП-ноды. В такой ситуации владельцы полных нод могли бы вступить в сговор (например, изменить выплату за найденный блок, выдать самим себе BTC (*полностью понять, что они могли бы сделать и чего не могли*)). "Лёгкие" ноды не смогут обнаружить такой вид мошенничества немедленно. Конечно, хотя бы один владелец полной ноды скорее всего честен, и спустя несколько часов информация о мошенничестве утечёт через условный Реддит, но это будет слишком поздно: простые пользователи должны будут организовывать свою работу по занесению конкретных блоков в чёрные списки, массивную и, вероятно, невыполнимую работу по координации такой же сложности, как отбить атаку 51%. В ситуации с биткойном это проблема, но есть модификация блокчейна, предложенная Петером Тоддом, которая смягчает эту сложность.

В ближайшем будущем Эфириум собирается использовать две дополнительные стратегии, чтобы преодолеть эту проблему. Во-первых, из-за того, что майнинг-алгоритмы основаны на блокчейне, каждый майнер должен будет держать полную ноду => количество полных нод будет не меньше, чем количество майнеров. Во-вторых, что более важно, мы планируем включить промежуточный state tree root в блокчейн после проведения каждой транзакции. Даже если валидация блоков недостаточно децентрализована, пока хотя бы один честный узел занимается верификацией, проблемы централизации можно избежать с помощью верификационного протокола. Если майнер публикует невалидный блок, этот блок либо имеет неправильный формат, либо состояние `S[n]` некорректно. Поскольку начальное состояние — `S[0]` — корректно, существует какое-то первое некорректное состояние `S[i]` (при корректном `S[i-1]`). Верифицирующий узел предоставит индекс `i` вместе с proof-of-invalidity, состоящим из подмножеств узлов дерева Патрисия, пытающихся произвести `APPLY(S[i-1],TX[i]) -> S[i]`. Узлы смогут использовать эти части дерева Патрисия для выполнения нужного куска вычисления, и убедиться, что получившееся `S[i]` не совпадает с предъявленным майнером.

Возможна более замысловатая атака: недобросовестные майнеры могут публиковать неполные блоки, так что полной информации, которая могла бы показать валидность/невалидность блока, попросту не существует. Решением здесь является challenge-response протокол: verification nodes issue "challenges" in the form of target transaction indices, and upon receiving a node a light node treats the block as untrusted until another node, whether the miner or another verifier, provides a subset of Patricia nodes as a proof of validity.

Заключение

Эфириум-протокол изначально начинался как улучшенная версия криптовалюты, предоставляющая дополнительные возможности, такие как гарант-сервис на блокчейне, задание предельно допустимой для снятия суммы, финансовые контракты, рынки азартных игр и подобное посредством высокоуровневого языка программирования. Эфириум-протокол не поддерживает какое-либо из "приложений" напрямую, но существование Тьюринг-полного языка программирования означает, что контракты могут быть составлены совершенно произвольно. Что особенно вдохновляет, Эфириум — это гораздо больше, чем криптовалюта. Протоколы децентрализованного хранения файлов, децентрализованных вычислений и децентрализованных рынков предсказаний, а также десятки других концепций, имеют потенциал существенно увеличить эффективность IT-индустрии. Разумеется, есть и значительное количество приложений, не имеющих никакого отношения к деньгам.

Концепция произвольности функции изменения состояния, включённая в Эфириум-протокол, предоставляет платформу с уникальным потенциалом; в отличие от узкоспециализированных проектов — скажем, в области шифрования информации, азартных игр или финансов — Эфириум имеет крайне широкую специализацию по архитектуре, и мы верим, что Эфириум прекрасно подходит в качестве базовой платформы для финансовых и нефинансовых протоколов ближайших лет.

Заметки и дальнейшее чтение

Заметки

1. Въедливый читатель может заметить, что на самом деле Биткойн-адрес является хэшем публичного ключа, но не самим публичным ключом. Однако называть хэш публичного ключа публичным ключом, можно сказать, уже стало частью криптографической терминологии. Криптография, используемая в Биткойн, может рассматриваться как конкретный алгоритм цифровой подписи, где публичный ключ состоит из хэша ECC (elliptic curve cryptography)-публичного ключа, цифровая подпись — из ECC-публичного ключа и ECC-подписи, и алгоритм верификации включает в себя сверку ECC-публичного ключа в подписи с хэшем ECC-публичного ключа как с публичным ключом, и затем сверку ECC-подписи с ECC-публичным ключом.

2. Технически — среднее значение по 11 предыдущим блокам.

3. С точки зрения системы, и 2, и "CHARLIE" — числа; with the latter being in big-endian base 256 representation. Числа лежат в интервале от 0 до 2^{256}-1 (концы включаются).

Дальнейшее чтение

1. Внутренняя стоимость: http://bitcoinmagazine.com/8640/an-exploration-of-intrinsic-value-what-it-is-why-bitcoin-doesnt-have-it-and-why-bitcoin-does-have-it/
2. Умная собственность: https://en.bitcoin.it/wiki/Smart_Property, http://bitnovosti.com/2014/06/10/ponyatie-umnoi-sobstvennosti/
3. Смарт-контракты: https://en.bitcoin.it/wiki/Contracts
4. B-money: http://www.weidai.com/bmoney.txt
5. Reusable proofs of work: http://www.finney.org/~hal/rpow/
6. Secure property titles with owner authority: http://szabo.best.vwh.net/securetitle.html
7. Манифест Биткойн: http://bitcoin.org/bitcoin.pdf
8. Namecoin: https://namecoin.org/
9. Zooko's triangle: http://en.wikipedia.org/wiki/Zooko's_triangle
10. Манифест "цветных монет": https://docs.google.com/a/buterin.com/document/d/1AnkP_cVZTCMLIzw4DvsW6M8Q2JC0lIzrTLuoWu2z1BE/edit
11. Манифест Мастеркойн: https://github.com/mastercoin-MSC/spec
12. Децентрализованные автономные корпорации, Bitcoin Magazine: http://bitcoinmagazine.com/7050/bootstrapping-a-decentralized-autonomous-corporation-part-i/
13. Упрощённая верификация платежей: https://en.bitcoin.it/wiki/Scalability#Simplifiedpaymentverification
14. Деревья Мёркла: http://en.wikipedia.org/wiki/Merkle_tree
15. Деревья Патрисия: http://en.wikipedia.org/wiki/Patricia_tree
16. GHOST: http://www.cs.huji.ac.il/~avivz/pubs/13/btc_scalability_full.pdf
17. StorJ and Autonomous Agents, Jeff Garzik: http://garzikrants.blogspot.ca/2013/01/storj-and-bitcoin-autonomous-agents.html
18. Mike Hearn on Smart Property at Turing Festival: http://www.youtube.com/watch?v=Pu4PAMFPo5Y
19. Ethereum RLP: https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-RLP
20. Ethereum Merkle Patricia trees: https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-Patricia-Tree
21. Peter Todd on Merkle sum trees: http://sourceforge.net/p/bitcoin/mailman/message/31709140/