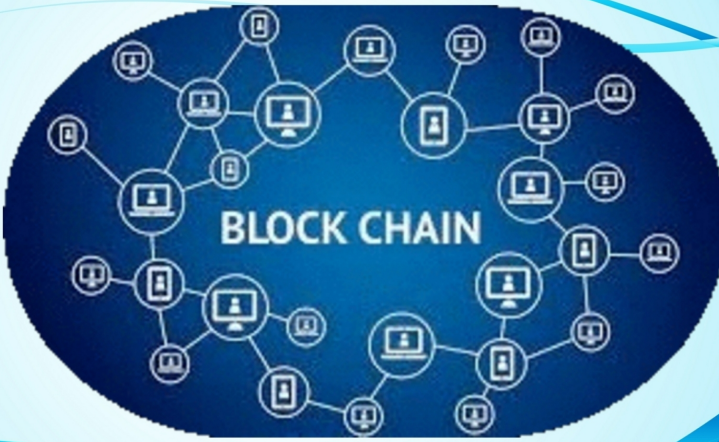
Введение в технологию Блокчейн



Тимур Машнин

18+

Тимур Сергеевич Машнин

Введение в технологию Блокчейн

http://www.litres.ru/pages/biblio_book/?art=66929618
SelfPub; 2021

Аннотация

Эта книга познакомит вас с технологией блокчейн, которая позволяет осуществлять передачу и хранение цифровых активов децентрализованным способом. Вы получите понимание и знание базовых концепций технологии блокчейн, познакомитесь с методами разработки приложений для блокчейн сетей. Эта книга объясняет основные компоненты блокчейна, такие как транзакция, блок, заголовок блока и цепочка блоков, операции блокчейна, такие как верификация, валидация и достижение консенсуса, а также алгоритмы, лежащие в основе блокчейна.

Содержание

Тимур Машнин
Введение в технологию Блокчейн

Введение в криптографию и криптовалюты

Начнем мы свое обсуждение с криптографических хеш-функций. Мы поговорим о том, что они собой представляют, и каковы их свойства. А потом мы обсудим их приложения.

Итак, криптографическая хэш-функция является математической функцией. И она обладает рядом свойств.

Криптографические хеш-функции — это выделенный класс хеш-функций, который имеет определенные свойства, делающие его пригодным для использования в криптографии.

Хеширование — преобразование массива входных данных произвольной длины в (выходную) битовую строку установленной длины, выполняемое определённым алгоритмом.

Функция, воплощающая алгоритм и выполняющая преобразование, называется «хеш-функцией» или «функцией свёртки».

Прежде всего, хеш-функция может принимать любую строку любого размера как входной параметр. И хеш-функция производит вывод строки фиксированного размера, мы будем использовать 256 бит, потому что это делает биткойн.

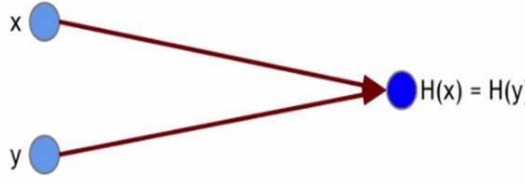
И хеш-функция должна быть эффективно вычисляемой, обрабатывая заданную строку за разумный промежуток времени.

Также хэш-функция должна быть криптографически безопасна.

В частности, функция не должна иметь коллизий или конфликтов, она должна иметь свойство скрытия, и она должна быть головоломкой.

Первое свойство, которое нам нужно иметь для криптографической хеш-функции это то, что она не содержит кон-

фликтов.



И это означает, что нельзя найти две разные строки с одинаковой хэш-функцией.

Теоретически коллизии существуют, так как на вход вы берете строку любой длины, а на выходе получаете строку 256 бит.

Вопрос в том, могут ли эти коллизии быть найденными обычными людьми, использующими обычные компьютеры?

С точки зрения вероятности, если взять 2 в 130 степени случайно выбранных входных строк, то с вероятностью 99.8 %, по крайней мере, две из них будут конфликтовать, независимо от используемой хэш функции.

Но, конечно, проблема в том, что вы должны вычислить

хэш функцию 2 в 130 степени раз.

И это, конечно, астрономическое число.

Не существует хеш функции, для которой было бы доказано, что она свободна от конфликтов.

Просто эти конфликты трудно найти, и мы принимаем то, что мы используем хэш функцию, свободную от конфликтов.

Если мы можем предположить, что у нас есть хеш-функция, свободная от коллизий, тогда мы можем использовать результаты этой хэш-функции как дайджест сообщений.

И я имею в виду следующее.

Если мы знаем, что x и y имеют одинаковый хеш, тогда можно с уверенностью предположить, что x и y одинаковы. И это позволяет нам использовать хэши как своего рода дайджест сообщений.

Предположим, например, что у нас есть большой файл.

И мы хотели бы уметь распознавать, будет ли другой файл таким же, как этот файл.

Один из способов сделать это, это сохранить весь большой файл. И затем, когда мы получим другой файл, просто сравнить их.

Но поскольку у нас есть хэш функция и хэши файлов, которые, как мы считаем, свободны от конфликтов, более эффективно просто запомнить хэш исходного файла.

Затем, если кто-то показывает нам новый файл и утверждает, что это то же самое, мы можем вычислить хэш этого нового файла и сравнить хеши.

Если хеши одинаковы, мы делаем вывод, что файлы одинаковые.

И это дает нам очень эффективный способ запомнить то, что мы видели раньше, так как хэш невелик, это всего лишь 256 бит, в то время как исходный файл может быть очень большим.

Второе свойство, которое мы хотим от хэш-функции, состоит в том, что она является скрывающей.

Если у нас есть результат хэш-функции, тогда нет никакого способа определить, что из себя представляет вход хэш функции.

Это работает, когда вход хэш функции представляет собой огромный набор различных вариантов, так что нельзя простым перебором, вычисляя хэши, найти соответствие хэша и определенного входа.

Если же у нас набор входных значений небольшой, мы можем решить эту проблему с помощью соединения нашего входного значения со значением, которое было выбрано из очень большого набора значений.

$$H(r \mid x)$$

Таким образом, хэш функция H (r | x) означает взять все биты r и поместить после них все биты x.

Если r – случайное значение, выбранное из широкого распределения, то, учитывая H (r | x), невозможно найти x.

Таким образом хэш r соединенного с x, будет скрывать x.

Теперь давайте посмотрим на применение этого скрывающего свойства.

Предположим, что мы берем число, заворачиваем его в конверт, и помещаем его на стол, где каждый может увидеть конверт. Но пока вы не открыли конверт, это число является секретом.

Позже вы можете открыть конверт и выдать значение.

Мы хотим сделать это в цифровом смысле. Например, вы можете передать сообщение.

И эта передача выдаст два значения, com и key.

Подумайте о com как о конверте, который вы собираетесь положить на стол, и ключе как о секретном ключе для разблокировки конверта.

Затем вы позволяете кому-то проверять, учитывая com, key и сообщение, что этот конверт, ключ и сообщение действительно идут вместе.

И эта проверка вернет истину или ложь.

Мы помещаем сообщение в конверт и передаем сообщение.

И эта передача возвращает конверт и ключ, а затем мы публикуем конверт.

Позже, чтобы открыть конверт, мы должны опубликовать ключ.

И тогда кто-то может использовать этот конверт, ключ и сообщение, чтобы проверить валидность открытия конверта.

Таким образом невозможно будет подменить сообщение в конверте.

```
(com, key) := commit(msg)
match := verify(com, key, msg)
```

Чтобы реализовать эту схему, мы генерируем случайное значение 256 бит и назовем его ключом.

И тогда мы вернем хэш ключа, соединенного с сообщением.

Затем, кто-то может проверить целостность сообщения, вычислив хэш ключа и сообщения и сравнив это значение с переданным значением com.

com – это хэш ключа, соединенного с msg.

И если есть этот хэш, по нему невозможно узнать само сообщение.

Это то самое свойство скрытия, о котором мы говорили раньше.

Ключ был выбран случайным 256-битным значением.

```
commit(msg) := ( H(key | msg), key )
                    where key is a random 256-bit value
verify(com, key, msg) := ( H(key | msg) == com )
```

И поэтому свойство сокрытия означает, что, если мы возьмем сообщение, и поставим перед ним что-то, что было выбрано из очень большого распределения, как в данном случае случайное 256-битное значение, тогда невозможно найти сообщение, как исходя из самого хэша, так и простым перебором возможных сообщений.

Одновременно мы получаем и отсутствие конфликтов, используя такую схему.

Невозможно будет найти два разных сообщения с одинаковым таким хэшем.

Используя все эти свойства хэша, мы можем сделать приложение – поиск пазла, математическую задачу, которая требует больших вычислений.

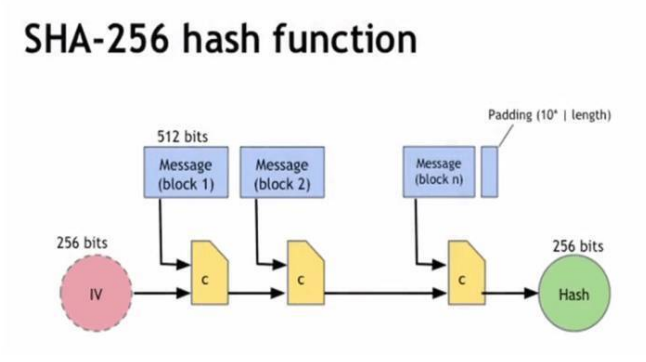
$$H(id \mid x) \in Y.$$

Используя id из большого распределения, нужно найти x, чтобы хэш id и x попадал в заранее определенное распределение y.

Если мы хотим создать головоломку, которую трудно решить, мы можем сделать это таким образом, генерируя идентификаторы головоломок в случайном порядке.

И мы будем использовать это позже, когда мы поговорим о майнинге биткойнов.

Это своего рода вычислительная загадка, которую мы будем использовать.



Существует множество хэш функций, но функция, которую использует биткойн, это функция, которая называется SHA-256, и она работает следующим образом.

Она принимает сообщение, которое вы хешируете, и она разбивает его на блоки размером 512 бит.

Сообщение не обязательно кратно размеру блока, поэтому мы должны добавить в конце дополнение. И это дополнение будет состоять из, в конце дополнения, поля длиной 64 бит, которое является длиной сообщения в битах.

И затем до этого находится один бит, за которым следует некоторое количество нулевых бит.

И вы выбираете число нулевых бит, чтобы выйти точно в конец блока.

После того как вы разбили сообщение на блоки, вы начи-

наете вычисление.

Вы начинаете с 256-битного начального значения и берете первый блок сообщения.

Затем вы берете эти 768 полных битов, и обрабатываете специальной функцией сжатия, которая на выходе дает 256 бит.

Вы берете полученные 256 бит и следующие 512 бит сообщения, снова пропускаете через функцию сжатия и так далее, пока не обработаете все блоки сообщения.

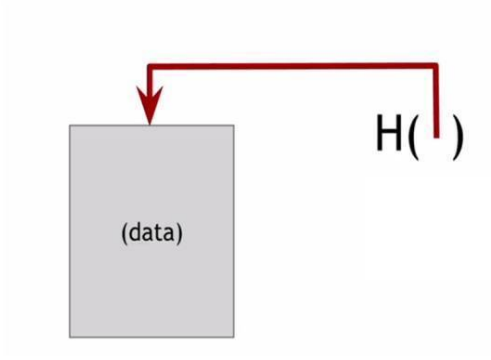
Таким образом вы получите хэш, как 256-битное значение.

И нетрудно показать, что если эта функция сжатия, C, является свободной от коллизий, то вся эта хэш-функция также будет свободна от коллизий.

Хэш указатели и структуры данных

Далее мы поговорим о хэш указателях и их применении.

Хэш указатель – это вид структуры данных, которая указывает где хранится некоторая информация.

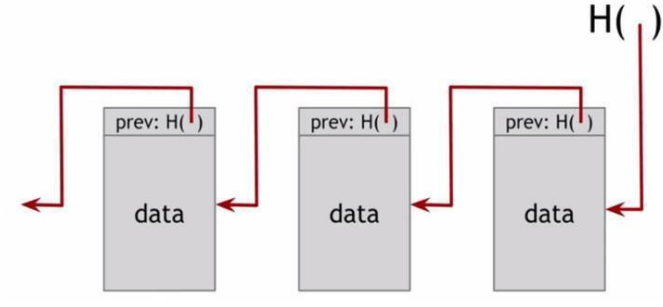


И в которой вместе с указателем хранится криптографический хэш самой информации.

Поэтому, если обычный указатель дает способ получения информации, хеш-указатель позволяет не только получить информацию, но и также проверить, что эта информация не изменилась.

И мы можем использовать хэш указатели для создания всех видов структур данных.

Здесь ключевая идея, использовать любую структуру данных, связанные списки или двоичное дерево поиска или что-то вроде этого, и реализовать это с помощью хеш-указателей.



Например, здесь есть связанный список, который мы построили с помощью хэш-указателей.

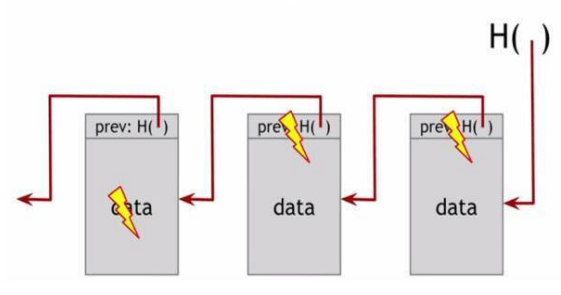
И это структура данных, которую мы собираемся назвать цепочкой блоков.

Так что, по сравнению с обычным связанным списком, где у вас есть серия блоков, и каждый блок имеет данные, а также указатель на предыдущий блок в списке, здесь указатель на предыдущий блок заменяется на хэш-указатель, ко-

торый хранит указатель на предыдущий блок и хэш всего содержимого блока.

Эта структура не только позволяет хранить данные, но и защищать их.

Теперь, что произойдет, если кто-то изменит данные в блоке.



Мы это легко обнаружим, сравнив хэш указатель и данные блока.

Если же кто-то изменит и хэш-указатель предыдущего блока, тогда возникнет несогласованность с хэш-указателем следующего блока, так как хэш-указатель хранит хэш не только самих данных, но содержимого всего блока.

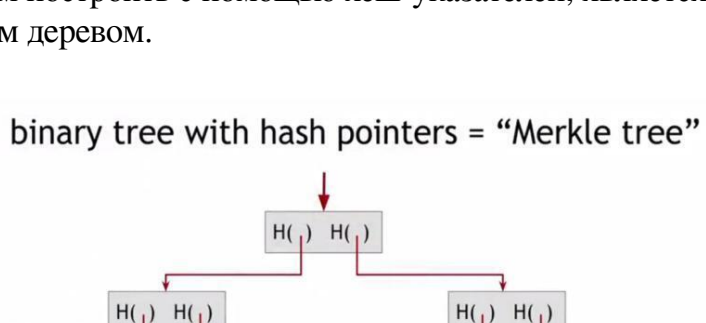
Поэтому злоумышленнику придется изменить хэш-указа-

тели всей цепочки до конца.

Таким образом, один хэш-указатель защищает весь список от этого блока и до конца.

Начальный блок такого списка называется блоком генезиса.

Теперь, еще одна полезная структура данных, которую мы можем построить с помощью хеш-указателей, является двоичным деревом.



Идея в том, что у нас есть куча блоков данных, которые показаны внизу.

И используя пары блоков, мы строим структуры данных с двумя хеш-указателями, по одному на каждый из этих блоков.

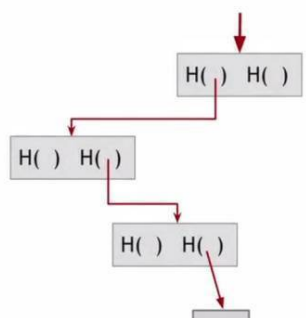
Затем мы переходим на другой уровень, и здесь этот блок будет содержать хэш-указатель этих двух детей. И так далее, вплоть до корня дерева, единственный хэш которого мы и запоминаем.

И мы можем быть уверены, что данные не будут подделаны.

Потому что злоумышленнику придется изменять хэши на всех уровнях, пока он не доберется до вершины дерева, но здесь он не сможет изменить хэш, так как мы его запомнили.

Таким образом, мы защищаем всю структуру данных, просто запомнив один хэш.

Теперь еще одна приятная особенность деревьев Merkle заключается в том, что в отличие от ранее рассмотренной цепочки блоков, если кто-то хочет доказать нам, что конкретный блок данных является членом этого дерева Merkle, все, что им нужно, это показать эти данные.



Мы вычисляем хэш этих данных и поднимается вверх до корня дерева, так как мы помним только корень дерева, проверяя соответствие хэш-указателям.

И это занимает около log n элементов.

Поэтому при очень большом числе блоков данных в дереве Merkle мы можем проверить членство данных за относительно короткое время.

Таким образом, деревья Меркле имеют преимущества в том, что даже если дерево содержит много элементов, нам просто нужно запомнить хеш корня дерева, который составляет всего 256 бит.

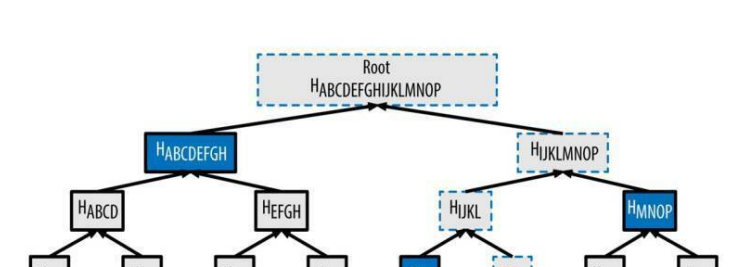
И мы можем проверить принадлежность к дереву за логарифмическое время.

Также есть вариант Merkle дерева – это сортированное де-

рево Merkle.

В этом дереве мы берем блоки данных внизу и сортируем их в некотором порядке.

Алфавитном, лексикографическом, числовом порядок или каком-либо другом порядке.



И как только мы отсортировали дерево Merkle, мы можем доказать, что конкретный блок не находится в дереве Merkle.

Мы можем сделать это, просто указав путь к элементу, который находится непосредственно перед местом, где этот элемент должен быть, и сразу после того, где он будет.

И тогда мы можем доказать, что оба эти элемента находятся в дереве Merkle последовательно.

И поэтому между ними нет места для элемента, который

мы ищем.

Цифровые подписи

Далее мы поговорим о цифровых подписях.

Это второй криптографический примитив наряду с хеш-функциями, которые нам нужны в качестве строительных блоков для обсуждения криптовалюты.

Итак, цифровая подпись – это как подпись на бумаге только в цифровой форме.

И что это значит, что мы хотим от подписи? Это две вещи.

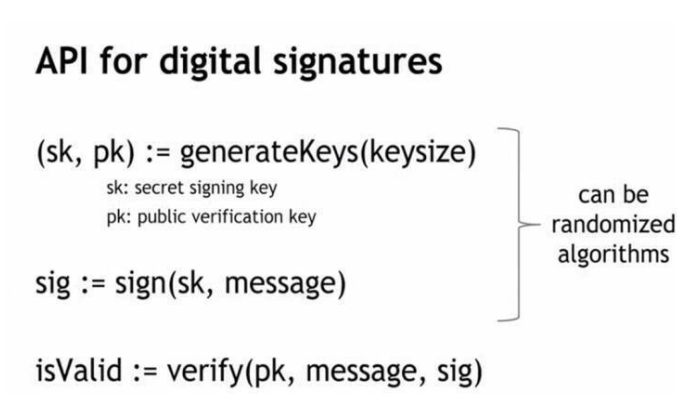
Во-первых, точно так же, как и для бумажной подписи, для цифровой подписи, только вы можете сделать свою подпись, но любой, кто видит вашу подпись может подтвердить, что она действительна.

И тогда вторая вещь, которую вы хотите, заключается в том, что подпись привязана к определенному документу.

Чтобы кто-то не мог взять вашу подпись с одного документа и приклеить ее на другой документ, потому что подпись – это не просто подпись.

Это означает ваше согласие или одобрение конкретного документа.

Теперь, как мы можем построить это в цифровой форме с использованием криптографии?



Вот API интерфейс для цифровых подписей.

Здесь есть три операции, которые мы должны делать.

Первое, нам нужно иметь возможность генерировать ключи, и поэтому у нас есть операции generateKeys, которая принимает размер ключа и создает два ключа, sk и pk.

Ключ sk будет секретным ключом подписи, это информация, которую вы держите в секрете и которую вы используете для создания своей подписи.

И ключ pk является общедоступным ключом проверки, который вы даете всем и который любой может использовать для проверки вашей подписи, когда они ее видят.

Вторая операция, это операция подписи.

Операция подписи берет секретный ключ подписи и сообщение, на котором вы хотите поставить свою подпись.

И она возвращает значение, которое является подписью и которое представляет собой лишь несколько бит, представляя вашу подпись.

И затем, третья операция – это проверка, которая берет то, что утверждается как правильная подпись, и подтверждает, что это действительно так или нет.

Эта операция принимает открытый ключ подписывающего лица, сообщение, к которому применена подпись, и принимает саму подпись.

И операция просто возвращает «да», если подпись действительна, или «нет», если она не действительна.

Таким образом, есть три операции, три алгоритма, которые составляют схему подписи.

Причем, первые два алгоритма могут быть рандомизированными алгоритмами.

Проверка всегда будет детерминированным алгоритмом.

Метод generateKeys должен быть рандомизированным, потому что он должен генерировать разные ключи для разных людей.

И подписи также должны отличаться для разных сообщений.

Подписи должны удовлетворять следующим двум требованиям.

Прежде всего, действительные подписи должны пройти проверку.

Если подпись действительна, т. е. если я подпишу сооб-

щение с моим секретным ключом, и, если кто-то затем позже попытается проверить ее, используя мой открытый ключ и то же самое сообщение, подпись будет корректно проверяться.

Второе требование, это то, что невозможно подделать подписи.

То есть, злоумышленник, который знает ваш открытый ключ, ключ проверки, и может видеть подписи на некоторых других сообщениях, не сможет подделать вашу подпись на каком-либо другом сообщении.

На практике существует ограничение на размер сообщения, который вы можете подписать, поскольку реальные схемы работают с битовыми строками ограниченной длины.

Поэтому используется хэш сообщения, а не сам текст сообщения.

Таким образом, сообщение может быть действительно большим, но хэш будет только 256 бит.

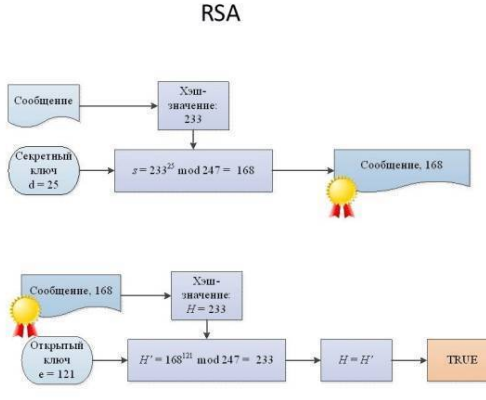
И поскольку функции хэша не имеют коллизий, хэш сообщения безопасно использовать в качестве входного сигнала для схемы цифровой подписи, а не само сообщение.

И, кстати, забавный трюк, который мы будем использовать позже, заключается в том, что вы можете подписать хэш-указатель.

И если вы подписываете хэш-указатель, то подпись покрывает или защищает всю структуру, а не только сам указатель хеширования, но и все, на что он указывает.

Например, если вы подписываете хеш-указатель, который

был в конце цепочки блоков, результатом будет то, что вы эффективно подписываете весь контент этой цепочки блоков.



Это пример цифровой подписи с использованием алгоритма RSA.

Биткойн использует определенную схему цифровой подписи, которая называется ECDSA.

Это алгоритм эллиптической кривой Elliptic Curve Digital Signature Algorithm.

И это стандарт правительства США.

Мы не будем вдаваться во все детали того, как работает ECDSA.

Он полагается на сложную математику.

И поверьте мне, вы не хотите видеть все детали того, как это работает. Поэтому мы это пропустим.

Одна вещь, которую я хочу заметить, это то, что для ECDSA важна хорошая рандомизация. Хорошая случайность особенно важна для ECDSA.

Если вы используете плохую рандомизацию для генерации ключей или даже подписи, вы можете дать доступ к своему секретному ключу.

Это свойство ECDSA, что, если вы используете плохую случайность, создавая подпись с использованием совершенно хорошего ключа, это может дать доступ к приватному ключу.

Поэтому вы должны быть особенно внимательны к этому на практике. Это распространенная ошибка.

Теперь, если мы возьмем публичный ключ, общедоступный ключ проверки цифровой подписи, то мы можем приравнять его к идентичности.

То есть, идентифицировать с помощью него личность человека, или действие, или систему.

Поэтому, если вы видите подпись сообщения, верифицированную или подтвержденную с помощью публичного ключа, тогда вы можете думать об этом как о том, что публичный ключ представляет сообщение.

Вы можете думать о публичном ключе, как о своего рода участнике в системе, потому что за публичным ключом находится приватный ключ, который может принадлежать только

одной личности.

Если мы собираемся рассматривать публичные ключи как личности, одним из последствий этого является то, что вы можете создать новую идентичность с помощью создания новой пары случайных ключей – секретный ключ и публичный ключ, используя операцию генерации ключей.

И публичный ключ будет публичным именем личности, хотя на практике используется хэш из публичного ключа, так как публичные ключи большие.

Вы контролируете личность, потому что только вы знаете секретный ключ, и, если вы сгенерировали ключи случайным образом, тогда никто не может знать, кто вы.

Вы можете создавать свежие идентичности, которые выглядят случайными.

Это подводит нас к идее децентрализованного управления идентификацией, где вместо того, чтобы иметь центральное место, куда вы должны пойти, чтобы зарегистрироваться как пользователь в системе, здесь вам не нужно получать имя пользователя.

Вам не нужно сообщать кому-то, что вы собираетесь использовать определенное имя.

Если вам нужна новая личность, просто создайте ее.

Любой может создать новую личность в любое время и может сделать их столько, сколько захочет.

Если вы захотите быть под пятью именами, без проблем, просто сделайте пять идентификаторов.

Если вы хотите быть анонимным, вы можете создать новую личность, использовать ее только на некоторое время, затем выбросить ее.

Все это возможно с помощью децентрализованного управления идентификацией.

И нет центральной точки контроля, так что вам не нужно иметь кого-то, кто отвечает за идентификацию.

Система работает полностью децентрализованным способом.

И таким способом Биткойн создает личности.

Эти личности называются адресами в терминологии Биткойна.

И поэтому, когда вы слышите термин «адрес» в разговоре о биткойне и криптовалютах, на самом деле это просто открытый ключ или хэш открытого ключа.

Это личность, которую кто-то создал в рамках децентрализованной схемы управления идентификацией.

Теперь возникает вопрос, когда вы говорите о децентрализованном управлении идентификацией и людях, создающих эти личности, насколько это конфиденциально?

И ответ в том, что это сложно.

С одной стороны, адреса, составленные таким образом, не связанны с вашей реальной личностью в реальном мире.

Вы выполняете рандомизированный алгоритм, он создает какой-то публичный ключ, который выглядит случайным.

И изначально ничего не существует, что может связать это

с тем, кто вы есть на самом деле.

Вы можете сделать это конфиденциально.

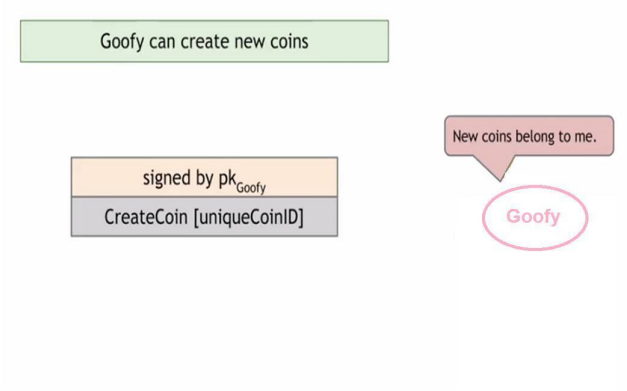
Но плохая новость в том, что даже если вы хотите действовать конфиденциально, и, если этот адрес, эта личность делает ряд утверждений со временем, если этот адрес участвует в ряде действий, люди могут увидеть, кто сделал эту определенную последовательность действий, и они смогут связать этот адрес с конкретным человеком.

Поэтому вопрос конфиденциальности в криптовалюте является сложным вопросом.

Простые криптовалюты

Теперь поговорим о некоторых упрощенных криптовалютах, чтобы потом понять, как работают такие системы, как BitCoin.

И сначала поговорим о GoofyCoin.



GoofyCoin – это простейшая криптовалюта, которая подчиняется двум правилам.

Первое правило состоит в том, что Гуфи может создавать новые монеты, когда он захочет, и когда он создает новую монету, она принадлежит ему.

Когда Гуфи создает монету, она представлена определенной структурой данных.

Чтобы создать монету, Goofy генерирует уникальный идентификатор монеты CoinID, который он никогда не генерировал раньше, и создает строку «CreateCoin [uniqueCoinID]».

Затем он подписывает эту строку цифровой подписью с помощью своего секретного ключа.

Он вычисляет цифровую подпись этой строки с использованием своего секретного ключа.

Эта строка вместе с подписью Гуфи – это и есть монета.

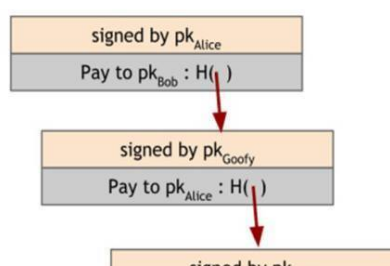
Кто угодно может проверить, что монета содержит действительную подпись Goofy для утверждения CreateCoin и, следовательно, она является действительной монетой.

И новые монеты принадлежат Гуфи по определению, потому что это правило, которое создал сам Гуфи.

Второе правило GoofyCoin заключается в том, что тот, кто владеет монетой, может передать ее кому-то другому.

Передача монеты – это не просто вопрос отправки структуры данных монеты получателю – она выполняется с использованием криптографических операций.

Предположим, Гуфи хочет передать монету, которую он создал, Алисе.



Для этого он создает новое выражение, в котором говорится: «Оплатите это Алисе», где «это» – это хэш-указатель, который ссылается на монету.

И, как мы определили ранее, идентификаторы на самом деле являются просто публичными ключами, поэтому «Алиса» это публичный ключ Алисы.

Наконец, Goofy подписывает строку, представляющую это выражение.

Поскольку Гуфи – это тот, кому принадлежит эта монета, он должен подписать любую транзакцию, которая передает эту монету.

Как только эта структура данных, представляющая сделку Гуфи, подписанная им же, существует, Алиса владеет этой монетой.

Она может доказать кому угодно, что она владеет монетой, потому что она может предоставить структуру данных с действительной подписью Гуфи.

Кроме того, эта структура данных указывает на валидную монету, принадлежащую Гуфи.

Таким образом, валидность и владение монетами самоочевидны в системе.

Теперь, когда Алиса владеет монетой, она может также ее потратить.

Для этого она создает выражение, в котором говорится: «Платите эту монету публичному ключу Боба, где «это» является хэш-указателем на монету, принадлежавшую ей.

И конечно, Алиса подписывает это выражение.

Любой, когда увидит эту монету, может проверить, что Боб является ее владельцем.

Он будет следовать цепочке хэш-указателей обратно к созданию монеты и сможет убедиться, что при каждом шаге, законный владелец подписал выражение, в котором говорится: «Платите эту монету новому владельцу».

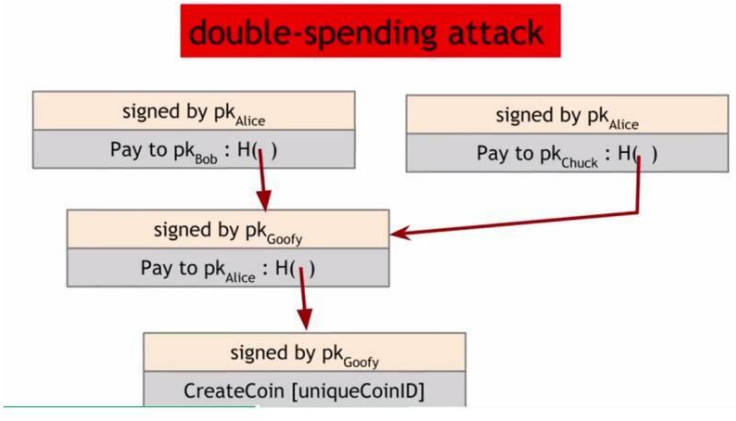
Подводя итог, GoofyCoin следует следующим правилам:

Goofy может создавать новые монеты, просто подписывая выражение о том, что он создает новую монету с уникальным идентификационным номером монеты.

Тот, кто владеет монетой, может передать ее кому-то другому, подписав выражение, в котором говорится: «Передайте эту монету X "(где X указывается как открытый ключ).

Любой может проверить действительность монеты, следуя цепочке хеш-указателей обратно к ее создателю Гуфи, проверяя все подписей на этом пути.

Конечно, с GoofyCoin существует фундаментальная проблема безопасности.



Скажем, Алиса передала свою монету Бобу, отправив подписанное выражение Бобу, но никому не об это не сказала.

Она может создать другое подписанное выражение, которое платит ту же самую монету Чаку.

Для Чака это кажется хорошей действительной транзакцией, и теперь он якобы является владельцем монеты.

Боб и Чак, оба теперь утверждают, что являются владельцем этой монеты.

Это называется атакой двойной траты – Алиса тратит одну и ту же монету дважды.

Интуитивно мы знаем, что монеты не должны работать таким образом.

Фактически, атаки с двойным расходованием являются одной из ключевых проблем, которые необходимо решить любой из криптовалют.

GoofyCoin не решает проблему двойного расходования, и поэтому она не защищена.

GoofyCoin – это простая криптовалюта, и ее механизм передачи монет на самом деле очень похож на биткойн, но поскольку она небезопасна, она не может использоваться как реальная криптовалюта.

Чтобы решить проблему с двойным расходованием, мы разработаем еще одну криптовалюту, которую мы назовем ScroogeCoin.

ScroogeCoin построена на основе GoofyCoin, но она немного сложнее с точки зрения структуры данных.

Первая ключевая идея заключается в том, что первоначальный объект Scrooge публикует бухгалтерскую книгу, в которую можно только добавлять записи, и которая содержит историю всех транзакций, которые произошли.

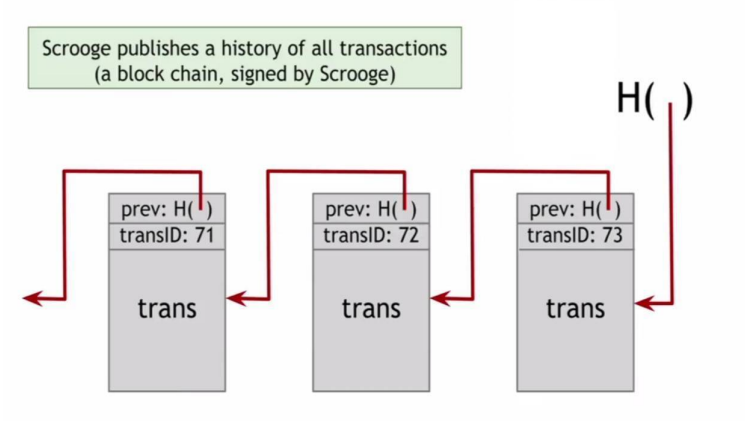
Свойство только добавления записей обеспечивает то, что любые данные, записанные в эту книгу, останутся навсегда.

Если книга действительно обладает свойством только добавления записей, мы можем использовать ее для защиты от

двойного расходования, требуя, чтобы все транзакции были записаны в книге, прежде чем они будут приняты.

Таким образом, транзакции будут общедоступны для просмотра, если монеты были ранее отправлены другому владельцу.

Чтобы реализовать эту функцию только добавления, Scrooge может построить цепочку блоков, структуру данных, которую мы уже видели, и которую он будет подписывать цифровой подписью.



Это серия блоков данных, каждый с одной транзакцией, на практике, в качестве оптимизации, помещается несколько транзакций в один и тот же блок, как у биткойна.

Каждый блок содержит идентификатор транзакции, со-

держимое транзакции и хэш указатель на предыдущий блок.

Scrooge в цифровой форме подписывает также и последний хэш-указатель, который связывает все данные в единую структуру и публикует подпись вместе с цепочкой блоков.

В ScroogeCoin транзакция учитывается только в том случае, если она находится в цепочке блоков, подписанной Scrooge.

Любой может убедиться, что транзакция была одобрена Scrooge, проверив подпись Scrooge для блока, который появляется в цепочке.

Скрудж гарантирует, что он не одобрит транзакцию, которая пытается дважды потратить уже потраченную монету.

Зачем нам нужна цепочка блоков с хэш указателями в дополнение к тому, что Scrooge подписывает каждый блок?

Это обеспечивает свойство только добавления.

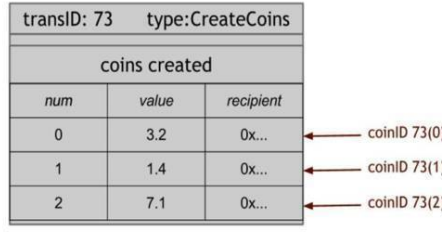
Если Scrooge попытается добавить или удалить транзакцию в истории, или изменить существующую транзакцию, это затронет все следующие блоки из-за хэш указателей.

Так как кто-то может отследить последний хэш-указатель, опубликованный Scrooge, это изменение будет очевидно и его легко обнаружить.

В системе, где бы Scrooge подписывал блоки по отдельности, вам бы нужно было отслеживать каждую подпись Scrooge, когда-либо выпущенную.

Цепочка блоков делает это очень легко, создавая единую историю транзакций, подписанную Скруджем.

В ScroogeCoin существует два вида транзакций.



Первый вид – это CreateCoins, которая аналогична операции GoofyCoin, которая делает новую монету.

В ScroogeCoin мы расширим немного семантику, чтобы разрешить создание нескольких монет в одной транзакции.

Эта транзакция CreateCoins создает несколько монет.

Каждая монета имеет серийный номер в транзакции.

И каждая монета также имеет значение, а именно стоимость как определенное количество ScroogeCoins.

И наконец, каждая монета имеет получателя, который является открытым ключом и который получает монету, когда она создана.

Таким образом, CreateCoins создает кучу новых монет с

разными значениями и присваивает их разным получателям как первоначальным владельцам.

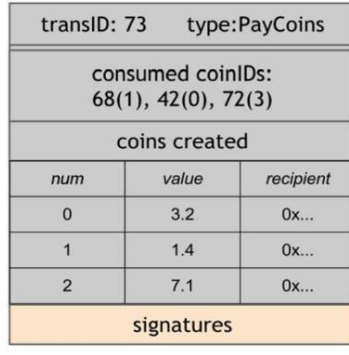
Мы ссылаемся на монеты с помощью CoinID.

CoinID – это комбинация идентификатора транзакции и серийного номера монеты в рамках этой транзакции.

Транзакция CreateCoins всегда действительна по определению, если она подписана Scrooge.

Мы не будем беспокоиться о том, когда Scrooge имеет право создавать монеты или сколько их создавать, точно так же, как мы не беспокоились в GoofyCoin о том, как Гуфи выбирается как сущность, которой позволено создавать монеты.

Второй вид транзакции – это PayCoins.



Эта транзакция потребляет несколько монет, то есть уничтожает их, и тут же создает новые монеты той же стоимостью.

Новые монеты могут принадлежать разным людям (публичным ключам).

Эта транзакция должна быть подписана всеми, кто платит монетой.

Поэтому, если вы являетесь владельцем одной из монет, которая будет потребляться в этой транзакции, тогда вам необходимо создать цифровую подпись этой транзакции, чтобы сказать, что вы действительно потратили эту монету.

Правила ScroogeCoin говорят, что транзакция PayCoins действительна, если действительны четыре вещи:

Используемые монеты являются валидными, то есть они действительно были созданы в предыдущих транзакциях.

Расходуемые монеты еще не были использованы в каких-либо предыдущих транзакциях. То есть, это не двойное расходование.

Общая стоимость монет, выходящих из этой транзакции, равна суммарному значению монет, которые вошли в транзакцию. То есть, только Scrooge может создать новую монету.

Транзакция действительно подписана владельцами всех потребляемых монет.

Если все эти условия выполнены, тогда транзакция PayCoins валидна, и Scrooge ее примет.

Он впишет эту транзакцию в историю, добавив ее в цепочку блоков, после чего каждый сможет увидеть, что эта транзакция случилась.

Только в этот момент все участники смогут принять, что транзакция на самом деле была произведена.

Пока она не будет опубликована, она может быть вытеснена транзакцией двойной траты, даже если она валидна в силу первых трех условий.

Монеты в этой системе неизменяемы – они никогда не меняются, не делятся и не объединяются.

Каждая монета создана один раз, в одной транзакции, и позже потребляется в какой-либо другой транзакции.

Но мы можем получить возможность делить или объединять монеты с помощью транзакций.

Например, чтобы разделить монету, Алиса может создать новую транзакцию, которая потребляет эту одну монету, а затем создает две новые монеты той же общей стоимостью.

Эти две новые монеты присваиваются ей обратно.

Теперь мы приходим к основной проблеме ScroogeCoin.

ScroogeCoin будет работать в том смысле, что люди могут видеть, какие монеты действительны.

Это предотвращает двойное расходование, потому что каждый может посмотреть на блок и убедиться, что все транзакции действительны и каждая монета потребляется только один раз.

Но проблема Скруджа в том, что у него слишком много

влияния.

Он не может создавать поддельные транзакции, потому что не может подделывать подписи других людей.

Но он может прекратить одобрять транзакции у некоторых пользователей, отказывая им в сервисе и делая их монеты бесполезными.

Если Скрудж жадный, как его тезка из мультфильма, он может отказаться публиковать транзакции, если они не передают ему какую-либо утвержденную комиссию за транзакцию.

Scrooge также может, создать столько новых монет для себя, сколько он хочет.

Или Скруджу может наскучить вся система, и он может полностью прекратить обновление цепочки блоков.

Проблема здесь в централизации.

Хотя Scrooge может и доволен этой системой, мы, как ее пользователи не можем быть довольными такой системой.

Хотя ScroogeCoin может показаться нереалистичной криптовалютой, большая часть ранних исследований криптосистем предполагали, что действительно будет какой-то центральный доверенный орган, обычно называемый банком.

В конце концов, у большинства валют реального мира есть доверенный эмитент, обычно правительственный монетный двор, который отвечает за создание валюты и определение того, какие денежные знаки действительны.

Однако, криптовалютам с центральным органом на практике не удалось реализоваться.

Этому есть много причин, но оглядываясь назад, кажется, что трудно заставить людей принять криптовалюту с централизованной властью.

Поэтому центральная техническая задача, которую нам необходимо решить, чтобы улучшить ScroogeCoin и создать работоспособную систему, это можем ли мы удалить Scrooge из системы?

То есть можем ли мы избавиться от этого централизованного органа?

Можем ли мы создать криптовалютносту, которая работает как ScroogeCoin во многих отношениях, но не имеет какой-либо центральной доверенной власти?

Для этого нам нужно выяснить, как все пользователи системы могут согласовать одну опубликованную блок-цепочку как историю всех транзакций, которые когда-то произошли.

Они должны все договориться о том, какие операции действительны, и какие транзакции действительно произошли.

Они также должны иметь возможность назначать идентификаторы децентрализованным способом.

Наконец, производство новых монет должно контролироваться децентрализованно.

Если мы сможем решить все эти проблемы, тогда мы сможем построить валюту, которая будет похожа на

ScroogeCoin, но без центральной власти.

Фактически, это будет система, очень похожая на биткойн.

Как биткойн обеспечивает децентрализацию

Децентрализация – это важная концепция, которая не является уникальной для Биткойна.

Вопрос конкурирующих парадигм централизации и децентрализации возникает во множестве различных ИТ технологий.

Чтобы лучше понять, как это работает в биткойне, полезно понять центральный конфликт – противоречие между этими двумя парадигмами – в других контекстах.

Например, с одной стороны, у нас есть Интернет, децентрализованная система, которая исторически конкурировала с информационными сервисами America Online и CompuServe.

Или есть электронная почта, которая по своей сути является децентрализованной системой, основанной на открытом стандарте Simple Mail Transfer Protocol (SMTP), а с другой стороны есть закрытые системы обмена сообщениями, такие как ВКонтакте, Facebook или LinkedIn.

Наконец, есть социальные сети – централизованные системы, такие как ВКонтакте, Facebook и LinkedIn, и распространенной децентрализованной альтернативы у них нет.

Надо учитывать, что децентрализация или централизация

не реализуются полностью.

Почти ни одна система не является чисто децентрализованной или чисто централизованной.

Например, электронная почта, в своей сути, представляет собой децентрализованную систему на основе стандартизованного протокола, SMTP, и любой, кто желает, может управлять собственным сервером электронной почты.

Тем не менее, развитие рынка сервисов электронной почты привело к его монополизации, и небольшое число централизованных поставщиков веб-почты стали доминирующими.

По аналогии, в то время как протокол Bitcoin является децентрализованным, такие сервисы, как обмен биткойнами, где вы можете конвертировать Биткойн в другие валюты, программное обеспечение для кошельков или программное обеспечение, которое позволяет людям управлять своими биткойнами могут быть централизованы или децентрализованы в различной степени.

Имея это в виду, давайте разделим вопрос о том, как протокол Bitcoin достигает децентрализации на пять более конкретных вопросов:

1. Кто хранит книгу транзакций?

2. Кто имеет власть утверждать действительность транзакций?

3. Кто создает новые биткойны?

4. Кто определяет, как изменяются правила системы?

5. Как биткойны получают обменную стоимость?

1. Кто хранит книгу транзакций?
2. Кто имеет власть утверждать действительность транзакций?
3. Кто создает новые биткойны?
4. Кто определяет, как изменяются правила системы?
5. Как биткойны получают обменную стоимость?

Первые три вопроса отражают технические подробности протокола биткойнов, и именно эти вопросы мы в первую очередь разберем.

Различные аспекты биткойна отражают разную степень централизации / децентрализации.

Одноранговая сеть является практически децентрализованной, так как любой может запустить узел биткойна.

Вы можете подключиться к сети и легко загрузить клиент

Bitcoin и запустить узел на вашем ноутбуке или на вашем ПК.

В настоящее время существует несколько тысяч таких узлов.

Биткойн-добыча технически также открыта для всех, но для этого требуются очень высокие затраты.

Из-за этого существует высокая степень централизации или концентрации в экосистеме добычи биткойнов. Многие в сообществе биткойнов считают это совершенно нежелательным.

Еще один аспект – это обновление программного обеспечения, на котором работают узлы Bitcoin, и это влияет на при изменении правил системы.

Существует множество совместимых реализаций протокола. Но на практике большинство узлов используют эталонную реализацию, разработчикам которого доверяет сообщество, и эти разработчики обладают большой властью.

Распределенное согласование

Давайте рассмотрим децентрализацию в Биткойне на техническом уровне.

Ключевой термин, который здесь возникнет, это консенсус и, в частности, распределенный консенсус.

Основная техническая проблема, которая возникает при создании распределенной системы цифровых денег – это достижение распределенного консенсуса.

Консенсус – это процесс получения согласованного результата группой участников, например, утверждение транзакций в распределённых системах.

Распределенный консенсус имеет различные применения, и он десятилетиями изучался компьютерной наукой.

Традиционное мотивирующее использование консенсуса – это обеспечение надежности в распределенных системах.

Для достижения надежности в распределенных системах необходимы протоколы, которые позволяют системе, как единому целому, функционировать, невзирая на отказ ограниченного числа ее компонентов.

Ключевым моментом здесь является не то, о чем договариваются процессы, а тот факт, что они все должны прийти к одинаковому результату.

Интуитивно понятно, что такое распределенный консенсус, но полезно рассмотреть техническое определение рас-

пределенного консенсуса, так как это поможет нам определить, соответствует ли определенный протокол этим требованиям.

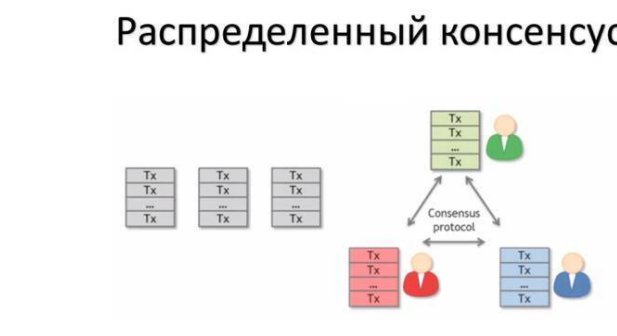
Предположим, что у нас есть n узлов, каждый из которых имеет входное значение.

И некоторые из этих узлов неисправны.

Распределенный консенсусный протокол имеет следующие два свойства:

Он должен завершиться с согласованным значением на всех корректных узлах.

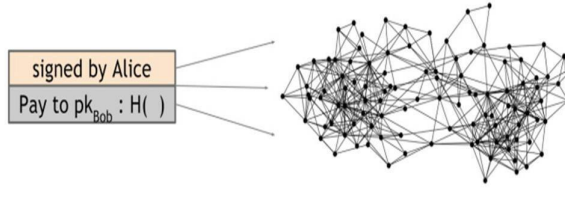
И это значение должно быть предложено корректным узлом, то есть оно само должно быть корректным.



Что это означает в контексте Биткойна?

Чтобы понять, как распределенный консенсус может работать в биткойне, помните, что биткойн – это одноранговая система.

Когда Алиса хочет заплатить Бобу, то, что она на самом деле делает, это транслирует транзакцию на все узлы Bitcoin, которые составляют одноранговую сеть.



Кстати, Алиса транслирует транзакцию на все одноранговые узлы Bitcoin, но при этом компьютер Боба может не находится в сети.

Конечно, возможно, что Боб запускает один из узлов в одноранговой сети.

Фактически, если он хочет получить уведомление о том, что эта транзакция действительно произошла и что он полу-

чил деньги, запуск узла Бобом был бы логичным.

Тем не менее, нет необходимости, чтобы Боб слушал в сети; запуск узла для Боба не требуется для получения средств.

Биткойны будут его, независимо от того, работает ли он в сети или нет.

Что именно здесь означает, что узлы достигают консенсуса в сети Bitcoin?

Учитывая, что множество пользователей транслируют эти транзакции в сеть, узлы должны согласовать учет, какие именно транзакции были транслированы, и порядок, в котором эти транзакции произошли.

Это приводит к созданию единого глобального журнала для системы.

Вспомним, что в ScroogeCoin для оптимизации мы помещаем транзакции в блоки.

Аналогичным образом, в биткойне мы принимаем консенсус на основе блокчейна.

Таким образом, все узлы в одноранговой сети содержат реестр, состоящий из последовательности блоков, каждый из которых содержит список транзакций, и таким образом они достигают консенсуса.

Кроме того, каждый узел содержит пул неутвержденных транзакций, о которых он слышал, но которые еще не включены в цепочку блоков.

Для этих транзакций консенсус еще не произошел, и поэтому по определению каждый узел может иметь немного от-

личающуюся версию пула неутвержденных транзакций.

На практике это происходит потому, что одноранговая сеть не идеальна, поэтому некоторые узлы, возможно, слышали о транзакции, о которой другие узлы не слышали.

Как именно узлы достигают консенсуса по блоку?

Один из способов сделать это: через равные промежутки времени, скажем каждые 10 минут, каждый узел в системе предлагает своему пулу неутвержденных транзакций быть следующим блоком.

Затем узлы выполняют некоторый консенсусный протокол, где вход каждого узла является его собственным предложенным блоком.

Теперь, некоторые узлы могут быть вредоносными и помещать недействительные транзакции в свои блоки, но мы можем предположить, что другие узлы будут корректными.

Если консенсусный протокол завершается успешно, в качестве вывода будет выбран валидный блок.

Даже если выбранный блок был предложен только одним узлом, это будет допустимый результат, если блок корректный.

Теперь, может быть какая-то валидная неутвержденная транзакция, которая не была включена в блок, но это не проблема.

Если какая-то транзакция каким-то образом не попала в этот конкретный блок, она может просто подождать и попасть в следующий блок.

Этот подход имеет некоторое сходство с тем, как работает биткойн, но это не совсем так, как на самом деле работает биткоин.

С этим подходом существует ряд технических проблем.

Во-первых, консенсус в целом является сложной проблемой, поскольку узлы могут дать сбой или быть злонамеренными.

Во-вторых, и особенно в контексте биткойнов, сеть крайне несовершенна.

Это одноранговая система, и не все пары узлов связаны друг с другом.

Например, в сети могут быть неисправности из-за плохого подключения к Интернету, и, таким образом, выполнение консенсусного протокола, в котором все узлы должны участвовать, на самом деле невозможно.

Наконец, в системе много задержек, потому что она охватывает весь Интернет.

Протокол биткойнов должен достичь консенсуса несмотря на две проблемы: ненадежность сети, так как есть задержки и сбой узлов, а также преднамеренные попытки некоторых узлов взломать процесс.

Одним из особых последствий задержек сети является отсутствие понятия глобального времени.

Это означает, что не все узлы могут согласиться на общий для всех узлов порядок событий, просто основанный на соблюдении временных меток.

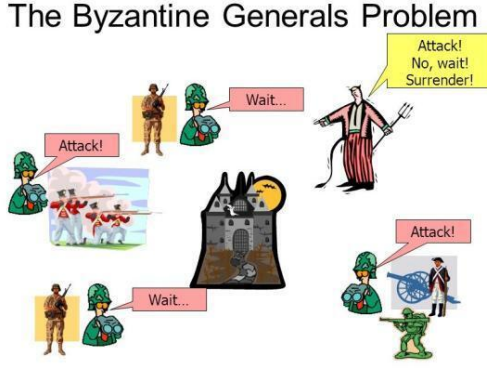
Поэтому консенсусный протокол не может содержать инструкции в виде: «Узел, который отправил самое первое сообщение на шаге 1 протокола, должен выполнить нечто на шаге 2».

Это не будет работать, потому что не все узлы согласятся, какое сообщение было отправлено первым на этапе 1 протокола.

Отсутствие глобального времени сильно ограничивает набор алгоритмов, которые могут использоваться в консенсусных протоколах.

Фактически, из-за этих ограничений большая часть исследований по распределенному консенсусу пессимистичны.

В качестве примера рассмотрим проблему византийских генералов.



В этой классической проблеме византийская армия разделена на дивизии, каждая из которых подчиняется генералу.

Генералы общаются с помощью посыльных, чтобы разработать совместный план действий.

Некоторые генералы могут быть предателями и могут преднамеренно попытаться подорвать процесс, чтобы лояльные генералы не могли прийти к единому плану.

Цель этой задачи состоит в том, чтобы все лояльные генералы пришли к одному и тому же плану, несмотря на попытки предателей генералов заставить их принять плохой план.

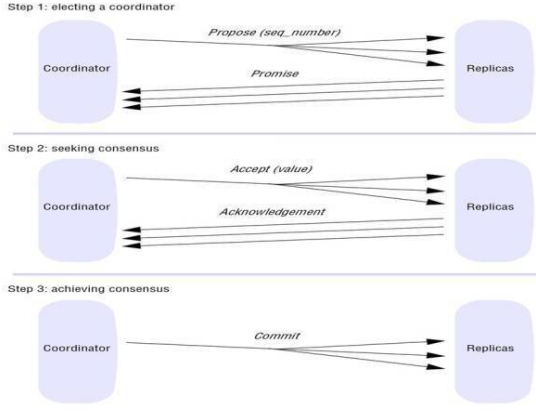
Было доказано, что этого невозможно достичь, если одна треть или более генералов являются предателями.

Другой результат невозможности достичь консенсуса, известный под именами авторов, которые его впервые доказали, называется результатом невозможности Фишера-Линча-Патерсона.

При некоторых условиях, которые включают в себя узлы, действующие детерминированным образом, они доказали, что консенсус невозможен даже при одном неисправном процессе.

Несмотря на эти результаты невозможности достичь консенсуса, все равно существуют некоторые консенсусные протоколы.

Одним из наиболее известных среди этих протоколов является Paxos.



Paxos допускает определенные компромиссы.

С одной стороны, он никогда не дает противоречивого результата.

С другой стороны, он принимает, что при определенных условиях, хотя и в редких случаях, протокол может зависнуть и не сможет добиться какого-либо результата.

Но есть хорошие новости: эти результаты невозможности были доказаны в очень конкретных моделях.

Они были предназначены для изучения распределенных баз данных, и эти модели не очень хорошо описывают биткойн, так как биткойн нарушает многие предположения, на которых эти модели были построены.

Как ни странно, при нынешнем состоянии исследований,

консенсус в Биткойне лучше работает на практике, чем в теории.

То есть мы наблюдаем за консенсусом, но не разработали теорию, чтобы полностью объяснить, почему он работает.

Но разработка такой теории важна, поскольку она может помочь нам предсказать непредвиденные атаки и проблемы, и только когда у нас будет сильное теоретическое понимание того, как работает Биткойн, мы будем иметь сильные гарантии безопасности и стабильности Bitcoin.

Какие предположения в традиционных моделях консенсуса, которые Биткойн нарушает?

Во-первых, в нем вводится идея стимулов, которая является новой для распределенного консенсусного протокола.

Это возможно только в биткойне, потому что это валюта, и поэтому он имеет естественный механизм, стимулирующий участников действовать честно.

Таким образом, биткойн не совсем решает проблему распределенного консенсуса в общем смысле, но решает ее в конкретном контексте валютной системы.

Во-вторых, биткойн включает в себя понятие случайности.

Как мы увидим далее, консенсусный алгоритм Биткойна в значительной степени зависит от рандомизации.

Кроме того, Биткойн устраняет понятие конкретной отправной точки и конечной точки для достижения консенсуса.

Вместо этого, консенсус происходит в течение длительного периода времени, около часа в реальной системе.

Но даже в конце этого периода времени узлы не могут быть уверены, что какая-либо конкретная транзакция или блок были включены в книгу.

Вместо этого, со временем, вероятность того, что прогноз будет соответствовать окончательному консенсусу, увеличивается, и вероятность расхождения консенсуса экспоненциально снижается.

Эти различия в модели являются ключевыми для способности Bitcoin обходить традиционные результаты невозможности достижения консенсуса для распределенных консенсусных протоколов.

Консенсус без идентификации: использование цепочки блоков

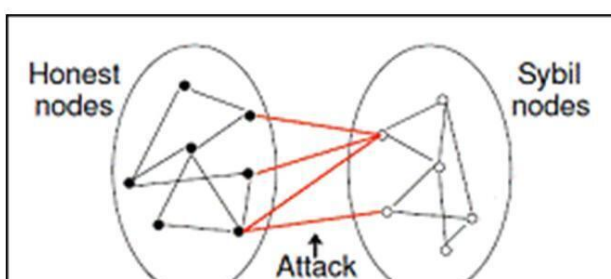
Далее мы рассмотрим технические детали алгоритма согласования биткойнов.

Напомним, что узлы биткойнов не имеют постоянных долгосрочных идентификаторов.

Это еще одно отличие от традиционных распределенных консенсусных алгоритмов.

Одной из причин этого недостатка идентификации является то, что в одноранговой системе нет центральной власти для назначения идентификаторов участникам и проверки того, что они не создают новые узлы по своему усмотрению.

Технический термин для этого – атака Сибиллы.



Сибиллы – это копии узлов, которые злонамеренный противник может создать, чтобы выглядеть как много разных участников, когда на самом деле все эти псевдо-участники контролируются одним и тем же противником.

Другая причина заключается в том, что псевдонимность по своей сути является целью Биткойна.

Напомню, что псевдонимность – это когда все транзакции между всеми адресами (кошельками) общедоступны, но нет данных о владельцах адресов. Однако личность владельца может быть установлена, если становится известна необходимая дополнительная информация.

Даже если бы было возможно или было легко установить идентификаторы для всех узлов или всех участников, мы бы не захотели этого делать.

Хотя Bitcoin не дает серьезных гарантий анонимности в том, что различные транзакции, которые вы делаете, часто могут быть связаны друг с другом, у него есть свойство, что никто не должен раскрывать свою реальную личность, такую как свое имя или IP-адрес, для участия в системе биткойн.

И это важное свойство и центральная особенность дизайна Биткойна.

Если бы узлы имели идентификаторы, дизайн был бы проще.

Для начала идентификаторы позволяли бы нам вводить в протокол инструкции формы: «Теперь узел с таким-то числовым идентификатором должен сделать такой-то шаг».

Без идентификаторов набор возможных инструкций более ограничен.

Но гораздо более серьезная причина для того, чтобы узлы имели идентификаторы – это для обеспечения безопасности.

Если бы узлы были идентифицированы и не нельзя было бы тривиально создавать новые идентификаторы узлов, то мы могли бы сделать предположения о числе узлов, которые являются вредоносными, и мы могли бы извлечь из этого какие-то свойства для обеспечения безопасности.

По обоим этим причинам отсутствие идентичности создает трудности для консенсусного протокола в Биткойне.

Мы можем компенсировать отсутствие идентичности, сделав не строгую идентификацию, а более слабую.

Предположим, что есть возможность выбрать случайный

узел в системе.

Хорошей мотивирующей аналогией для этого является лотерея.

То, что мы можем сделать в этом контексте, – это выдавать токены или билеты или что-то вроде этого.

Это позволяет нам позже выбрать случайный идентификатор токена и вызвать владельца этого идентификатора.

Поэтому на данный момент сделаем предположение, что таким образом можно выбрать случайный узел из сети биткойнов.

Далее предположим, что этот алгоритм генерации и распределения токенов достаточно умный, так что, если противник попытается создать множество узлов Сибиллы, все эти Сибиллы вместе получат только один токен.

Это означает, что противник не сможет увеличивать свою силу, создавая новые узлы.

Позже мы удалим эти предположения и подробно рассмотрим, как реализуются свойства, эквивалентные этим, в биткойне.

Это предположение о случайном выборе узла делает возможным то, что называется неявным консенсусом.

В нашем протоколе есть множество раундов, каждый из которых соответствует блоку в цепочке блоков.

В каждом раунде каким-то образом выбирается случайный узел, и этот узел получает возможность предложить следующий блок в цепочке.

Нет никакого консенсусного алгоритма для выбора блока и нет никакого голосования.

Выбранный узел в одностороннем порядке предлагает, какой будет следующий блок в цепочке блоков.

Но что, если этот узел злонамеренный?

Для обработки этого есть процесс, но он неявный.

Другие узлы будут неявно принимать или отклонять этот блок.

Если они согласятся с этим блоком, они просигнализируют о своем решении, расширяя цепочку блоков, и включая принятый блок.

Напротив, если они отклонят этот блок, они будут в дальнейшем расширять цепочку, игнорируя этот блок, и начиная с предыдущего блока в цепочке блоков.

Напомним, что каждый следующий блок содержит хеш блока, который он расширяет.

Это технический механизм, который позволяет узлам сигнализировать, какой блок узел расширяет.

Таким образом, упрощенный консенсусный алгоритм биткойна состоит в следующем.

1. Новые транзакции передаются всем узлам.

2. Каждый узел собирает новые транзакции в блок

3. В каждом раунде случайный узел получает возможность транслировать свой блок.

4. Другие узлы принимают блок только в том случае, если все транзакции в нем действительны (все подписи валидны).

5. Узлы выражают свое принятие блока, включая его хеш в следующем блоке, который они создают.

Этот алгоритм упрощен в том, что он предполагает возможность выбора случайного узла таким образом, что делает этот выбор не уязвимым для атак Сибиллы.

1. Новые транзакции передаются всем узлам.

2. Каждый узел собирает новые транзакции в блок

3. В каждом раунде случайный узел получает возможность транслировать свой блок.

4. Другие узлы принимают блок только в том случае, если все транзакции в нем действительны (все подписи валидны).

5. Узлы выражают свое принятие блока, включая его хеш в следующем блоке, который они создают.

Давайте теперь попытаемся понять, почему этот консенсусный алгоритм работает.

Для этого давайте рассмотрим, как вредоносный против-

ник, которого мы назовем Алисой, может подорвать этот процесс.

Рассмотрим кражу биткойнов.

Может ли Алиса просто украсть биткойны, принадлежащие другому пользователю, по адресу, который она не контролирует?

Нет. Даже если настанет очередь Алисы предложить следующий блок в цепочке, она не сможет украсть биткойны других пользователей.

Для этого потребуется, чтобы Алиса создала действительную транзакцию, которая делает проводку этой монеты.

Для этого нужно, чтобы Алиса подделала подписи владельцев, что она не может сделать, если используется безопасная схема цифровой подписи.

Таким образом, до тех пор, пока основная криптография будет строгой и надежной, она не сможет просто украсть биткойны.

Теперь рассмотрим возможность атаки на отказ в обслуживании.

Скажем, Алисе сильно не нравится какой-то другой пользователь Боб.

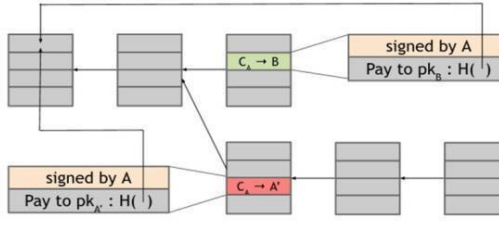
Поэтому Алиса может решить, что она не будет включать какие-либо транзакции, происходящие из адреса Боба, в любом блоке, который она предлагает, чтобы попасть в цепочку блоков.

Другими словами, она отказывает в сервисе Бобу.

К счастью, у Алисы в итоге ничего не получится, это будет не более чем незначительная досада.

Если транзакция Боба не будет включена в следующий блок, который предлагает Алиса, эта транзакция просто подождёт, пока честный узел не получит предложение предложить блок, а затем его транзакция попадет в этот блок.

Рассмотрим атаку двойной траты.



Алиса может попытаться запустить атаку двойной траты.

Чтобы понять, как это работает, предположим, что Алиса является клиентом какого-либо онлайн-продавца или вебсайта, который ведет Боб, который предоставляет некоторые онлайн-услуги в обмен на оплату в биткойнах.

Скажем, сервис Боба позволяет загружать некоторые про-

граммы.

Вот как может работать атака с двойной тратой.

Алиса добавляет товар в свою корзину на веб-сайте Боба, и сервер запрашивает платеж.

Затем Алиса создает транзакцию биткойна со своего адреса Бобу и транслирует ее в сеть.

Предположим, что какой-то честный узел создает следующий блок и включает эту транзакцию в этот блок.

Теперь есть блок, который был создан честным узлом, который содержит транзакцию, которая представляет платеж от Алисы покупателю Бобу.

Напомним, что транзакция представляет собой структуру данных, содержащую подпись Алисы, инструкцию для оплаты открытому ключу Боба и хэш.

Этот хэш представляет собой указатель на предыдущую транзакцию, который Алиса получила перед этим и сейчас тратит.

Этот указатель должен ссылаться на транзакцию, которая была включена в какой-то предыдущий блок в консенсусной цепочке.

Заметьте, кстати, что здесь есть два разных типа хеш-указателей, что может запутать.

Во-первых, Блоки содержат хеш-указатель на предыдущий блок, который они расширяют.

И во-вторых, транзакции включают один или несколько хэш указателей на предыдущие транзакции, которые не яв-

ляются потраченными.

Вернемся к тому, как Алиса может начать атаку двойной траты.

Последний блок был создан честным узлом и включает транзакцию, в которой Алиса платит Бобу за загрузку программного обеспечения.

Увидев эту транзакцию, включенную в цепочку блоков, Боб приходит к выводу, что Алиса заплатила ему и позволяет Алисе загрузить программное обеспечение.

Предположим, что следующий случайный узел, выбранный в следующем раунде протокола, контролируется Алисой.

Теперь, когда Алиса предложит следующий блок, она может предложить блок, который игнорирует блок, содержащий платеж Бобу, и вместо этого содержит указатель на предыдущий ему блок.

Кроме того, в блоке, который она предлагает, Алиса включает транзакцию, которая передает те самые монеты, которые она посылала Бобу, на другой адрес, который она же сама контролирует.

Это классический шаблон двойной траты.

Поскольку две транзакции делают проводку одних и тех же монет, только одна из них может быть включена в цепочку блоков.

Если Алисе удастся включить платеж на свой собственный адрес в цепочку блоков, тогда транзакция, в которой она

платит Бобу, становится бесполезной, так как эта транзакция никогда не может быть включена позже в цепочку блоков.

И как мы узнаем, удастся ли эта попытка двойной траты или нет?

Это зависит от того, какой блок в конечном итоге будет в конце консенсусной цепочкой – тот, который связан с транзакцией Алиса → Боб или с транзакцией Алиса → Алиса.

Что определяет, какой блок будет включен?

Честные узлы следуют политике расширения самой длинной действующей ветви, поэтому какая ветвь будет расширяться?

Правильного ответа нет!

На этом этапе две ветви имеют одинаковую длину – они отличаются только в последнем блоке, и оба этих блока действительны.

Затем узел, который выбирает следующий блок, может решить использовать один из этих блоков, и этот выбор определяет, удастся ли выполнить двойную трату.

С моральной точки зрения существует четкая разница между блоком, содержащим транзакцию, которая платит Бобу и блоку, содержащему транзакцию, в которой Алиса дважды тратит эти монеты на свой собственный адрес.

Но это различие основано только на наших знаниях о том, что Алиса впервые заплатила Бобу, а затем попыталась выполнить двойную трату.

С технологической точки зрения, однако, эти две транзак-

ции полностью идентичны, и оба блока одинаково действительны.

Узлы, которые смотрят на это, действительно не могут сказать, что является морально верной транзакцией. Для них они обе валидны.

На практике узлы часто расширяют блок, о котором они впервые услышали в одноранговой сети.

Но это не строгое правило.

И в любом случае, из-за латентности сети, легко может оказаться, что блок, о котором сначала услышал узел, на самом деле является блоком, который был создан позже.

Таким образом, существует некоторый шанс, что следующий узел, который может предложить блок, расширит блок, содержащий двойную трату.

Алиса могла бы еще больше увеличить вероятность того, что это произойдет, договорившись или контролируя следующий узел, чтобы это сделать.

Если следующий узел предложит блок, который ссылается на блок с двойной тратой по какой-либо причине, тогда эта цепочка теперь будет длиннее той, которая включает транзакцию для Боба.

На этом этапе следующий честный узел с большой вероятностью будет продолжать строить эту цепочку, поскольку он длиннее.

Этот процесс будет продолжаться, и становится все более вероятным, что блок, содержащий двойную трату, станет ча-

стью долгосрочной консенсусной цепи.

С другой стороны, блок, содержащий транзакцию для Боба, полностью проигнорируется сетью, и теперь это будет называться сиротским блоком.

Давайте теперь пересмотрим всю эту ситуацию с точки зрения Боба-торговца.

Понимание того, как Боб может защитить себя от этой атаки двойной траты, является ключевой частью понимания безопасности биткойнов.

Когда Алиса транслирует транзакцию, которая представляет ее платеж Бобу, Боб слушает в сети и слышит об этой транзакции еще до создания следующего блока.

Если бы Боб был еще более безрассудным, чем мы описали ранее, он может завершить процесс оформления заказа на веб-сайте и позволить Алисе загрузить программное обеспечение прямо в этот момент.

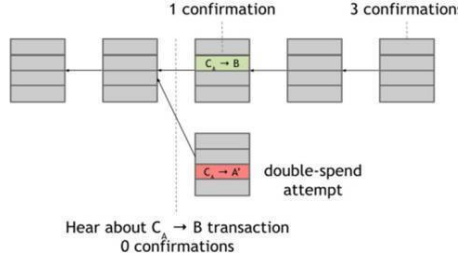
Это называется транзакцией с нулевым подтверждением.

Это приводит к появлению еще большей вероятности атаки двойной траты, чем было описано до этого.

Раньше для атаки двойной траты приходилось предполагать, что злоумышленник контролирует узел, предлагающий следующий блок.

Но если Боб разрешает Алисе загружать программное обеспечение до того, как транзакция получит хотя бы одно подтверждение в цепочке блоков, тогда Алиса может немедленно транслировать транзакцию с двойной тратой, а чест-

ный узел может ее включать в следующий блок, а не транзакцию, которая платит Бобу.



С другой стороны, осторожный торговец не даст программное обеспечение Алисе даже после того, как транзакция была включена в один блок и будет продолжать ждать.

Если Боб увидит, что Алиса успешно запускает атаку двойной траты, он поймет, что блок, содержащий оплату Алисы для него, остался сиротой.

Он должен отказаться от транзакции и не позволить Алисе загрузить программное обеспечение.

Если же вместо этого, несмотря на попытку двойной траты, следующие несколько узлов будут строить блокчейн с помощью блока транзакции Алиса → Боб, тогда Боб будет уве-

рен, что его сделка будет находиться в долгосрочной консенсусной цепочке.

В целом, чем больше существует подтверждений транзакции, тем выше вероятность того, что она будет содержаться в долгосрочной консенсусной цепочке.

Напомню, что поведение честных узлов всегда заключается в расширении самой длинной допустимой ветви, которую они видят.

Шанс, что более короткая ветка с двойной тратой догонит длинную ветку, становится все более незначительным, поскольку она растет дольше, чем любая другая ветвь.

Это предположение верно, если только меньшинство узлов является зловредными – для более короткой ветки, чтобы догнать более длинную ветку, несколько зловредных узлов должны будут подряд собирать блокчейн.

Фактически, вероятность двойного расходования экспоненциально уменьшается с количеством подтверждений.

Если транзакция, в которой вы заинтересованы, получила k подтверждений, то вероятность того, что транзакция с двойным расходованием будет в долгосрочной консенсусной цепочке, эта вероятность будет экспоненциально убывать как функция от k.

Наиболее распространенная практика, используемая в экосистеме биткойнов, – это ждать шесть подтверждений.

В числе шесть нет ничего особенного.

Это просто хороший компромисс между количеством

времени, которое вы должны ждать, и вашей гарантией того, что транзакция, в которой вы заинтересованы, попадает в цепочку консенсусного блока.

Напомню, что защита от недействительных транзакций полностью криптографическая.

Но эта защита осуществляется на основе консенсуса, а это означает, что если узел попытается включить криптографически недействительную транзакцию, то единственная причина, по которой транзакция не окажется в долгосрочной консенсусной цепочке, заключается в том, что большинство узлов являются честными и не будут включать недопустимую транзакцию в цепочку блоков.

С другой стороны, защита от двойных трат осуществляется исключительно и только на основе консенсуса.

Криптографии нечего сказать об этом, и две транзакции, которые представляют собой попытку двойной траты, действительны с криптографической точки зрения.

Но именно консенсус определяет, какая из них окажется в долгосрочной консенсусной цепочке.

И, наконец, вы никогда на 100 процентов не можете быть уверены, что транзакция, в которой вы заинтересованы, находится в консенсусной ветке.

Но экспоненциальная вероятностная гарантия этого довольно хорошая.

После шести транзакций практически нет шансов, что вы ошибетесь.

Стимулы и доказательства работы

Теперь у нас есть базовое представление консенсусного алгоритма Биткойна и представление, почему мы считаем, что это безопасно.

Но вспомним, что децентрализация Боткойна – это с одной стороны технический механизм, а с другой стороны умная стимуляция.

До сих пор мы в основном рассматривали технический механизм.

Теперь давайте поговорим о стимулах, которые применяются в Биткойне.

Также вспомним, что мы сделали предположение, что мы можем выбрать случайный узел и, возможно, более проблематично, что, по крайней мере, в 50 процентов случаев, при этом мы выберем честный узел.

Это предположение о честности особенно проблематично, если есть финансовые стимулы того, чтобы участники хотели подорвать процесс, и в этом случае мы не можем с точностью предположить, что узел будет честным.

Тогда возникает вопрос: можем ли мы дать узлам стимул вести себя честно?

Рассмотрим снова попытку двойной траты после одного подтверждения.

Можем ли мы каким-то образом наказывать узел, который

создал блок с транзакцией с двумя тратами?

Ответ – не совсем.

Как я говорил ранее, трудно понять, что является морально легитимной сделкой.

Но даже если бы мы это сделали, все равно трудно наказывать узлы, так как у них нет идентификаторов.

Поэтому вместо этого давайте подойдем с другой стороны и спросим, можем ли мы вознаградить каждый из узлов, которые создали блоки, которые в конечном итоге оказались в долгосрочной консенсусной цепочке?

И, опять же, поскольку эти узлы не раскрывают свою реальную идентичность, мы не можем отправить им наличные на их домашние адреса.

Только если бы существовала какая-то цифровая валюта, которую мы могли бы использовать для их стимулирования, возможно, децентрализованная.

Вероятно, вы уже видите, к чему я иду.

Другими словами, мы собираемся использовать биткойны, чтобы стимулировать узлы, которые создали эти блоки.

Давайте на мгновение остановимся.

Все, что мы описали до сих пор, является абстрактным алгоритмом для достижения распределенного консенсуса и не является специфичным для приложения.

Теперь мы уйдем из этой модели, и мы будем использовать тот факт, что приложение, которое мы строим с помощью этого распределенного процесса консенсуса, фактиче-

ски является валютой.

В частности, мы собираемся стимулировать узлы вести себя честно, платя им в единицах этой валюты.

Как это делается?

В Биткойне есть два отдельных механизма стимулирования.

Первое стимулирование – это награда за блок.

Согласно правилам биткойна, узел, который создает блок, включает специальную транзакцию в этот блок.

Эта транзакция представляет собой транзакцию создания монет, аналогичную CreateCoins в Scroogecoin, и узел также может выбрать адрес получателя этой транзакции.

Конечно, этот узел, как правило, выбирает адрес, принадлежащий самому себе.

Вы можете думать об этом как о плате узлу в обмен на услугу создания блока в консенсусной цепочке.

На сегодняшний момент значение вознаграждения блока фиксировано на уровне 12.5 биткойнов.

Но на самом деле это вознаграждение делится пополам при добавлении каждых 210 000 блоков.

Основываясь на скорости создания блока, которую мы вскоре увидим, это означает, что вознаграждение уполовинивается примерно каждые четыре года.

Сейчас мы находимся в третьем периоде.

В течение первых четырех лет существования Биткойна награда за блок составляла 50 биткойнов.

Теперь она равна 12.5 биткойнов.

Это имеет некоторые интересные последствия, которые мы увидим.

Возможно, будет интересно понять, почему вознаграждение в блоке стимулирует честное поведение.

Возможно, узел будет получать вознаграждение за блок, независимо от того, предлагает ли он действительный блок или злонамеренно пытается подменить блок.

Это не так!

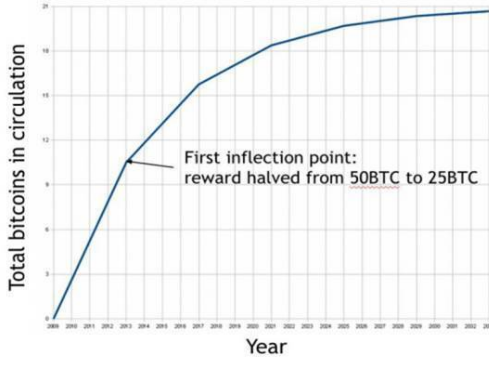
Потому что узел получит свою награду только в том случае, если этот блок попадет в долгосрочную консенсусную ветвь, потому что, как и любая другая транзакция, транзакция создания монеты будет приниматься другими узлами, только если она окажется в консенсусной цепочке.

Это ключевая идея этого механизма стимулирования.

Так как большая часть сети следит за самой длинной действующей ветвью, этот стимул побуждает узлы быть честными.

Это первый механизм стимулирования Биткойна.

Я уже сказал, что каждые 210 000 блоков (или примерно четыре года) вознаграждение за блок сокращается вдвое.



На слайде наклон этой кривой постоянно уменьшается вдвое, пока не уменьшится до нуля в 2140 году.

Bitcoin построен таким образом, чтобы увеличивать количество средств в логарифмической прогрессии пока не будет достигнута цифра в 21 млн. биткойнов.

Важно отметить, что майнинг, это единственный способ создания новых биткойнов.

Не существует другого механизма генерации монет, и именно поэтому 21 млн – это окончательное и общее число (такие сейчас правила), сколько биткойнов может когда-либо быть.

Поэтому вознаграждение за создание нового блока закончится в 2140 году.

Означает ли это, что система остановит свою работу в

2140 году и станет небезопасной, потому что у узлов больше не будет стимула вести себя честно?

Не совсем.

Награда за блок является лишь первым из двух механизмов стимулирования в Биткойне.

Второй механизм стимулирования называется комиссией за транзакцию.

Создатель любой транзакция может выбрать комиссию или разницу между общим значением транзакционных выходов и общим значением транзакционных входов.

Тот, кто создает блок и который первым помещает эту транзакцию в цепочку блоков, получает эту разницу, которая действует как транзакционная плата.

Поэтому, если вы являетесь узлом, который создает блок, содержащий, скажем, 200 транзакций, тогда сумма всех этих 200 комиссионных сборов выплачивается по адресу, который вы помещаете в этот блок.

Плата за транзакцию является чисто добровольной, но ожидается, что по мере того, как вознаграждение для блока начнет заканчиваться, становится все более и более важным, почти обязательным, чтобы пользователи включали транзакционные сборы, чтобы получить разумное качество обслуживания.

В определенной степени это уже сейчас происходит.

Но пока неясно, как будет развиваться система.

Это интересная область открытых исследований в Бит-

койне.

И по-прежнему остается несколько проблем с консенсусным механизмом, который мы описали.

Первой проблемой является предположение, что мы можем выбрать случайный узел.

Во-вторых, мы создали новую проблему, предоставив узлам стимулы для участия.

Система может стать нестабильной, так как стимулы создают среду, где каждый хочет запустить биткойн узел в надежде получить вознаграждение.

И третья – более сложная версия этой проблемы, которая заключается в том, что злоумышленник может создать большое количество узлов Сибиллы, чтобы попытаться подорвать консенсусный процесс.

Оказывается, все эти проблемы связаны, и все они имеют решение, которое называется доказательством работы.

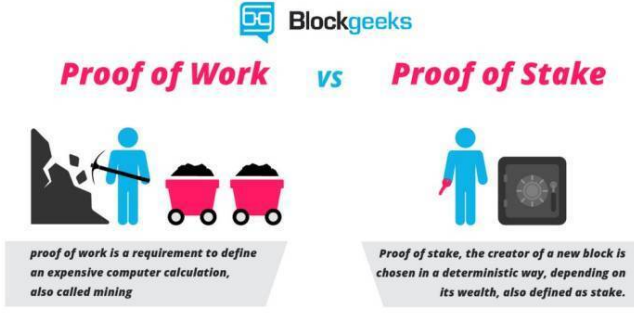
Ключевая идея доказательства работы заключается в том, что вместо просто выбора случайного узла, мы увеличиваем вероятность выбора узла или приближаем выбор случайного узла, пропорционально ресурсу, который как мы надеемся, никто не сможет монополизировать.

Если, например, этот ресурс является вычислительной мощностью, то тогда это будет системой доказательства работы.

В качестве альтернативы это может быть пропорционально владению валютой, и это называется системой доказатель-

ства ставки.

Хотя эта система не используется в биткойне, доказательства ставки является альтернативной моделью, и такая система используется в других криптовалютах.



Мы рассмотрим доказательства ставки и другие варианты доказательства работы позже.

Вернемся к доказательству работы.

Давайте попробуем лучше понять, что значит выбирать узлы пропорционально их вычислительной мощности.

Другим способом понять это является то, что мы позволяем узлам конкурировать друг с другом, используя их вычислительную мощность, и это приводит к тому, что узлы автоматически выбираются в этой пропорции.

Еще один взгляд на доказательство работы заключается в том, что создание нового идентификатора узла будет умеренно тяжелым.

Это своего рода налог на создание идентичности и, следовательно, на атаку Сибиллы.

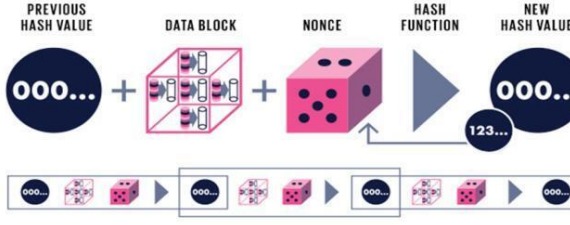
Это может показаться немного расплывчатым, поэтому давайте посмотрим на детали системы доказательства работы, которые используются в биткойне, чтобы прояснить эту систему.

Биткойн получает доказательства работы с использованием головоломок.

Чтобы создать новый блок, узел, который предлагает этот блок, должен найти номер или nonce.

Когда вы объединяете nonce, хеш предыдущего блока и список транзакций, которые составляют новый блок, и получаете хэш всей этой строки, то этот полученный хэш должен быть числом, которое попадает в диапазон, которое довольно мал по отношению к значительно большему выходному пространству этой хэш-функции.

Мы можем определить такое целевое пространство или диапазон, как любое значение, которое попадает ниже определенного целевого значения. В этом случае nonce должен будет удовлетворять неравенству.



Как мы видели ранее, обычно блок содержит ряд транзакций, которые предлагает узел.

Кроме того, блок также содержит указатель хеша на предыдущий блок.

И кроме того, теперь мы требуем, чтобы блок также содержал nonce.

Идея состоит в том, что мы хотим сделать сложной задачу нахождения nonce, которое удовлетворяет определенному свойству, состоящему в том, что хеширование всего блока вместе, в том числе и nonce, приведет к определенному типу результата.

Если хеш-функция удовлетворяет свойству головоломки, то единственный способ добиться успеха в решении этой задачи с хэшем состоит в том, чтобы просто перебирать nonce

один за другим, пока вам не повезет.

Так что конкретно, если целевое пространство для результата составляет всего один процент от общего объема выходных данных, вам нужно попробовать около 100 nonce, прежде чем вы попадете в диапазон.

$$H(nonce \parallel prev\_hash \parallel tx \parallel tx \parallel ... \parallel tx) < target$$

В реальности, размер этого целевого пространства намного меньше одного процента выходного пространства.

Это понятие головоломок и доказательство работы полностью устраняет необходимость волшебного выбора случайного узла.

Вместо этого узлы просто самостоятельно конкурируют, чтобы все время решать головоломки.

Время от времени одному из них везет, и он находит слу-

чайный нонс, который удовлетворяет этому свойству.

Затем этот удачливый узел предлагает следующий блок.

Вот каким образом система полностью децентрализована.

Никто не решает, какой узел получает возможность предложить следующий блок.

Существует три важных свойства головоломки.

Во-первых, ее довольно сложно вычислить.

И эта сложность меняется со временем.

По состоянию на конец 2014 года уровень сложности составлял около 10 в 20 степени хэшей на блок.

Другими словами, размер целевого пространства составлял всего 1/(10 в 20 степени) от размера выходного пространства хэш-функции.

Это очень много вычислений – например, это за пределами возможности обычного ноутбука.

Из-за этого далеко не все узлы пытаются конкурировать в этом процессе создания блока.

Этот процесс повторяющихся попыток и решений этих головоломок хэша известен как майнинг биткойна и участвующие в этом процессе узлы называются майнерами.

Хотя технически кто угодно может быть майнером, для этого сейчас высокая стоимость входа в этот процесс.

Второе свойство головоломки состоит в том, что мы хотим, чтобы стоимость решения была параметризируемой, а не фиксированной стоимостью все время.

То, как это делается, заключается в том, что все узлы в од-

норанговой сети Bitcoin автоматически пересчитывают размер целевого пространства в виде доли выходного пространства, каждые 2016 блоков.

Они пересчитывают диапазон таким образом, чтобы среднее время между последовательными блоками, создаваемыми в сети Биткойн, составляла около 10 минут.

С 10-минутным среднем временем между блоками, 2016 блоков работают до двух недель.

Другими словами, пересчет целевого пространства происходит примерно раз в две недели.

Давайте подумаем, что это значит.

Если вы майнер, и вы вложили определенное количество компьютерного железа в добычу биткойнов, но общая экосистема майнинга растет, в нее поступают все больше майнеров или они развертывают все более мощное аппаратное обеспечение, это означает, что за двухнедельный период, будет найдено немного больше блоков, чем ожидалось.

Таким образом, узлы будут автоматически перенастраивать диапазон головоломки, и объем работы, которую вы должны будете сделать, чтобы найти блок, будет увеличиваться.

Поэтому, если вы вложили фиксированный объем инвестиций в оборудование, скорость, с которой вы находите блоки, фактически зависит от того, что делают другие майнеры.

Есть очень хорошая формула, чтобы зафиксировать то, что вероятность того, что любой данный майнер, например,

Алиса, выиграет следующий блок, пропорциональна доли глобальной хэш-мощности, которую она контролирует.

Это означает, что, если у Alice есть оборудование для майнинга, которое составляет около 0,1 процента от общей хэш-мощности экосистемы, она будет добывать примерно один из каждых 1000 блоков.

Какая цель у этой корректировки сложности головоломки?

Почему мы хотим сохранить этот 10-минутный интервал между новыми блоками?

Причина довольно проста.

Если бы блоки были очень близки друг к другу, тогда мы потеряли бы преимущества оптимизации, заключающиеся в том, что мы можем вносить много транзакций в один блок.

Нет ничего волшебного в отношении числа 10.

И было много дискуссий о идеальной латентности блока, которую альткоины или альтернативные криптовалюты должны иметь.

Но, несмотря на некоторые разногласия относительно идеальной задержки, все согласны с тем, что это должна быть фиксированная величина.

Вот почему у нас есть функция автоматического пересчета целевого пространства или сложности головоломки.

Способ настройки доказательства работы позволяет нам переформулировать наше предположение о безопасности.

Вместо того, чтобы сказать, что почему-то большинство

узлов являются честными в контексте, где узлы даже не имеют идентификаторов и не понимают, что это значит, мы можем теперь четко заявить, что атаки на биткойн маловероятны, если большинство майнеров, взвешенные по хэш-мощности, следуют протоколу – или, являются честными.

Это так, потому что, если большинство майнеров, взвешенные по хеш-мощности, честны, конкуренция за предложение следующего блока автоматически гарантирует, что существует по меньшей мере 50-процентный шанс, что следующий блок, который будет предложен в любой момент, исходит от честного узла.

В области исследований распределенных систем и компьютерной безопасности, как правило предполагают, что некоторый процент узлов является честным, и показывают, что система работает как предназначено, даже если другие узлы ведут себя произвольно.

Это тот же подход, которому следовали и мы, за исключением того, что мы взвешиваем узлы по хэш-мощности при определении большинства узлов.

В исходном документе Биткойн также содержится этот тип анализа.

Но теория игр дает совершенно другой, и, возможно, более сложный и реалистичный способ определения того, как будет себя вести система.

В этом представлении мы не разделяем узлы на честные и злонамеренные.

Вместо этого мы предполагаем, что каждый узел действует в соответствии со своими стимулами.

Каждый узел выбирает (рандомизированную) стратегию, чтобы максимизировать свой выигрыш, учитывая потенциальные стратегии других узлов.

Если протокол и стимулы разработаны хорошо, то большинство узлов будут следовать правилам большую часть времени.

«Честное» поведение – это всего лишь одна стратегия из многих.

Исходя из теории игр, большой вопрос заключается в том, является ли поведение майнера по умолчанию «равновесием Нэша», то есть представляет ли оно стабильную ситуацию, в которой ни один майнер не может реализовать более высокий выигрыш, отклонившись от честного поведения.

Этот вопрос по-прежнему спорный и является активной областью исследований.

Равнове́сие На́ша – одно из ключевых понятий теории игр. Так называется набор стратегий в игре для двух и более игроков, в котором ни один участник не может увеличить выигрыш, изменив свою стратегию, если другие участники своих стратегий не меняют.

Решение головоломок-хэшей вероятностно, потому что никто не может предсказать, какой из nonce приведет к решению головоломки.

Единственный способ сделать это – попробовать nonce

один за другим и надеяться, что это удастся.

Математически этот процесс называется испытаниями Бернулли.

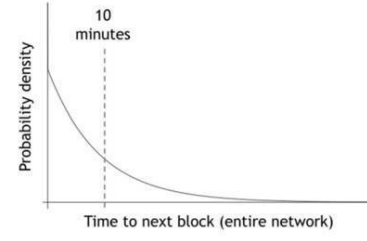
Испытание Бернулли – это эксперимент с двумя возможными результатами, и вероятность получения каждого результата зафиксирована между последовательными испытаниями.

Здесь два результата: попадает ли хэш в целевое пространство или нет.

И, предполагая, что функция хэша ведет себя как случайная функция, вероятность этих исходов фиксирована.

Как правило, узлы пробуют так много nonce, что испытания Бернулли, являющиеся дискретным вероятностным процессом, могут быть хорошо аппроксимированы непрерывным вероятностным процессом, называемым пуассоновским процессом, процессом, в котором события происходят независимо с постоянной средней скоростью.

Конечным результатом всего этого является то, что функция плотности вероятности, которая показывает относительную вероятность времени, пока не будет найден следующий блок, выглядит следующим образом.



$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

λ = 1/(10 minutes)

Это называется экспоненциальным распределением.

Вблизи 10 минут, если существует небольшая вероятность того, что блок будет найден сейчас, следующий блок будет найден очень скоро, скажем, в течение нескольких секунд или минуты.

И существует также небольшая вероятность того, что потребуется целый час, чтобы найти следующий блок.

Но в целом сеть автоматически регулирует сложность, так что межблочное время поддерживается в среднем, в течение 10 минут.

Если вы майнер, вам, вероятно, интересно, сколько времени вам понадобится, чтобы найти блок.

Как выглядит эта функция плотности вероятности?

Она будет иметь ту же форму, но просто будет иметь дру-

гой масштаб по оси x.

Опять же, это может быть представлено уравнением.

$$\textit{mean time to next block} = \frac{10\ \textit{minutes}}{\textit{fraction of hash power}}$$

Для конкретного майнера это будет зависеть от его хэш-мощности.

Если у вас есть 0,1 процента от общей хэш-мощности сети, это уравнение говорит нам, что вы будете находить блоки каждые 10 000 минут, что составляет примерно одну неделю.

Мало того, что ваше среднее время между нахождениями блоков будет очень высоким, но дисперсия времени между найденными вами блоками также будет очень высокой. Это имеет некоторые важные последствия, которые мы рассмотрим позже.

Перейдем теперь к третьему важному свойству голово-

ломки или функции доказательства работы, которое позволяет легко проверить, что узел действительно правильно вычислил доказательство работы.

После, скажем, 10 в 20 степени попыток найти nonce, который делает хеш блока ниже целевого значения, этот nonce должен быть опубликован как часть блока.

Таким образом, очень просто, любой другой узел может просмотреть содержимое блока, вычислить хэш его содержимого и проверить, что результат будет меньше целевого значения.

Это важное свойство, потому что, опять же, оно позволяет нам избавиться от централизации.

Нам не нужна никакая центральная власть, которая проверяет, что майнеры выполняют свою работу правильно.

Любой узел или любой майнер может мгновенно проверить, что блок, найденный другим майнером, удовлетворяет этому свойству проверки доказательства работы.

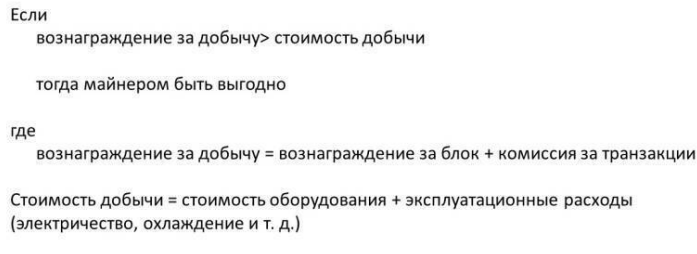
Давайте теперь посмотрим на экономику майнинга.

Все мы уже слышали, что быть майнером, это довольно дорого.

При текущем уровне сложности поиск одного блока требует вычисления около 10 в 25 степени хэшей, а вознаграждение в блоке составляет около 12.5 биткойнов, что представляет собой значительную сумму денег по текущему обменному курсу.

Эти цифры позволяют легко вычислить, выгодно ли это

для меня, и мы можем определить это решение простым выражением:



Если
вознаграждение за добычу> стоимость добычи
тогда майнеров быть выгодно
где
вознаграждение за добычу = вознаграждение за блок + комиссия за транзакции
при этом, стоимость добычи = стоимость оборудования + эксплуатационные расходы (электричество, охлаждение и т. д.)

Награда за майнинг, которую получает майнер, состоит из вознаграждения за блок и транзакционных сборов.

Майнер должен сравнить это с общими расходами, а именно стоимостью оборудования и электроэнергии.

Но для этого простого уравнения есть некоторые сложности.

Во-первых, стоимость оборудования является фиксированной стоимостью, тогда как стоимость электроэнергии представляет собой переменную стоимость, которая может изменяться с течением времени.

Еще одна сложность заключается в том, что вознаграждение, получаемое майнерами, зависит от скорости, с которой они находят блоки, что зависит не столько от мощности их оборудования, как от соотношения их хэш-мощности к общей глобальной хэш-мощности.

Третья сложность состоит в том, что затраты, которые несет майнер, обычно выражаются в долларах или какой-то другой традиционной валюте, но их вознаграждение выражено в биткойне.

Таким образом, это уравнение имеет скрытую зависимость от обменного курса Биткойна в любой момент времени.

И, наконец, до сих пор мы предполагали, что майнер заинтересован честно следовать протоколу.

Но майнер может выбрать какую-то другую стратегию добычи биткойнов, а не пытаться продлить самую длинную действующую ветвь.

Таким образом, это уравнение не отражает все нюансы

различных стратегий, которые может использовать майнер.

На самом деле анализ того, имеет ли смысл добыча, – сложная проблема теории игр, на которую нелегко ответить.

На данный момент мы получили довольно хорошее представление о том, как биткойн достигает децентрализации.

Теперь соединим все это вместе, чтобы получить лучшее понимание.

Начнем с идентификаторов.

Как мы узнали, для участия в протоколе биткойнов нет реальных идентификаторов.

Любой пользователь может создать пару псевдонимных ключей в любой момент, любое их количество.

Когда Алиса хочет заплатить Бобу за биткойны, в протоколе биткойнов не указано, как Алиса узнает адрес Боба.

Принимая эти пары псевдонимных ключей как идентификаторы, транзакции – это сообщения, которые передаются в одноранговую сеть Bitcoin и которые являются инструкциями по передаче монет с одного адреса на другой.

Биткойны – это только транзакционные выходы, и мы обсудим это более подробно далее.

Биткойн не имеет фиксированных номиналов, так как доллары США, и, в частности, нет специального обозначения «1 биткойн».

Биткойны – это только транзакционные выходы, и при текущих правилах они могут иметь произвольное значение с 8 десятичными знаками после запятой.

Наименьшее возможное значение – 0.000 000 01 BTC (биткойны), которое называется 1 Satoshi.

Целью одноранговой сети Bitcoin является распространение всех новых транзакций и новых блоков на все одноранговые узлы Bitcoin.

Но сеть крайне несовершенна и делает попытки ретрансляции этой информации с максимальной эффективностью.

Безопасность системы не основывается на совершенстве одноранговой сети.

Вместо этого безопасность основывается на цепочки блоков и консенсусного протокола.

Когда мы говорим, что транзакция включена в цепочку блоков, мы имеем в виду, что транзакция получила многочисленные подтверждения.

Нет фиксированного числа, сколько требуется подтверждений, прежде чем мы достаточно убедимся о включении транзакции, но шесть подтверждений – это обычная практика.

Чем больше подтверждений, полученных транзакцией, тем больше вы уверены, что эта транзакция является частью консенсусной цепи.

Также существуют сиротские блоки или блоки, которые не включаются в консенсусную цепочку.

Существует множество причин, которые могут привести к появлению сиротских блоков.

Блок может содержать недопустимую транзакцию или по-

пытку двойной траты.

Это также может быть просто результатом латентности сети.

То есть, два майнера могут просто найти новые блоки всего за несколько секунд друг от друга.

Поэтому оба этих блока будут транслироваться почти одновременно в сеть, и один из них неизбежно останется сиротой.

Наконец, мы рассмотрели хэш головоломку и добычу.

Майнеры – это специальные типы узлов, которые решают конкурировать в игре по созданию новых блоков.

Они вознаграждаются за это новыми биткойнами (как вознаграждение за новый блок) и существующими биткойнами (как транзакционные сборы).

Далее, тонкий, но важный момент: скажем, что Алиса и Боб – два разных майнера, и у Алисы в 100 раз больше вычислительной мощности, чем у Боба.

Это не означает, что Алиса будет всегда выигрывать гонку против Боба, чтобы найти следующий блок.

Вместо этого Алиса и Боб имеют коэффициент вероятности нахождения следующего блока в пропорции 100 к 1.

В долгосрочной перспективе Боб найдет в среднем один процент от числа блоков, которые находит Алиса.

Мы ожидаем, что майнеры, как правило, будут находиться где-то близко в экономическом равновесии, в том смысле, что расходы, которые они берут на себя в отношении обо-

рудования и электроэнергии, будут примерно компенсироваться получаемым ими вознаграждениями.

Потому что, если майнер будет постоянно в убытке, он, вероятно, прекратит добычу.

С другой стороны, если добыча очень выгодна при типичных расходах на оборудование и электроэнергию, тогда в сеть войдет больше оборудования для майнинга.

Тогда увеличенная скорость хэширования приведет к увеличению сложности, и ожидаемая награда каждого майнера упадет.

Понятие распределенного консенсуса пронизывает Биткойн довольно глубоко.

В традиционной валюте консенсус работает в определенной мере.

В частности, существует консенсусный процесс, который определяет обменный курс валюты.

Это, также, верно и в биткойне.

Здесь нам также нужен консенсус относительно ценности биткойна.

Но в Биткойне, кроме того, нам нужен консенсус относительно состояния книги или реестра, который утверждает цепочки блоков.

Другими словами, даже подсчет того, как много биткойнов у вас есть, подчиняется консенсусу.

Когда мы говорим, что Алисе принадлежит определенная сумма или количество биткойнов, мы фактически имеем в

виду, что одноранговая сеть Bitcoin, согласно записи в блокчейне, считает, что общему количеству всех адресов Алисы принадлежит данное число биткойнов.

Это и есть природа биткойна.

Владение биткойнами – это не что иное, согласие других узлов с тем, что данная сторона владеет этими биткойнами.

И наконец, нам нужен консенсус относительно правил системы, потому что изредка правила системы должны меняться.

Существует два типа изменений в правилах биткойнов, известные как мягкий форк и жесткий форк.

Мы рассмотрим форки позже.

Еще одна тонкая концепция – это процесс начальной загрузки.

В Биткойне существует сложное взаимодействие между тремя различными идеями: безопасность цепочки блоков, здоровье экосистемы майнинга и стоимость валюты.

Мы, очевидно, хотим, чтобы блок-цепочка была безопасной для Биткойна, чтобы биткойн был жизнеспособной валютой.

И чтобы блокчейн был безопасным, злоумышленник не должен нарушать консенсусный процесс.

Это, в свою очередь, означает, что противник не может создать много узлов для майнинга и забрать себе более 50 % создания новых блоков.

Необходимым условием безопасности биткойна является

наличие здоровой экосистемы майнинга, состоящей, в основном, из честных, следующих протоколу узлов.

Но когда большое количество майнеров будет вкладывать много вычислительной мощности в решение головоломки с хэшем?

Они будут делать это только в том случае, если обменный курс биткойна довольно высокий, потому что вознаграждения, которые они получают, выражены в биткойнах, тогда как их расходы в долларах.

Таким образом, чем больше ценность валюты повышается, тем больше стимулов у этих майнеров будет.

Теперь, что обеспечивает высокую и стабильную стоимость валюты?

Это может произойти только в том случае, если пользователи в целом доверяют безопасности цепочки блоков.

Если они посчитают, что сеть может быть захвачена в любой момент злоумышленником, тогда биткойн не будет иметь ценности в качестве валюты.

Таким образом, у вас есть взаимозависимость между безопасностью цепочки блоков, здоровой экосистемой майнинга и обменным курсом.

Из-за цикличности этой трехсторонней зависимости существование каждой из этих вещей основано на существовании других.

Когда Биткойн был впервые создан, ни одно из этих трех свойств не существовало.

Не было никаких майнеров, кроме Накамото, который сам запускал программное обеспечение для майнинга.

Биткойн не имел большой ценности в качестве валюты.

И цепочка блоков была, по сути, небезопасной, потому что не было много майнеров, и кто-то мог легко перегрузить этот процесс.

Нет простого объяснения тому, как Биткойн перешел от неимения ни одного из этих свойств, чтобы иметь все три из них.

Внимание средств массовой информации было частью истории – чем больше людей слышат о Биткойне, тем больше они будут заинтересованы в майнинге.

И чем больше они будут заинтересованы в майнинге, тем больше уверенности у людей будет в безопасности цепочки блоков, потому что в настоящее время продолжается большой майнинг и т. д. по кругу.

Кстати, каждый новый Altcoin, который хочет добиться успеха, также должен как-то решать эту проблему самораскрутки.

Наконец, давайте рассмотрим, что произойдет, если консенсус потерпит неудачу, и появится 51-процентный злоумышленник, который будет контролировать 51 % или более мощности майнинга в сети Bitcoin.

Мы рассмотрим несколько возможных атак и посмотрим, какие из них действительно могут быть выполнены таким злоумышленником.

Прежде всего, может ли этот злоумышленник украсть монеты из существующего адреса?

Как вы, возможно, догадались, ответ отрицательный, потому что кража из существующего адреса невозможна, если вы не взломаете криптографию.

Недостаточно подорвать консенсусный процесс.

Предположим, что 51-процентный злоумышленник создает недопустимый блок, который содержит недопустимую транзакцию, которая представляет кражу биткойнов из существующего адреса, который злоумышленник не контролирует, и передает эти биткойны на свой собственный адрес.

Нападавший может притворяться, что это действительная транзакция и продолжать строить блокчейн на основе этого блока.

Злоумышленник может даже добиться успеха в создании самой длинной ветки.

Но другие честные узлы просто не будут принимать этот блок с недопустимой транзакцией и будут майнить на основе последнего действительного блока, который они обнаружили в сети.

Итак, что произойдет в этом случае, так это то, что мы будем называть вилкой в цепочке или форком.

Теперь представьте себе это с точки зрения злоумышленника, пытающегося потратить эти недействительные монеты, и отправить их некому продавцу Бобу в качестве оплаты за некоторые товары или услугу.

Боб предположительно запускает свой биткойн-узел, и это будет честный узел.

Узел Боба отклонит эту ветвь как недействительную, так как она содержит недействительную транзакцию.

Эта ветка недействительна, потому что подписи не проходят проверку.

Поэтому узел Боба просто игнорирует самую длинную ветвь, потому что это недействительная ветка.

И из-за этого, недостаточное подорвать консенсус.

Вы должны подорвать криптографию, чтобы украсть биткойны.

Таким образом, мы делаем вывод, что эта атака невозможна для 51-процентного злоумышленника.

Следует отметить, что все это всего лишь мысленный эксперимент.

Если бы были фактические признаки 51-процентной атаки, то, вероятно, разработчики заметят это и отреагируют на это.

Они будут обновлять программное обеспечение Bitcoin, и мы могли бы ожидать, что правила системы, включая одноранговую сеть, могут измениться в той или иной форме, чтобы затруднить успешную атаку.

Но мы не можем этого предсказать. Таким образом, мы работаем в упрощенной модели, где 51-процентная атака происходит, но нет никаких изменений или настроек правил системы.

Давайте рассмотрим еще одну атаку.

Может ли 51-процентный злоумышленник сдерживать некоторые транзакции?

Скажем, есть некий пользователь, Кэрол, которого нападающий очень не любит.

Злоумышленник знает некоторые адреса Кэрол и хочет сделать так, чтобы монеты, принадлежащие одному из этих адресов, не смогли быть потрачены.

Это возможно? Поскольку он контролирует консенсусный процесс блочной цепи, злоумышленник может просто отказываться создавать любые новые блоки, содержащие транзакции с одного из адресов Кэрол.

Злоумышленник может также отказаться не только от создания, но и от использования блоков, содержащих такие транзакции.

Однако он не может запретить передачу этих транзакций в одноранговую сеть, поскольку сеть не зависит от цепочки блоков или от консенсуса, и мы предполагаем, что злоумышленник не полностью контролирует сеть.

Злоумышленник не может остановить передачу транзакции большинству узлов.

Может ли злоумышленник изменить вознаграждение блока?

То есть, может ли злоумышленник притвориться, что награда за блок, вместо 12.5 биткойнов, составляет 100 биткойнов?

Это изменение правил системы, и поскольку злоумышленник не контролирует копии программного обеспечения Bitcoin, на котором работают все честные узлы, это также невозможно.

Это похоже на то, почему злоумышленник не может включать недопустимые транзакции.

Другие узлы просто не узнают про увеличение вознаграждения блока, а атакующий, таким образом, не сможет его потратить.

Наконец, может ли атакующий каким-то образом уничтожить доверие к Биткойну?

Давайте представим, что произойдет.

Если бы было много попыток двойной траты, ситуаций, в которых узлы не расширяли бы самую длинную действующую ветвь и другие атаки, тогда люди, скорее всего, решат, что Биткойн больше не работает как децентрализованная книга, которой они могут доверять.

Люди потеряют уверенность в валюте, и мы можем ожидать, что обменный курс биткойнов резко упадет.

На самом деле, если будет известно, что существует сторона, которая контролирует 51 % хэш-мощности, тогда возможно, что люди потеряют уверенность в биткойне, даже если злоумышленник не обязательно попытается атаковать.

Таким образом, это не только возможно, но и на самом деле вероятно, что 51-процентный злоумышленник любого типа уничтожит доверие к валюте.

Действительно, это основная практическая угроза, что когда-либо появится 51 процентный владелец хэш-мощности.

Никакие другие атаки не представляют угрозу, учитывая объем расходов, которые злоумышленник должен понести, чтобы атаковать Биткойн и достичь 51-процентного большинства, эти атаки не имеют смысла с финансовой точки зрения.

Надеюсь, теперь вы получили хорошее представление о том, как в биткойне достигается децентрализация, как идентификаторы работают в Bitcoin, как распространяются и проверяются транзакции, роль одноранговой сети в Bitcoin, как цепочка блоков используется для достижения консенсуса, а также то, как работает головоломка и майнинг.

Дальше мы будет рассматривать детали и нюансы биткойна.

Bitcoin транзакции

Далее мы рассмотрим механизм Биткойна более детально.

Мы рассмотрим реальные структуры данных и реальные скрипты.

Для начала вспомним, где мы остановились в прошлый раз.

Механизм консенсуса биткойнов дает нам реестр только для добавления, поэтому мы можем только записывать структуры данных, и как только данные записываются, они существуют навсегда.

И есть децентрализованный протокол для установления консенсуса относительно значений этого реестра.

И есть майнеры, которые выполняют этот протокол и которые проверяют транзакции, все вместе гарантируя, что транзакции хорошо сформированы, что нет двойных расходов, и в конечном итоге, что этот реестр и сеть могут функционировать как валюта.

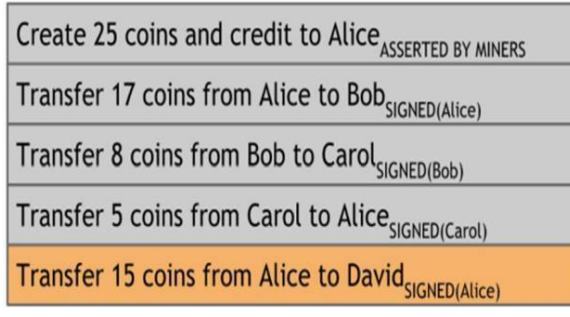
И все потому, что мы предположили, что эта валюта может мотивировать этих майнеров.

Теперь, давайте в деталях рассмотрим, что представляет собой транзакция в биткойне, его центральная часть.

На данный момент мы будем использовать упрощенную модель реестра.

Вместо блоков предположим, что в реестр добавляются

отдельные транзакции по одной за раз.



Как мы можем построить валюту на основе такой книги или реестра?

Первая модель, о которой вы могли бы подумать, которая на самом деле является ментальной моделью, которую многие люди используют для понимания того, как Биткойн работает, заключается в том, что у вас есть система на основе учетной записи.

Вы можете добавить некоторые транзакции, которые создают новые монеты и передают их кому-либо.

А затем позже вы можете эти монеты передавать дальше.

Транзакция будет содержать что-то вроде «мы передаем 17 монет от Алисы к Бобу», и эта транзакия будет подписана

Алисой.

И эта информация о транзакции будет содержаться в книге.

В этом примере, после того как Алиса получила 25 монет в первой транзакции, затем передает 17 монет Бобу во второй транзакции, и у нее осталось еще 8 биткойнов на ее счете.

Недостатком этого способа ведения реестра является то, что любой, кто хочет определить, действительно ли транзакция валидна, должен будет отслеживать эти остатки на счетах.

В этом примере, что нужно сделать, чтобы понять, есть ли у Алисы 15 монет, которые она пытается передать Дэвиду?

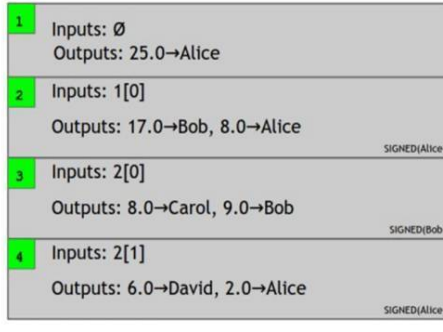
Для этого вам нужно будет просмотреть всю книгу назад во времени, чтобы увидеть каждую транзакцию, относящуюся к Алисе, и вычислить ее баланс на момент, когда она пытается передать 15 монет Дэвиду.

Конечно, мы можем сделать это немного более эффективно с помощью дополнительной структуру данных, которая отслеживает баланс Алисы после каждой транзакции.

Но это потребует большого количества дополнительного обслуживания, кроме обслуживания самой книги.

Поэтому Bitcoin не использует модель на основе учетной записи.

Вместо этого Bitcoin использует книгу, которая просто отслеживает транзакции, подобные ScroogeCoin.



Транзакции указывают количество входов и количество выходов.

Вы можете думать о входах как о монетах, которые потребляются и которые были созданы в предыдущей транзакции, и о выходах как о создаваемых монет.

Для транзакций, в которых создаются совершенно новые монеты, нет монет, которые потребляются.

Каждая транзакция имеет уникальный идентификатор.

Выходы индексируются, начиная с 0.

В этом примере, первая транзакция не имеет входа, потому что эта транзакция создает новые монеты, и у нее есть выход из 25 монет, отправляемых Алисе.

Кроме того, поскольку это транзакция, в которой созда-

ются новые монеты, здесь подпись не требуется.

Теперь предположим, что Алиса хочет отправить некоторые из этих монет Бобу.

Для этого она создает новую транзакцию, вторую транзакцию в нашем примере.

В транзакции она должна явно ссылаться на предыдущую транзакцию, в которой эти монеты поступили к ней.

Здесь она ссылается на выход 0 транзакции 1 (единственный выход транзакции 1), который присвоил Алисе 25 биткоинов.

Она также должна указать выходные адреса в этой транзакции.

В этом примере Алиса указывает два выхода, 17 монет Бобу и 8 монет Алисе.

И, конечно же, эта транзакция подписывается Алисой, так что мы знаем, что Алиса разрешает эту транзакцию.

Почему Алиса должна отправить деньги самой себе в этом примере?

Так как монеты являются неизменяемыми, в биткойне весь вывод транзакции должен потребляться другой транзакцией.

То есть транзакция должна быть самодостаточной, чтобы не калькулировать балансы.

Алиса хочет заплатить Бобу только 17 биткойнов, но выход, который у нее есть, стоит 25 биткойнов.

Поэтому ей нужно создать еще один выход, где 8 биткой-

нов отправляются обратно ей.

Это может отличаться от адреса, которому принадлежат 25 биткойнов, но он должен принадлежать ей. Это называется изменение адреса или change address.

Когда новая транзакция добавляется в реестр, как можно легко проверить, что она является валидной?

В нашем примере нам нужно найти выход транзакции, на который ссылается Алиса, и убедиться, что он имеет значение 25 биткойнов и что он еще не был потрачен.

Найти выход транзакции легко, так как мы используем хэш указатели.

Чтобы убедиться, что этот выход не был потрачен, нам нужно отсканировать цепочку блоков между указанной транзакцией и последним блоком.

Нам не нужно проходить весь путь назад к началу цепочки блоков, и это не требует хранения каких-либо дополнительных структур данных, хотя, как мы увидим, дополнительные структуры данных ускорят работу.

Так как транзакции могут иметь много входов и много выходов, разделять и объединять значения легко.

Например, Боб получил деньги в двух разных транзакциях – 17 биткойнов в одной и 2 биткойна в другой.

Боб может сказать, что хотел бы иметь одну транзакцию, которую он может потратить позже, где у него будут все 19 биткойнов.

Сделать это легко – он создает транзакцию с двумя входа-

ми и одним выходом, причем выходной адрес принадлежит ему.

Это позволяет ему консолидировать эти две транзакции.

Также, легко сделать и совместные платежи.

Предположим, Кэрол и Боб оба хотят заплатить Дэвиду.

Они могут создать транзакцию с двумя входами, которые принадлежат разным людям, и одним выходом.

И единственное отличие от предыдущего примера состоит в том, что, поскольку два выхода из предыдущих транзакций, которые здесь заявляются, относятся к разным адресам, для новой транзакции потребуется две отдельные подписи: одна – Кэрол, а другая – Боба.

Концептуально это все, что связано с транзакцией биткойнов.

Теперь посмотрим, как она представлена на низком уровне в биткойне.

В конечном счете, каждая структура данных, которая отправляется в сеть, представляет собой строку бит.

То, что здесь показано, является низкоуровневым форматом, но далее это дополнительно компилируется до компактного двоичного формата, который не читается человеком.

Как вы можете видеть в этом примере, транзакция делится на три части: некоторые метаданные, серия входов и серия выходов.

Что качается метаданных, здесь есть некоторая информация о самой транзакции – размер транзакции, количество входов и количества выходов.

Также здесь указан хэш всей транзакции, который служит уникальным идентификатором транзакции.

Это позволяет нам использовать хеш-указатели для ссы-

лок на транзакции.

Наконец, есть поле «lock_time», к которому мы вернемся позже.

Теперь о входах.

Входы транзакций образуют массив, и каждый вход имеет один и тот же формат.

Вход указывает предыдущую транзакцию с помощью хэша этой транзакции, который работает как хэш-указатель на предыдущую транзакцию.

Вход также содержит индекс выхода предыдущей транзакции, на которую идет ссылка.

И здесь еще есть подпись.

Помните, что мы должны подписать транзакцию, чтобы показать, что мы на самом деле имеем возможность претендовать на эти предыдущие выходы транзакций.

Теперь о выходах.

Выходы также представляют собой массив.

Каждый выход имеет только два поля. У каждого выхода есть значение, а сумма всех выходных значений должна быть меньше или равна сумме всех входных значений.

Если сумма выходных значений меньше суммы входных значений, разница представляет собой плату за транзакцию для майнера, который публикует эту транзакцию.

Также, у выхода есть строка, которая выглядит как адрес получателя.

Мы видим, что здесь есть хэш публичного ключа, а также

есть набор команд.

Так что это поле на самом деле является скриптом.

Bitcoin скрипты

Каждый выход транзакции не просто указывает публичный ключ или адрес следующего получателя монет.

На самом деле он определяет скрипт.

Что такое скрипт и почему мы используем скрипты?

Далее мы рассмотрим язык Bitcoin скриптов и поймем, почему скрипт используется вместо простого указания открытого ключа.

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY
OP_CHECKSIG
```

Наиболее распространенным типом транзакции в биткойне является трата выхода предыдущей транзакции путем

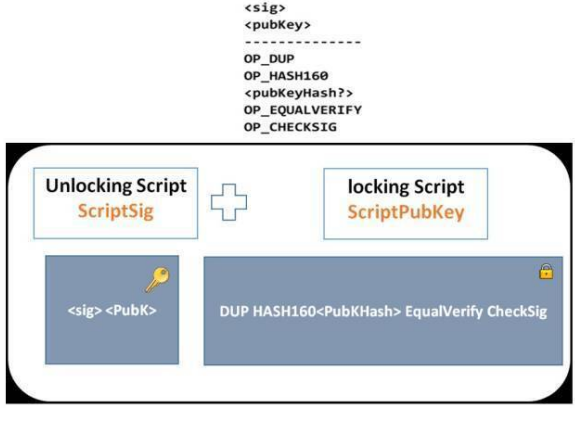
подписания с помощью правильного ключа.

В этом случае мы хотим, чтобы на выходе транзакции говорилось: «этот выход транзакции может быть потрачен с помощью подписи следующего владельца указанного адреса.»

Напомним, что адрес является хешем публичного ключа.

Поэтому просто указание адреса не дает нам публичный ключ, и не дает нам возможности проверить подпись!

Таким образом, вместо этого выход транзакции говорит нам: «этот выход транзакции может быть потрачен публичным ключом, который хешируется, а также подписью владельца этого публичного ключа».



Теперь, что происходит с этим скриптом?

Кто его запускает, и как именно эта последователь-

ность инструкций обеспечивает соблюдение вышеуказанного утверждения?

Секрет в том, что входы также содержат скрипты вместо просто подписей.

Чтобы проверить, что новая транзакция правильно потребляет выход предыдущей транзакции, мы объединяем входной скрипт новой транзакции и выходной скрипт предыдущей транзакции.

Мы просто соединяем их вместе, и полученный скрипт должен успешно выполниться, чтобы новая транзакция была действительной.

Эти два скрипта называются scriptPubKey и scriptSig, потому что в простейшем случае выходной скрипт просто указывает хэшированный публичный ключ или адрес, который может потребить этот выход транзакции, а входной скрипт следующей транзакции указывает подпись с этим публичным ключом.

Bitcoin язык скриптов был создан специально для биткойнов и называется просто «Скрипт».

Он имеет много общего с языком под названием Forth, который является старым, простым, основанным на стеках языком программирования.

Но вам не нужно изучать Форт, чтобы понимать скрипты биткойнов.

Язык Script был создан, чтобы иметь что-то простое и компактное, но с собственной поддержкой криптографиче-

ских операций.

Поэтому в нем существуют специальные инструкции для вычисления хеш-функций, а также для вычисления и проверки подписей.

Язык Script основан на стеках.

Это означает, что каждая инструкция выполняется ровно один раз, линейно.

В частности, в языке Script биткойнов нет циклов.

Таким образом, количество инструкций в скрипте дает нам верхнюю оценку того, сколько времени потребуется для запуска скрипта и сколько памяти он может использовать.

Этот язык не имеет возможности вычислять произвольно мощные функции.

И по замыслу, именно майнеры должны запускать эти скрипты, которые предоставляются произвольными участниками сети.

Поэтому мы не хотим дать этим произвольным участникам возможность представить сценарий, который может иметь бесконечный цикл.

Таким образом, чтобы проверить, правильно ли новая транзакция потребляет выход предыдущей транзакции, мы создаем комбинированный скрипт, добавляя скрипт scriptPubKey предыдущей транзакции снизу к скрипту scriptSig новой транзакции.

Обратите внимание, что <pubKeyHash?> содержит ?'.

Мы используем это обозначение, чтобы указать, что мы

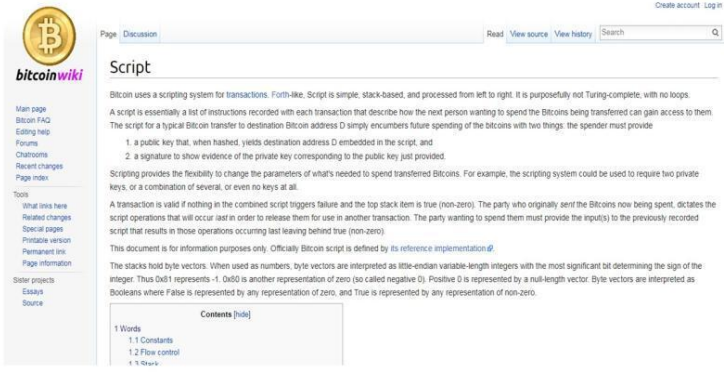
позже проверим, что это значение равно хешу публичного ключа, который предоставлен во входном скрипте.

Есть только два возможных результата при выполнении скрипта биткойнов.

Он либо успешно выполняется без ошибок, и в этом случае транзакция действительна.

Или, если во время выполнения скрипта есть какая-либо ошибка, тогда вся транзакция будет недействительной и не должна приниматься в цепочку блоков.

Язык скриптов биткойнов очень маленький.



В нем есть только 256 инструкций, и каждая из них представлена одним байтом.

Байт состоит из восьми бит. Используя один байт, можно

закодировать один символ из 256 возможных (256 = 2 в 8 степени). Таким образом, один байт равен одному символу, то есть 8 битам.

Из этих 256 инструкций, 15 в настоящее время отключены, и 75 зарезервированы.

Зарезервированные инструкции еще не получили никакого специального значения.

Многие из основных инструкций – это те, которые вы ожидаете в любом языке программирования.

Там есть базовая арифметика, базовая логика, такая как ‘if’ и ‘then, выброс ошибки, возврат return.

Наконец, существуют криптографические инструкции, которые включают хеш-функции, инструкции для проверки подписи, а также специальную и важную инструкцию CHECKMULTISIG, которая позволяет проверять несколько подписей в одной инструкции.

OP_DUP	Duplicates the top item on the stack
OP_HASH160	Hashes twice: first using SHA-256 and then RIPEMD-160
OP_EQUALVERIFY	Returns true if the inputs are equal. Returns false and marks the transaction as invalid if they are unequal
OP_CHECKSIG	Checks that the input signature is a valid signature using the input public key for the hash of the current transaction
OP_CHECKMULTISIG	Checks that the *k* signatures on the transaction are valid signatures from *k* of the specified public keys.

Для инструкции CHECKMULTISIG требуется указать n открытых ключей и параметр t как пороговое значение.

Чтобы эта инструкция выполнялась корректно, должно быть не менее t подписей от t из n этих открытых ключей, которые действительны. В этой инструкции мы можем определить компактным образом, что t из n указанных открытых ключей должны дать правильные подписи, чтобы транзакция была действительной.

Позже мы рассмотрим несколько примеров того, как используются мультиподписи.

Кстати, существует ошибка в реализации мультиподписи, и она была там все время.

Инструкция CHECKMULTISIG выталкивает значение дополнительных данных из стека и игнорирует его.

Это всего лишь причуда языка биткойнов, и с этим приходится иметь дело, добавляя в стек дополнительную фиктивную переменную.

Ошибка была в первоначальной реализации, и затраты на ее исправление намного выше, чем причиненный ущерб, как мы увидим позже.

На данный момент эта ошибка считается просто особенностью биткойна.

Чтобы выполнить скрипт на стековом языке программирования, все, что нам понадобится, это стек, в который мы можем вносить данные и из которого можем извлекать данные.

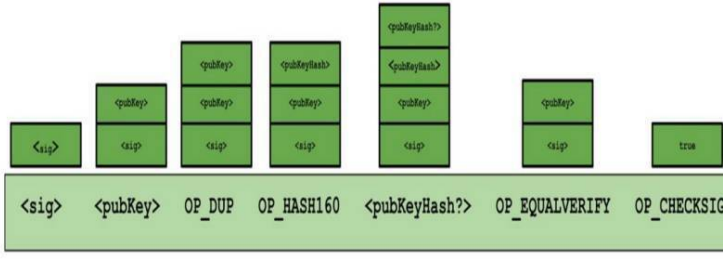
Нам не нужна никакая дополнительная память или переменные.

Это делает скрипт простым в вычислении.

В скрипте существует два типа инструкций: инструкции данных и коды операций или опкоды.

Когда в скрипте появляется инструкция данных, эти данные просто вставляются в верхнюю часть стека.

С другой стороны, опкоды выполняют некоторую функцию, часто беря данные, находящиеся вверху стека, как входные данные.



Теперь давайте посмотрим, как выполняется Bitcoin скрипт.

Здесь мы показываем состояние стека после каждой инструкции.

Первые две инструкции в этом скрипте – это инструкции данных – подпись и публичный ключ этой подписи.

Они были указаны в элементе scriptSig или входном скрипте.

Как мы уже сказали, когда мы видим инструкцию данных, мы просто вносим данные в стек.

Дальше идет скрипт scriptPubKey.

Здесь сначала у нас есть команда дублирования OP_DUP, поэтому мы просто вносим копию публичного ключа в верхнюю часть стека.

Следующей инструкцией является OP_HASH160, в которой говорится, что нужно вытолкнуть из стека верхнее значение, вычислить его криптографический хеш и внести результат в верхнюю часть стека.

Когда эта команда завершит выполнение, мы заменим публичный ключ на вершине стека его хешем.

Здесь речь идет о публичном ключе текущего владельца биткойнов.

Затем мы вносим в стек хэш публичного ключа, который был указан в предыдущей транзакции как получатель монет и который должен использоваться для создания подписи, чтобы потратить полученные монеты.

Таким образом, на данный момент в верхней части стека есть два значения.

Существует хэш публичного ключа, который был указан в выходном скрипте, и хэш публичного ключа, который используется при трате монет и который указан во входном скрипте.

На этом этапе мы запускаем команду EQUALVERIFY, которая проверит, что эти два значения в верхней части стека равны.

Если это не так, произойдет ошибка, и скрипт прекратит выполнение.

В нашем примере мы будем считать, что они равны, то есть получатель монет использовал правильный публичный ключ.

Эта инструкция потребляет те два элемента данных, которые находятся в верхней части стека.

И теперь стек содержит два элемента – подпись и публичный ключ, который использовался для этой подписи.

Мы уже проверили, что этот публичный ключ является публичным ключом, который требуется, и теперь мы должны проверить, действительна ли подпись.

Это отличный пример того, как язык скриптов Bitcoin построен с учетом криптографии.

Несмотря на то, что это довольно простой язык с точки зрения логики, в нем есть некоторые довольно сильные инструкции, такие как инструкция «OP_CHECKSIG».

Эта инструкция выталкивает эти два значения из стека и выполняет всю проверку подписи за один раз.

Но чего это подпись?

Какой был вход функции подписи?

Оказывается, есть только одна вещь, которую вы можете подписать в биткойн – это целая транзакция.

Таким образом, инструкция «CHECKSIG» выталкивает из стека два значения, открытый ключ и подпись, и проверяет, является ли эта подпись валидной для всей транзакции, используя этот публичный ключ.

Теперь мы выполнили каждую инструкцию в скрипте, и в стеке ничего не осталось.

Если ошибок не было, выход этого скрипта будет просто true, указывая, что транзакция действительна.

Теоретически, скрипт позволяет нам в каком-то смысле указать произвольные условия, которые должны быть выполнены для того, чтобы потратить монеты.

Но на сегодняшний день эта гибкость практически не используется.

Если мы посмотрим на скрипты, которые на самом деле были использованы в истории Биткойна, подавляющее большинство, 99,9 %, – это точно такой же скрипт pay-to-public-key-hash, который мы использовали в нашем примере.

Как мы видели, этот скрипт pay-to-public-key-hash просто указывает один публичный ключ, вернее его хэш, и требует подписи для этого публичного ключа, чтобы потратить монеты.

Однако существуют несколько других инструкций, которые действительно полезны.

Иногда используется специальный тип скрипта под названием «Pay-to-Script-Hash», который обрабатывает мультиподписи MULTISIG и который мы обсудим позже.

Вообще говоря, не существует большого разнообразия используемых скриптов.

Это связано с тем, что узлы биткойнов по умолчанию имеют белый список стандартных скриптов, и они отказываются принимать скрипты, отсутствующие в списке.

Это не означает, что эти другие скрипты нельзя использовать вообще; это просто усложняет их использование.

На самом деле это различие – это очень тонкая вещь, к

которой мы вернемся, когда мы будем говорить об одноранговой сети Bitcoin.

Далее рассмотрим несколько видов стандартных скриптов.

Proof of burn доказательство сжигания – это скрипт, в котором биткойны никогда не могут быть потрачены.

Отправка монет в скрипт с доказательством сжигания устанавливает, что они уничтожены, так как нет никакой возможности для их расходования.

Одно из использований доказательства сжигания заключается в том, чтобы загрузить альтернативу биткойну, заставив людей уничтожить биткойн, чтобы получить монеты в новой системе.

Мы обсудим это более подробно позже.

Доказательство сжигания довольно просто реализовать: опкод OP_RETURN выбрасывает ошибку и маркирует транзакцию как недействительную.

Таким образом, любая новая транзакция, которая попытается использовать выход с OP_RETURN, будет недействительной и не будет учитываться в блокчейне.

Независимо от того, какие значения вы ставите перед OP_RETURN, эта инструкция будет выполнена и скрипт вернет false.

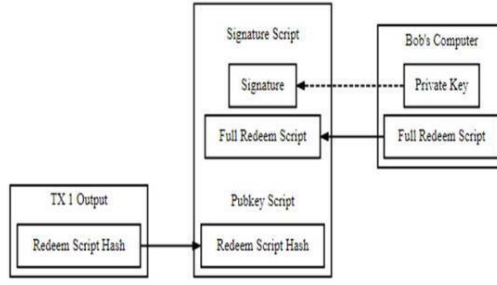
Так как выбрасывается ошибка, данные в скрипте, которые появляются после OP_RETURN, не будут обрабатываться.

Таким образом, это также возможность помещать произвольные данные в скрипт и, следовательно, в цепочку блоков.

Если по какой-то причине вы хотите написать свое имя или хотите установить отметку времени и доказать, что знаете определенные данные в определенное время, тем самым, например, внести доказательство авторских прав на документ, вы можете создать транзакцию биткойнов с очень малой суммой и инструкцией OP_RETURN.

Вы можете уничтожить очень маленькую сумму валюты, но вы можете написать все, что захотите, в цепочку блоков, которая будет храниться всегда.

Теперь о скрипте Pay-to-script-hash.



Механизм работы скриптов Биткойна подразумевает, что

отправитель монет должен точно указать скрипт.

Но это иногда становится затруднительным.

Скажем, например, вы являетесь покупателем интернет-магазина, и вы собираетесь что-то заказать.

И вы говорите: «Хорошо, я готов заплатить. Скажите мне адрес, на который я должен отправить свои монеты».

Теперь предположим, что компания, в которой вы заказываете товар, использует адрес MULTISIG с несколькими приватными ключами, то есть для их подписи используются несколько приватных ключей для дополнительной защиты.

Затем, так как вы должны указать это, продавец должен будет сказать вам: «Мы используем MULTISIG, и мы попросим вас отправить монеты с использованием сложного скрипта».

На что вы могли бы сказать: «Я не знаю, как это сделать. Это слишком сложно. Как потребитель, я просто хочу отправить монеты на простой адрес».

Для решения этой проблемы, в биткойне есть умный хак.

Вместо того, чтобы отправитель указывал весь скрипт, отправитель может указать только хэш скрипта, который понадобится для последующей траты этих монет.

Поэтому отправителю нужно просто указать очень простой скрипт, который просто хэширует верхнее значение в стеке и проверяет, соответствует ли оно требуемому скрипту траты монет.

Получателю этих монет необходимо указать в качестве

значения данных, сам скрипт, чей хэш указан отправителем.

После этого произойдет второй этап проверки.

То есть, верхнее значение данных из стека будет переинтерпретировано в качестве инструкций, а затем оно будет выполняться во второй раз уже как скрипт.

Итак, мы видим, что здесь выполняются два этапа.

Сначала это был традиционный скрипт, который проверял, что скрипт потребления монет имеет правильный хеш.

И после этого скрипт потребления монет де-сериализуется и запускается как скрипт.

И вот где будет происходить фактическая проверка подписи.

Создание поддержки для этого типа скриптов P2SH было довольно сложным, поскольку этот тип скриптов не был частью первоначальной спецификации Bitcoin.

Он был добавлен позже.

Это, вероятно, самая известная функция, добавленная в биткойн, которой не было в исходной спецификации.

И она решает несколько важных проблем.

Она облегчает жизнь отправителю, так как получатель может просто указать хэш, на который отправитель отправляет деньги.

В нашем примере, Алисе не нужно беспокоиться о том, что Боб использует multisig, она просто отправляет деньги на адрес P2SH Боба, и ответственность Боба заключается в том, чтобы указать этот сложный скрипт, когда он хочет по-

тратить монеты.

P2SH также ускоряет обработку.

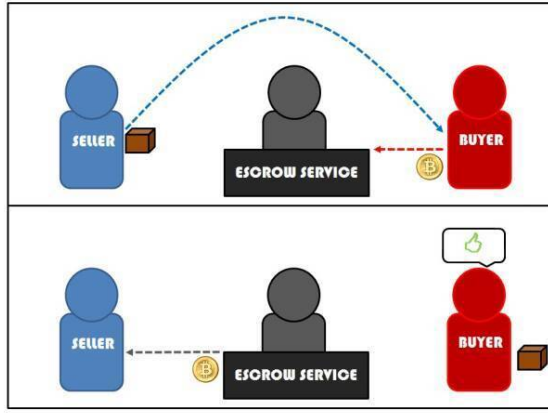
Так, майнерам нужно отслеживать набор выходных скриптов, которые еще не были потрачены, а с P2SH выходные скрипты намного меньше, так как они указывают только хеш.

Вся сложность переносится на входные скрипты.

Теперь, когда мы понимаем, как работают скрипты Bitcoin, давайте взглянем на некоторые из применений, которые могут быть реализованы с помощью этого языка сценариев.

Оказывается, что мы можем делать много полезных вещей, которые оправдывают использование языка сценариев вместо того, чтобы просто указывать публичные ключи.

Рассмотрим транзакцию условного депонирования – это депонирование определенной суммы покупателем у третьего лица под определенные условия при сомнениях в добросовестности продавца.



Скажем, Алиса и Боб хотят вести дела друг с другом – Алиса хочет заплатить Бобу в Биткойнах, чтобы Боб отправил некоторый физический товар Алисе.

Проблема в том, что Алиса не хочет платить до тех пор, пока она не получит товар, а Боб не хочет отправлять товар до тех пор, пока он не будет оплачен.

Что мы можем с этим поделать?

Хорошим решением в биткойне, которое можно было бы использовать на практике, является введение третьей стороны и транзакция с депонированием.

Транзакция с депонированием может быть реализована достаточно просто с использованием мультиподписи MULTISIG.

Алиса не отправляет деньги непосредственно Бобу, а вместо этого создает транзакцию MULTISIG, для которой требуется два из трех человек, чтобы потратить монеты.

И эти три человека будут Алисой, Бобом, и некоторым сторонним арбитром Джуди, который вступит в игру, если возникнут какие-либо споры.

Поэтому, Алиса создает 2-из-3 транзакцию MULTISIG, которая отправляет некоторое количество монет, которыми она владеет, и указывает, что они могут быть потрачены, если будут две подписи из трех Алисы, Боба и Джуди.

Эта транзакция включена в цепочку блоков, и на данный момент эти монеты хранятся в депонировании между Алисой, Бобом и Джуди, так что любые два из них могут указать, куда должны уйти монеты.

В этот момент Боб убежден, что безопасно отправить товар Алисе, поэтому он отправит его по почте или каким-то другим способом.

Теперь, предположим, Алиса и Боб оба честны.

Поэтому Боб отправит товар, который ожидает Алиса, и когда Алиса получит товар, Алиса и Боб подпишут транзакцию, потратив средства из депонирования и отправив их Бобу.

Обратите внимание, что в этом случае, когда Алиса и Боб честны, Джуди не нужно вмешиваться.

Не было никакого спора, и подписи Алисы и Боба соответствовали требованию 2-из-3 транзакции MULTISIG.

Так что в нормальном случае это ничем не отличается, как если бы Алиса просто отправила бы Бобу деньги, но это требует одной дополнительной транзакции.

Но что могло бы произойти, если бы Боб не отправил товар или товар потерялся бы на почте?

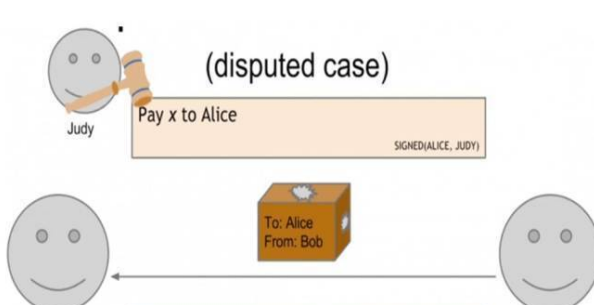
Или, может быть, товар отличался бы от того, который заказывала Алиса?

Алиса теперь не хочет платить Бобу, потому что думает, что ее обманули, и она хочет вернуть свои деньги.

Поэтому Алиса определенно не собирается подписывать транзакцию, которая передает деньги Бобу.

Но Боб также может отрицать любые нарушения и отказываться подписывать транзакцию, которая возвращает деньги Алисе.

Здесь необходимо принять участие Джуди.



Джуди придется решить, кто из этих двух людей заслуживает денег.

Если Джуди решит, что Боб обманул, Джуди подпишет сделку вместе с Алисой, отправив деньги с эсквота обратно Алисе.

Подписи Алисы и Джуди отвечают требованиям 2-из-3 транзакции MULTISIG, и Алиса вернет себе деньги.

И, конечно, если Джуди думает, что здесь виновата Алиса, и Алиса просто отказывается платить, Джуди может подписать транзакцию вместе с Бобом, отправив деньги Бобу.

Поэтому в этом случае Джуди решает, какой будет результат.

Но ей не придется участвовать, если нет спора.

Еще одно интересное применение скриптов – это то, что называют зелеными адресами.

Предположим, Алиса хочет заплатить Бобу, но Боб не в сети.

Так как он не в сети, Боб не может взять и посмотреть на цепочку блоков, чтобы увидеть, находится ли там транзакция, которую послала Алиса.

Также возможно, что Боб находится в сети, но у него нет времени, чтобы посмотреть на цепочку блоков и дождаться подтверждения транзакции.

Помните, что обычно мы ждем, чтобы транзакция попала в цепочку блоков и была подтверждена шестью последующими блоками, что занимает около часа, прежде чем мы будем уверены, что транзакция действительно находится в цепочке блоков.

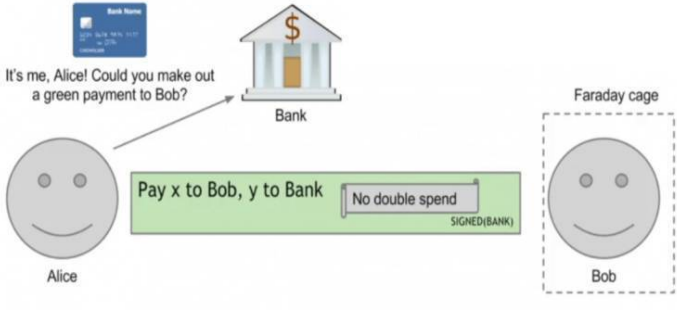
Но для некоторых товаров, таких как еда, Боб не может ждать час, чтобы начать доставку.

Если бы Боб был уличным продавцом, продающим хотдоги, маловероятно, что Алиса будет ждать около часа, чтобы получить еду.

Или, возможно, Боб по какой-либо другой причине вообще не имеет никакого подключения к Интернету и, следовательно, не сможет проверить цепочку блоков.

Чтобы решить эту проблему, чтобы вы могли отправлять деньги с помощью биткойна, не получая доступ к блокчей-

ну, мы должны предоставить другую третью сторону, которую мы будем называть банком (на практике это может быть любой финансовый посредник).



Тогда Алиса связывается с ее банком и говорит: «Это я, Алиса. Я твой лояльный клиент. Вот моя карточка или мой идентификатор. И я бы хотела заплатить Бобу, не могли бы вы мне помочь?»

На что банк говорит: «Конечно. Я спишу эту сумму с вашего счета. И составлю транзакцию с одного из моих зеленых адресов на Боба».

Поэтому обратите внимание, что эти деньги поступают напрямую от банка к Бобу.

Какая-то сумма, конечно, может быть в обратном адресе,

возвращающемся в банк.

Но, по сути, банк платит Бобу с банковского адреса, который мы называем зеленым адресом.

Более того, банк гарантирует, что он не будет дважды тратить эти деньги.

И, как только Боб видит, что эта транзакция подписана банком, если он доверяет гарантии банка не делать двойную трату этих денег, он может заранее принять то, что в конечном итоге это будут его деньги, когда транзакция будет подтверждена в цепочке блоков.

Обратите внимание, что это не гарантия, основанная на биткойнах.

Это реальная гарантия, и для того, чтобы эта система работала, Боб должен верить, что банк в реальном мире заботится о своей репутации и не будет по этой причине делать двойные траты.

И банк сможет сказать: «Вы можете посмотреть на мою историю. Я давно использую этот зеленый адрес, и я никогда не совершал по нему двойной траты. Поэтому я вряд ли сделаю это в будущем».

Таким образом, Бобу больше не нужно думать о доверии Алисе, о которой он ничего не знает.

Вместо этого он доверяет банку, что они не будут дважды тратить деньги, которые они ему отправили.

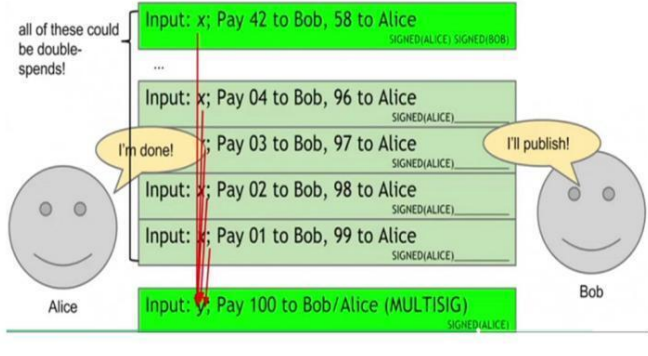
Конечно, если банк когда-либо сделает двойную трату, люди перестанут доверять этим зеленым адресам.

Фактически, двумя наиболее известными онлайн-службами, которые реализовали зеленые адреса, были Instawallet и Mt. Gox, и обе в итоге закрылись, так как совершили двойную трату.

Сегодня зеленые адреса практически не используются.

Когда идея была впервые предложена, она привлекла внимание как способ сделать платежи быстрее и без доступа к цепочке блоков.

Затем, однако, люди разочаровались в ней по причине того, что она требует слишком большого доверия банку.



Третий пример применения скриптов биткойнов – это способ совершать эффективные микроплатежи.

Предположим, что Алиса – клиент, который хочет посто-

янно платить Бобу небольшую сумму денег за некоторую услугу, которую предоставляет Боб.

Например, Боб может быть поставщиком услуг мобильной связи для Алисы, и требует, чтобы она платила небольшую плату за каждую минуту, которую она проговорила со своего телефона.

Создание транзакции биткойнов для каждой минуты, которую Алиса проговорит по телефону, не будет работать.

Это создаст слишком много транзакций с комиссией за каждую транзакцию.

Поэтому, учитывая комиссию за каждую транзакцию, плата Алисы за услуги будет слишком высокой.

Поэтому нам хотелось бы, чтобы все эти небольшие платежи были объединены в один большой платеж в конце.

Оказывается, существует приемлемый способ это сделать.

Мы начинаем с транзакции, которая платит максимальную сумму на адрес MULTISIG, сумму которую Алисе когда-либо потребуется потратить, и этот адрес MULTISIG требует подписи как Алисы, так и Боба, чтобы разблокировать эти монеты.

Теперь, после первой истраченной минуты, когда Алиса использовала услугу, или, когда в первый раз Алисе нужно сделать микроплатеж, она подписывает транзакцию, тратя те монеты, которые были отправлены на адрес MULTISIG, отправляя одну единицу платежа Бобу и возвращая остальные монеты Алисе. После следующей использованной мину-

ты Алиса подписывает еще одну транзакцию, на этот раз отдавая Бобу две единицы и отправляя остальную часть себе.

Обратите внимание, что эти транзакции подписаны только Алисой и еще не подписаны Бобом, и они не публикуются в блокчейне.

И Алиса будет продолжать отправлять эти транзакции Бобу каждую минуту, когда она использует эту услугу.

В конце концов, Алиса закончит использовать эту услугу и сообщит Бобу: «Я закончила, пожалуйста, прекратите мое обслуживание».

В этот момент Алиса прекратит подписывать дополнительные транзакции.

Услышав это, Боб скажет: «Отлично. Я отключаю ваш сервис, и я беру последнюю транзакцию, которую вы прислали мне, подписываю ее, и публикую ее в цепочке блоков».

Таким образом, последняя транзакция, которую Боб подписывает, выплачивает ему полностью за предоставленную им услугу и возвращает остальную часть денег Алисе.

Все эти промежуточные транзакции, подписанные Алисой, не попадут в цепочку блоков. И Бобу не нужно их подписывать. Они просто будут отброшены.

Технически все эти транзакции подразумевают возможность двойной траты.

Таким образом, в отличие от случая с зелеными адресами, где мы специально пытались избежать двойных трат с большой гарантией, в этом протоколе микроплатежей, мы

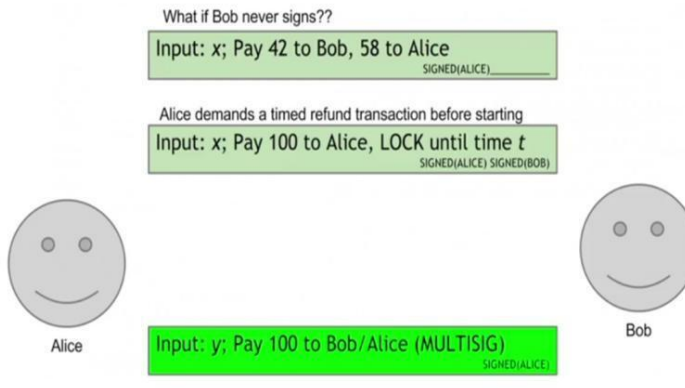
фактически генерируем огромное количество потенциальных двойных трат.

На практике, однако, если обе стороны работают нормально, Боб никогда не будет подписывать какую-либо транзакцию, кроме последней, и в этом случае блокчейн фактически не увидит попыток двойной траты.

Здесь есть еще одна сложность: что, если Боб никогда не подпишет последнюю транзакцию?

Он может просто сказать: «Я хочу, чтобы монеты находились на депонировании навсегда», и в этом случае, монеты не будут расходоваться, но при этом Алиса может потерять сдачу или остаток монет, который она расчитывала получить обратно.

Существует очень умный способ избежать этой проблемы, используя функцию, о которой мы упоминали ранее.



Чтобы избежать этой проблемы, прежде чем начнется протокол микроплатежей, Алиса и Боб оба подпишут транзакцию, которая вернет все деньги Алисы, но возврат будет «заблокирован» до определенного момента времени в будущем.

Поэтому после того, как Алиса подпишет, но прежде чем она транслирует первую транзакцию MULTISIG, которая вложит ее средства в депонирование, она получит эту возвратную транзакцию от Боба и будет удерживать ее.

Это гарантирует, что, если до определенного момента времени Боб не подпишет ни одной из небольших транзакций, которые отправила Алиса, Алиса может опубликовать эту транзакцию, которая вернет все деньги непосредственно ей.

Что это значит, что транзакция заблокирована до опреде-

ленного момента времени?

Вспомните, когда мы смотрели на метаданные в транзакциях биткойнов, там был параметр lock_time, который мы оставили без рассмотрения.

Он работает следующим образом.

Если вы укажете любое значение, отличное от нуля, представляющее время блокировки, это значение сообщает майнерам не публиковать транзакцию до указанного момента времени, или пока определенное количество блоков не будет внесено в блокчейн.

Locktime представляет собой четырехзначное целое число без знака, которое можно обработать двумя способами:

Если это число меньше 500 миллионов, Locktime интерпретируется как высота блока, то есть количество блоков в блокчейне от нулевого до последнего блока.

Транзакция может быть добавлена в любой блок с этой высотой или выше.

Если значение больше или равно 500 миллионам, Locktime интерпретируется с использованием формата времени эпохи Unix (количество секунд, прошедших с 1970-01-01T00: 00 UTC, в настоящее время превышает 1.395 миллиардов).

Транзакция может быть добавлена в любой блок по прошествии этого времени.

Также есть много других примеров использования биткойн скриптов, которые были предложены.

Одно из применений – это многопользовательские лотереи с очень сложным многоступенчатым протоколом с множеством транзакций, имеющих депонирование и разные времена блокировки.

Существуют также некоторые протоколы, в которых используется язык скриптов, позволяющие разным людям объединять свои монеты и смешивать их, так что сложнее проследить, кому принадлежит монеты. Мы рассмотрим это подробнее позже.

Общий термин для протоколов, подобных тем, которые мы рассмотрели, является умные контракты или смарт контракты.

Это контракты, которые традиционно обеспечиваются с помощью законов или арбитражных судов, но для реализации которых у нас есть определенная степень технического обеспечения в Биткойне.

Это замечательная функция Bitcoin, где мы можем использовать скрипты, майнеров и проверку транзакций для реализации, например, протокола депонирования или протокола микроплатежей, не требующих централизованного управления.

Также существует много типов смарт контрактов, которые люди хотели бы использовать, но которые на сегодняшний день не поддерживаются языком скриптов Bitcoin.

Или, по крайней мере, никто не придумал способ их реализации.

Блоки Bitcoin

До сих пор мы рассматривали, как создаются и потребляются отдельные транзакции.

Но, как мы видели раньше, транзакции сгруппированы в блоки.

Зачем это нужно?

В принципе, это ничто иное как оптимизация.

Если бы майнеры должны были бы приходить к консенсусу по каждой транзакции отдельно, скорость, с которой новые транзакции принимались бы системой, была бы намного ниже.

Кроме того, хеш-цепочка блоков намного короче, чем хеш-цепочка транзакций, поскольку в каждый блок может быть помещено большое количество транзакций.

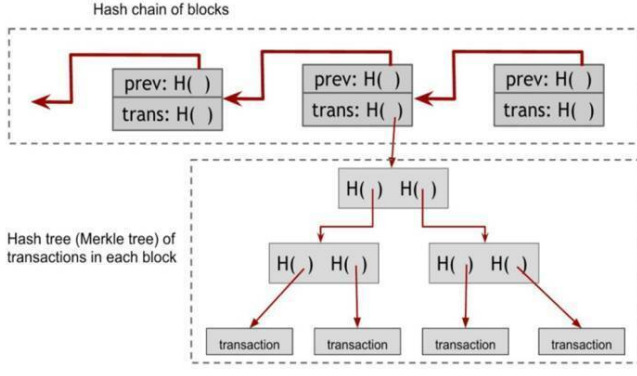
Это позволяет повысить эффективность проверки структуры цепочки блоков.

Каждый блок должен включать одну или несколько транзакций.

Первая из этих транзакций должна быть транзакцией coinbase или транзакцией генерации, которая собирает и тратит вознаграждение за блок, состоящее из вознаграждения за сам блок и любых транзакционных сборов, оплаченных транзакциями, включенными в этот блок.

Цепочка блоков – это умная комбинация двух разных

структур данных на основе хэшей.



Первая структура – это хеш-цепочка блоков.

Каждый блок имеет заголовок блока с хэш-указателем на транзакции и хэш-указателем на предыдущий блок в последовательности.

Вторая структура данных – это дерево всех транзакций для каждого блока, которые включены в этот блок.

Все транзакции, включая транзакцию coinbase, кодируются в блоке в бинарном формате.

Этот формат хэшируется для создания идентификатора транзакции.

Из этих идентификаторов строится дерево merkle путем спаривания каждого идентификатора с одним другим иден-

тификатором и последующим их объединением.

Если существует нечетное число идентификаторов, идентификатор без партнера хешируется с копией самого себя.

Затем результирующие хеши объединяются по парам и хэшируются вместе.

Любой хеш без партнера хэшируется сам с собой.

Процесс повторяется до тех пор, пока не останется только один хеш, или корень merkle.

Таким образом, это дерево Merkle позволяет нам иметь дайджест всех транзакций в блоке эффективным образом.

Как мы видели ранее, чтобы доказать, что транзакция включена в конкретный блок, мы можем предоставить путь через дерево, длина которого – это логарифм числа транзакций в блоке.

Для повторения, блок состоит из заголовка с данными, за которыми следует список транзакций, расположенных в древовидной структуре.

Заголовок блока содержит информацию, связанную с майнингом.

```
02000000 ........................... Block version: 2

b6ff0b1b1680a2862a30ca44d346d9e8
910d334beb48ca0c0000000000000000 ... Hash of previous block's header
9d10aa52ee949386ca9385695f04ede2
70dda20810decd12bc9b048aaab31471 ... Merkle root

24d95a54 ........................... Unix time: 1415239972
30c31b18 ........................... Target: 0x1bc330 * 256**(0x18-3)
fe9f0864 ........................... Nonce
```

Вспомните, что хеш заголовка блока должен начинаться с большого количества нулей для того, чтобы блок был действительным.

Доказательство работы подразумевает, что хэш заголовка блока должен быть меньше указанного.

Другой способ сказать это, что хэш заголовка блока должен начинаться с определенного количества нулей.

Любой блок, заголовок которого не создает хэш, который меньше целевого значения, будет отклонен сетью.

Целевое значение настраивается каждые две недели, чтобы поддерживать среднее время создания нового блока 10 минут.

Заголовок также содержит значение «nonce», которое изменяется майнером, чтобы хэш заголовка блока был меньше

целевого значения.

Заголовок содержит отметку времени и целевое значение, которое определяет трудность нахождения этого блока.

Таким образом, если после того, как вы обработали каждую транзакцию и нашли корень дерева Merkle, добавили его в заголовок блока с хешем предыдущего блока и nonce, хэшировали заголовок и создали значение, которое с правильным количеством нулей не попадает в указанный целевым значением диапазон, тогда изменяется значение nonce, заголовок блока снова хэшируется и так до тех пор, пока хэш заголовка блока не будет меньше целевого значения.

Заголовок – это единственное, что хэшируется во время майнинга.

Чтобы проверить цепочку блоков, нам нужно только посмотреть на заголовки.

Единственными данными транзакций, включенными в заголовок, является корень дерева транзакций – поле «mrkl_root».

```
"in":[
      {
        "prev_out":{
          "hash":"000000.....0000000",
          "n":4294967295
        },
          "coinbase":"..."
      },
      "out":[
  {
    "value":"25.03371419",
    "scriptPubKey":"OPDUP OPHASH160 ... "
        }
```


Как я уже сказал, блок содержит специальную транзакцию в дереве Merkle, называемую транзакцией «coinbase».

Здесь происходит создание новых монет в Биткойне.

Эта транзакция выглядит как обычная транзакция, но имеет несколько отличий:

Она всегда имеет один вход и один выход, при этом вход не тратит предыдущий выход и, следовательно, содержит нулевой хэш указатель, так как он чеканит новые биткойны и не тратит существующие монеты.

Также стоимость выхода в настоящее время составляет около 6,25 биткойнов.

Выходное значение – это доход майнера от блока.

Оно состоит из двух компонентов: награды за добычу блока, которая устанавливается системой, и которая делится по-

полам каждые 210 000 блоков (около 4 лет) и комиссионных сборов, взимаемых с каждой транзакции, включенной в блок.

Также coinbase транзакция содержит специальный параметр coinbase, который абсолютно произволен – майнеры могут записывать туда все, что захотят.

Известно, что в самом первом блоке, добытом в Биткойне, параметр coinbase ссылался на историю в газете Times of London, в которой участвовал канцлер, спасающий банки.

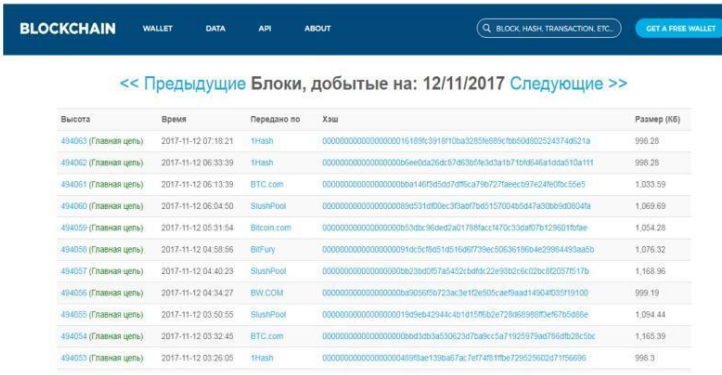
Это интерпретировалось как политический комментарий о мотивации для запуска биткойн.

Это также служит доказательством того, что первый блок был добыт после того, как газета вышла 3 января 2009 года.

Также параметр coinbase может использоваться для уведомления майнерами о поддержке различных новых функций.

Чтобы лучше понять формат блока и формат транзакции, лучше всего изучить цепочку блоков.

Есть много веб-сайтов, которые делают эти данные доступными, например, blockchain.info.



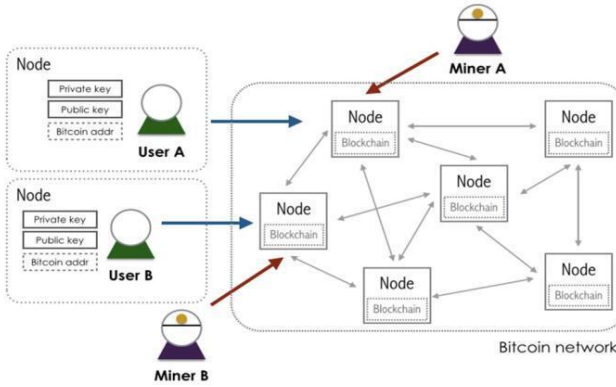
Вы можете посмотреть графики транзакций, посмотреть, какие транзакции тратят другие транзакции, искать транзакции со сложными сценариями и просматривать структуру блоков и видеть, как блоки ссылаются на другие блоки.

Поскольку цепочка блоков представляет собой структуру публичных данных, разработчики создали красивые инструменты для ее графического анализа.

Сеть Bitcoin

До сих пор мы говорили о способности участников публиковать транзакцию и вводить ее в цепочку блоков, как будто это происходит по волшебству.

На самом деле это происходит с помощью сети Биткойн.



Это одноранговая сеть, и она наследует многие идеи одноранговых сетей.

В сети Bitcoin все узлы равны.

Здесь нет иерархии, и нет особых узлов или главных узлов.

Эта сеть работает через протокол TCP и имеет случайную топологию, где каждый узел связан с другими случайными узлами.

Новые узлы могут присоединиться в любое время.

Фактически, сегодня вы можете скачать клиент Bitcoin, развернуть свой компьютер в качестве узла, и он будет иметь равные права и возможности, как и любой другой узел в сети Bitcoin.

Сеть меняется со временем и очень динамична из-за входа и выхода узлов.

Не существует явного способа покинуть сеть.

Вместо этого, если узел не был слышен некоторое время – три часа – это длительность, которая жестко закодирована в большинстве клиентах – другие узлы начинают забывать этот узел.

Таким образом, сеть обрабатывает узлы, переходящие в офлайн режим.

Напомним, что узлы соединяются со случайными одноранговыми узлами и нет никакой географической топологии.

Теперь предположим, что вы запустили новый узел и хотите присоединиться к сети.

Вы начинаете с простого сообщения для одного узла, о котором вы знаете.

Обычно он называется вашим семенным узлом seed node, и есть несколько разных способов поиска списков семенных узлов, к которым можно подключиться.

Вы отправляете специальное сообщение «Скажите мне адреса всех других узлов в сети, о которых вы знаете».

Вы можете повторить процесс с новыми узлами, о которых вы узнаете, столько раз, сколько хотите.

Затем вы можете выбрать, к каким из них следует подключиться, и вы станете полнофункциональным членом сети Bitcoin.

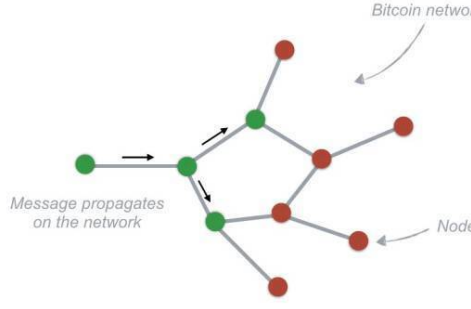
Существует несколько шагов, которые используют случайность, и результатом является то, что вы получаете случайный набор узлов.

Чтобы присоединиться к сети, все, что вам нужно знать, – это связаться с одним узлом, который уже находится в сети.

Для чего нужна сеть?

Конечно, чтобы поддерживать цепочку блоков.

Чтобы опубликовать транзакцию, мы хотим, чтобы об этом узнала вся сеть.



Это происходит с помощью простого алгоритма наводнения, иногда называемого протоколом сплетен.

Если Алиса хочет заплатить Бобу, ее клиент создает, и ее узел отправляет эту транзакцию ко всем узлам, с которыми она связана.

Каждый из этих узлов выполняет серию проверок, чтобы определить, принять и транслировать или нет дальше эту транзакцию.

Если проверки проходят успешно, узел, в свою очередь, отправляет транзакцию всем своим узлам-пирам.

Узлы, которые принимают эту транзакцию, помещают ее в пул транзакций, о которых они слышали, но которые еще не находятся в цепочке блоков.

Если узел получает транзакцию, которая уже находится в

пуле, он больше не транслирует ее.

Это гарантирует, что протокол наводнения прекратится, и транзакции не будут циклически перемещаться по сети вечно.

Помните, что каждая транзакция однозначно определяется своим хешем, поэтому ее легко найти в пуле.

Теперь вопрос, когда узлы узнают о новой транзакции, как они решают, должны ли они ретранслировать ее?

Для этого существуют четыре проверки.

Первая и самая важная проверка – проверка транзакции – транзакция должна быть валидной для текущей цепочке блоков.

Узлы запускают скрипт для каждого предыдущего выхода, который тратится, и проверяют, чтобы скрипт вернул true.

Во-вторых, они проверяют, чтобы выходы, потраченные в этой транзакции, еще не были потрачены.

В-третьих, они не будут передавать уже полученную ранее транзакцию.

В-четвертых, по умолчанию, узлы будут принимать и ретранслировать «стандартные» скрипты на основе небольшого белого списка скриптов.

Все эти проверки – это просто проверки на основе здравомыслия.

Честные узлы выполняют их, чтобы попытаться поддерживать работоспособность сети, но не существует правила, согласно которому узлы должны выполнять эти проверки.

Так как это одноранговая сеть, и может присоединиться любой узел, всегда существует вероятность того, что какой-то узел будет транслировать двойные траты, нестандартные транзакции или совершенно недействительные транзакции.

Вот почему, желательно, чтобы каждый узел выполнял эти проверки.

Так как в сети существует латентность, возможно, что узлы будут иметь разные пулы ожидающих включение в блокчейн транзакций.

Это становится особенно интересным и важным, когда есть попытка двойной траты.

Предположим, что Алиса пытается заплатит один и тот же биткойн как Бобу, так и Чарли, и отправляет две транзакции примерно в одно и то же время.

Некоторые узлы сначала услышат о транзакции Алиса → Боб, а другие сначала услышат о транзакции Alice → Charlie.

Когда узел слышит любую из этих транзакций, он добавляет ее в свой пул транзакций, и, если он услышит о другой транзакции позже, это будет выглядеть как двойная трата.

Узел откажется от последней транзакции и не будет ретранслировать ее или добавлять ее в пул транзакций.

В результате временно между узлами возникнет разногласие о том, какие транзакции следует поместить в следующий блок.

Это называется созданием условия гонки.

Хорошей новостью является то, что все будет в порядке.

Тот, кто замайнит следующий блок, по существу разорвет эту связь и решит, какая из этих двух ожидающих транзакций должна быть окончательно помещена в блок.

Скажем, транзакция Алисы → Чарли попадает в блок.

Когда узлы с транзакцией Алиса → Боб услышат об этом блоке, они выведут эту транзакцию из своих пулов ожидающих транзакций, потому что это двойная трата.

Также, когда узлы с транзакцией Alice → Charlie услышат об этом блоке, они выведут и эту транзакцию из своих пулов, потому что она уже попала в цепочку блоков.

Так что больше не будет никакого разногласия, когда этот блок распространится в сеть.

Так как поведение по умолчанию заключается в том, что узлы должны принимать во внимание то, что они слышат в первую очередь, имеет значение местоположение узлов в сети.

Если две конфликтующие транзакции или блока будут объявлены в двух разных позициях в сети, оба они начнут распространяться по всей сети и какая транзакция узла будет услышана первой, будет зависеть от того, где этот узел находится в сети.

Конечно, это предполагает, что каждый узел реализует эту логику, согласно которой узлы сохраняют все, что они слышат в первую очередь.

Однако не существует центрального органа, обеспечива-

ющего соблюдение этой логики, и узлы могут реализовать любую другую логику, которую они захотят и могут выбрать, какие транзакции ретранслировать.

Мы рассмотрим этот вопрос позже и обсудим, почему майнеры, в частности, могут захотеть реализовать другую логику, отличную от поведения по умолчанию.

До сих пор мы в основном обсуждали распространение транзакций.

Логика объявления новых блоков, когда шахтеры добывают новый блок, почти точно такая же, как распространение новой транзакции, и они также подвержены тем же условиям гонки.

Если одновременно запускаются два валидных блока, только один из них может быть включен в долгосрочную консенсусную цепочку.

В конечном итоге, какой из этих блоков будет включен в долгосрочную консенсусную цепочку, будет зависеть от того, из каких блоков строят цепочки другие узлы, и тот блок, который не попадает в консенсусную цепочку, останется сиротой.

Проверка блока является более сложной, чем проверка транзакций.

В дополнение к проверке заголовка и того, что значение хеша заголовка находится в допустимом диапазоне, узлы должны проверять каждую транзакцию, включенную в блок.

Наконец, узел будет ретранслировать блок только в том

случае, если он основывается на самой длинной ветви цепочки блоков.

Это позволяет избежать образования вилок.

Но, как и при транзакциях, узлы могут реализовать различную логику, если они этого захотят – они могут ретранслировать блоки, которые недопустимы, или блоки, которые строятся из более ранней точки в цепочке блоков.

Это создаст вилку, но все будет в порядке, потому что протокол биткойна разработан таким образом, чтобы противостоять этому.

Теперь вопрос, какова латентность алгоритма наводнения или сплетен?

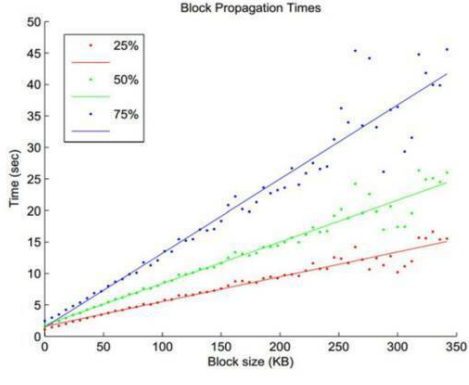


График на слайде показывает среднее время, за которое

новые блоки распространяются в каждый узел сети.

Три линии показывают время покрытия блоком 25, 50 и 75 процентов сети.

Как вы можете видеть, время распространения пропорционально размеру блока.

Это связано с тем, что пропускная способность сети является узким местом.

Большие блоки занимают более 30 секунд для распространения на большинство узлов в сети.

Таким образом, это не очень эффективный протокол.

В Интернете 30 секунд довольно долгое время.

В дизайне Биткойна наличие простой сети с простой структурой, где узлы равны и могут приходить и уходить в любое время, имеют приоритет над эффективностью.

Таким образом, блоку может потребоваться пройти через множество узлов, прежде чем он достигнет самых отдаленных узлов в сети.

Если бы сеть была разработана сверху вниз для повышения эффективности, тогда путь между любыми двумя узлами был бы коротким.

Теперь вопрос, какой размер сети биткойнов.

Трудно измерить, насколько велика сеть, так как она динамична, и не существует центрального узла.

Ряд исследователей дают разные оценки.

Некоторые говорят, что более миллиона IP-адресов действуют одновременно, как узлы биткойна.

С другой стороны, оценивают, что в сети находятся только от 5000 до 10000 узлов, которые постоянно подключены и полностью проверяют каждую транзакцию, которую они слышат.

Полностью проверяющие узлы должны быть постоянно подключены, чтобы слышать обо всех данных.

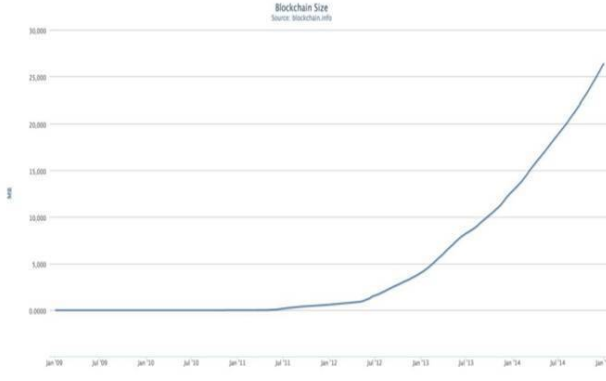
Чем дольше узел находится в автономном режиме, тем ему дольше придется восстанавливать актуальность данных, когда он снова присоединиться к сети.

Такие узлы должны хранить полную цепочку блоков и нуждаются в хорошем сетевом соединении, чтобы иметь возможность слышать каждую новую транзакцию и пересылать ее пирам.

Полностью проверяющим узлам требуется хранить несколько десятков гигабайт данных полного блокчейна.

Наконец, полностью проверяющие узлы должны создавать полный набор неизрасходованных выходов транзакций, которые можно потратить, и хранить этот набор в оперативной памяти, так что, слушая новую предлагаемую транзакцию в сети, узел смог бы быстро найти выходы транзакций, которые пытаются потратить, запустить скрипты, посмотреть, действительно ли действительны подписи, и добавить транзакцию в пул транзакций.

На сегодняшний день, в блочной цепочке насчитывается около 300 миллионов транзакций, из которых около 60 миллионов остались неизрасходованными.



В отличие от полностью проверяющих узлов, существуют легковесные узлы, также называемые тонкими клиентами или клиентами Simple Payment Verification (SPV).

Фактически, подавляющее большинство узлов в сети Биткойн – это легкие узлы.

Они отличаются от полностью проверяющих узлов тем, что они не хранят полную цепочку блоков.

Они хранят только те части, которые им необходимы, чтобы проверить конкретные транзакции, которые им интересны.

Если вы используете программу кошелька, она обычно включает узел Simple Payment Verification (SPV).

Узел загружает заголовки блоков и транзакции, которые

представляют платежи на ваши адреса.

У узла SPV нет уровня безопасности полностью проверяющего узла.

Поскольку узел имеет заголовки блоков, он может проверить, что блоки были добыты, но он не может проверить, что каждая транзакция, включенная в блок, действительно валидная, так как узел не имеет истории транзакций и не знает набор неизрасходованных транзакций.

Узлы SPV могут только проверять транзакции, которые на самом деле влияют на них.

Поэтому они, по сути, доверяют полностью проверяющим узлам, которые подтвердили все остальные транзакции, которые там есть.

Это не плохой компромисс.

Они предполагают, что существуют полностью проверяющие узлы, которые выполняют тяжелую работу, и что, если майнеры добыли этот блок, что является очень дорогостоящим процессом, они, вероятно, также выполнили проверки, чтобы убедиться, что этот блок не будет отброшен.

Экономия затрат на создание узла SPV огромная.

Заголовки блоков составляют примерно 1/1000 размера цепочки блоков.

Поэтому вместо хранения нескольких десятков гигабайт, это всего несколько десятков мегабайт.

Даже смартфон может легко выступать в качестве узла SPV в сети Bitcoin.

Так как биткойн опирается на открытый протокол, в идеале должно быть много разных реализаций, которые легко взаимодействуют друг с другом.

Таким образом, если в одной из них есть ошибка, это вряд ли приведет к разрушению всей сети.

Хорошей новостью является то, что протокол успешно реализуется.

Есть реализации на разных языках, таких как C ++ и Go, и люди создают много других реализаций.

Плохая новость заключается в том, что большинство узлов в сети работают с базовой библиотекой биткойнов, написанной на C ++, поддерживаемой базовыми разработчиками Bitcoin, а некоторые из этих узлов используют предыдущие устаревшие версии, которые не были обновлены.

В любом случае большинство из узлов используют некоторые вариации этого стандартного общего клиента.

Ограничения протокола

Поговорим о некоторых встроенных ограничениях протокола Bitcoin и почему их сложно улучшить.

Есть много ограничений, жестко закодированных в протоколе биткойнов, которые были определены, когда Биткойн был предложен в 2009 году, прежде чем кто-либо действительно подумал, что он может превратиться в глобально важную валюту.

Среднее время на блок 10 минут

Размер блоков 1 мегабайт

Количество подписей в блоке 20000

Биткойны делятся на 8 знаков после запятой

Общее количество биткойнов 21 миллион

Вознаграждение за блок: 50 биткойнов делится пополам каждые 210000 блоков

К ним относятся ограничения на среднее время на блок 10 минут, размер блоков 1 мегабайт, количество подписей в

блоке 20000, и делимость валюты (биткойны делятся только на 8 знаков после запятой), общее количество биткойнов 21 миллион, и структуру вознаграждения блока, которое начиналось с 50 биткойнов и делится пополам каждые 210000 блоков, и другие ограничения.

Ограничения на общее количество существующих биткойнов, а также структура вознаграждения за майнинг, скорее всего, никогда не будут изменены, поскольку экономические последствия их изменения слишком велики.

Майнеры и инвесторы сделали слишком большие ставки в системе, полагая, что структура вознаграждения Биткойна и ограниченная поставка биткойнов останутся такими, как планировалось.

Если это изменится, возникнут большие финансовые последствия для людей.

Поэтому сообщество в основном согласилось с тем, что эти аспекты, независимо от того, были ли они выбраны разумно, не изменятся.

Однако существуют другие изменения, которые, казалось бы, улучшают протокол, потому что некоторые первоначальные варианты дизайна не кажутся совершенно правильными с учетом ретроспективного анализа.

Главными из них являются ограничения, влияющие на пропускную способность системы.

Сколько транзакций может выполнять сетевой процесс Bitcoin в секунду?

Это ограничение исходит из жесткого кодированного ограничения на размер блоков.

Каждый блок ограничен мегабайтом, около миллиона байт.

Каждая транзакция составляет не менее 250 байт.

Разделив 1,000,000 на 250, мы видим, что каждый блок имеет ограничение в 4000 транзакций, и, учитывая, что блоки добываются примерно каждые 10 минут, у нас получается около 7 транзакций в секунду, что все, что может обслуживать сеть Bitcoin.

Может показаться, что изменение этих ограничений было бы вопросом настройки константы в исходном файле кода где-нибудь.

Однако на практике это очень сложно осуществить по причинам, которые мы рассмотрим.

Итак, с чем сравнить семь транзакций в секунду?

Это довольно медленно по сравнению с пропускной способностью любого крупного оператора кредитных карт.

Например, сеть Visa обслуживает в среднем около 2000 транзакций в секунду по всему миру и способна обрабатывать 10 000 транзакций в секунду в пиковые периоды.

Даже Paypal, который меньше, чем Visa, может обрабатывать 100 транзакций в секунду в пиковые периоды.

Это на порядок больше, чем биткойн.

Еще одно ограничение, которое обсуждают, заключается в том, что набор криптографических алгоритмов в биткойне

фиксирован.

Доступно только несколько алгоритмов хэширования и только один алгоритм подписи (ECDSA, по определенной эллиптической кривой, называемой secp256k1).

Существует некоторая озабоченность тем, что в течение жизни Биткойна, которая, как надеются люди, будет очень долгой – этот алгоритм может быть взломан.

Криптографы могут придумать умную новую атаку, которую мы не предвидели, что сделает алгоритм небезопасным.

То же самое относится к хэш-функциям; на самом деле, в последние десятилетия хеш-функции усиленно изучались.

Хэш функция SHA-1, которая используется в биткойне, уже имеет некоторые известные криптографические недостатки, хотя и не фатальные.

Чтобы обойти это ограничение, требуется расширить язык скриптов биткойнов для поддержки новых криптографических алгоритмов.

Как мы можем внедрять новые функции в протокол Bitcoin?

Вам может показаться, что это просто – просто выпустить новую версию программного обеспечения и сообщить всем узлам об обновлении.

В действительности, однако, это довольно сложно.

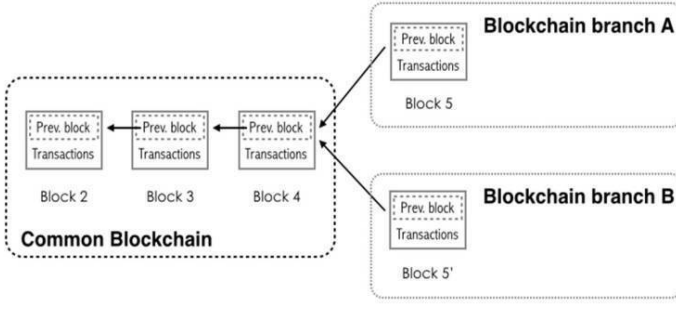
На практике невозможно быть уверенным, что каждый узел обновится.

Некоторые узлы в сети могут не получить новое про-

граммное обеспечение или могут не получить его вовремя.

Последствия обновления большинства узлов, в то время как некоторые узлы будут использовать старую версию, во многом зависят от характера изменений в программном обеспечении.

Мы можем различать два типа изменений: те, которые вызовут жесткий форк, и те, которые вызовут мягкий форк.



Жесткий форк, это когда вводятся новые функции, которые раньше считались бы недействительными.

То есть новая версия программного обеспечения распознает блоки как действительные, однако старое программное обеспечение эти блоки отбросит.

Теперь рассмотрим, что произойдет, когда большинство

узлов обновятся, а некоторые узлы не обновятся.

Очень скоро самая длинная ветвь будет содержать блоки, которые не обновленные узлы считают недействительными.

Поэтому не обновленные узлы будут отбрасывать эту ветвь и будут работать с веткой цепочки блоков, которая не содержит блоки с новой функцией.

Пока они не обновят свое программное обеспечение, они будут считать, что их, более короткая ветка является самой длинной валидной веткой блокчейна.

Этот тип изменений называется жестким форком, поскольку он жестко разделяет цепочку блоков.

Каждый узел в сети будет находиться на той или иной стороне, в зависимости от того, какая версия протокола на нем запущена.

Конечно, при этом ветки никогда не смогут объединиться снова.

Это считается неприемлемым для сообщества, поэтому старые узлы будут эффективно отключены от сети Bitcoin, если они не обновят свое программное обеспечение.

Второй тип изменений или мягкий форк, которые мы можем внести в биткойн, – это добавление функций, которые делают правила проверки более строгими.

То есть они ограничивают набор допустимых транзакций или набор допустимых блоков, так что старая версия будет принимать все блоки, тогда как новая версия будет отбрасывать некоторые блоки.

Этот тип изменений называется мягким форком, и здесь можно избежать постоянного раскола, который создает жесткий форк.

Подумайте, что произойдет, когда мы представим новую версию программного обеспечения с мягким форком.

Узлы, на которых запущено новое программное обеспечение, будут применять новый, более жесткий набор правил.

При условии, что большинство узлов переключится на новое программное обеспечение, эти узлы обеспечат применение новых правил.

Существует риск того, что не обновленные майнеры могут использовать недействительные блоки, поскольку они будут включать некоторые транзакции, которые недействительны в соответствии с новыми, более строгими правилами.

Но не обновленные узлы, по крайней мере, поймут, что некоторые из их блоков отклоняются, даже если они не понимают причину этого.

Это может побудить их обновить программное обеспечение.

Кроме того, если их ветку обгонят обновленные майнеры, не обновленные майнеры переключаются на новую ветку, потому, что блоки, считающиеся действительными для новых майнеров, также считаются действительными старыми майнерами.

Таким образом, здесь не будет жесткой развилки; вместо этого будет много маленьких временных вилок.

И в конце концов все они объединятся в длинную ветку, при условии, что большинство улов обновится, так как все эти маленькие ветки будут обгоняться длинной веткой обновленных узлов.

Классическим примером изменения, которое было сделано с помощью мягкого форка, является введение функции pay-to-script-hash, о котором мы говорили ранее.

В первой версии протокола биткойнов нет pay-to-script-hash.

Это мягкая вилка, потому что с точки зрения старых узлов действительная транзакция с оплатой за скрипт будет проверяться корректно.

С точки зрения старых узлов, скрипт прост – он хэширует одно значение данных и проверяет, соответствует ли этот хэш значению, указанном в выходном скрипте.

Старые узлы не знают, как выполнить дополнительный шаг для запуска этого значения или скрипта, чтобы увидеть, является ли он допустимым скриптом.

Мы полагаемся на новые узлы для обеспечения соблюдения новых правил, то есть скрипт должен потратить эту транзакцию.

Что мы можем добавить с помощью мягкой вилки?

Кроме Pay-to-script-hash.

Также возможно, что новые криптографические схемы могут быть добавлены мягким форком.

Мы могли бы также добавить некоторые дополнительные

метаданные в параметр coinbase, которые бы что-то значили.

Сегодня любое значение принимается в параметре coinbase.

Но мы могли бы в будущем сказать, что coinbase должнен иметь определенный формат.

Одна идея, которая была предложена, состоит в том, что в каждом новом блоке coinbase будет указывать корень Merkle дерева, содержащего весь набор не потраченных транзакций.

Это приведет только к мягкой вилке, потому что старые узлы могут иметь блок, у которого нет требуемого нового параметра coinbase и который был отклонен сетью, но они наверстают упущенное и присоединятся к основной цепочке, которую ведет сеть.

Для других изменений может потребоваться жесткая вилка.

Примером этого является добавление новых опккодов в биткойн, изменение ограничений на размер блоков или транзакций, или различные исправления ошибок.

Чтобы исправить ошибку, о которой мы говорили ранее, где инструкция MULTISIG выталкивает дополнительное значение из стека, на самом деле требуется жесткая вилка.

Это объясняет, почему, несмотря на то, что это раздражающая ошибка, гораздо проще оставить ее в протоколе и заставить людей работать с ней, а не создавать жесткий форк в биткойне.

Жесткие форки, даже если они были бы успешными, вряд ли произойдут в нынешнем Биткойне.

Но многие из этих идей были протестированы и оказались успешными в альтернативных крипторесурсах, которые стартовали с нуля.

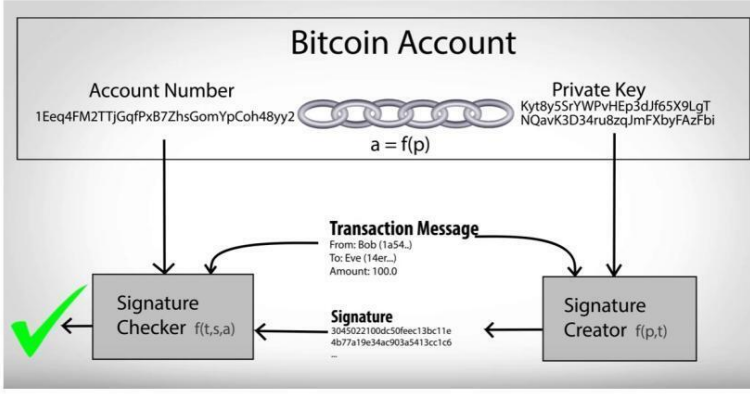
Как хранить и использовать биткойны

Начнем с простейшего способа хранения биткойнов, который просто помещает их на локальное устройство.

Чтобы потратить биткойн, вам нужно знать определенную публичную информацию и определенную секретную информацию.

Публичная информация – это то, что происходит в цепочке блоков – движение монеты, ее стоимость и так далее.

Секретная информация – это секретный ключ владельца биткойна.



Вам не нужно беспокоиться о том, как хранить публичную информацию, потому что вы всегда можете ее получить, когда она вам понадобится.

Но секретный ключ цифровой подписи – это то, что вам следует тщательно охранять.

Поэтому на практике хранение биткойнов – это хранение и управление секретными ключами.

Когда вы определяете, как хранить и управлять ключами, нужно иметь в виду три цели.

Первая – это доступность: возможность тратить свои монеты, когда вы этого захотите.

Вторая – это безопасность: нужно быть уверенным, что никто не сможет потратить ваши монеты, кроме вас самих.

Если кто-то получит доступ к ключу, они могут просто отправить ваши монеты себе.

Третья цель – удобство, то есть управление ключами должно быть относительно легким.

Как вы можете себе представить, достижение всех трех целей одновременно может быть проблемой.

Различные подходы к управлению ключами предлагают различные компромиссы между доступностью, безопасностью и удобством.

Самый простой способ управления ключами – хранить их в файле на своем локальном устройстве: компьютере, телефоне или каком-либо другом гаджете, который вы носите,

или владеете, или управляете.

Этим достигается удобство: наличие приложения для смартфонов позволяет тратить монеты с помощью нескольких кнопок.

Но это нарушает доступность или безопасность – вы можете потерять устройство, устройство может выйти из строя, и вы можете стереть данные, или если ваш файл поврежден, ваши ключи будут потеряны, и, следовательно, вы потеряете ваши монеты.

Аналогично это нарушает обеспечение безопасности: кто-то может украсть или сломать ваше устройство или заразить вредоносным ПО, кто-то может копировать ваши ключи, а затем отправить все ваши монеты себе.

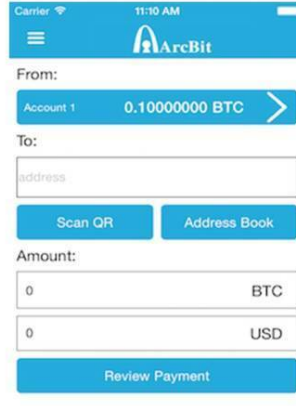
Другими словами, хранение ваших приватных ключей на локальном устройстве, особенно мобильном устройстве, очень похоже на хранение денег в вашем кошельке.

Полезно иметь некоторые карманные деньги, но вы не носите с собой все свои сбережения, потому что можете их потерять, или кто-то может их украсть.

Так что обычно вы храните немного информации / немного денег в кошельке и храните большую часть своих денег в другом месте.

Если вы храните биткойны локально, вы обычно используете программное обеспечение для кошелька, которое является программным обеспечением, отслеживающим все ваши монеты, которое управляет вашими ключами и имеет

удобный пользовательский интерфейс.



Программное обеспечение кошелька позволяет вам легко использовать целую кучу разных адресов с разными ключами.

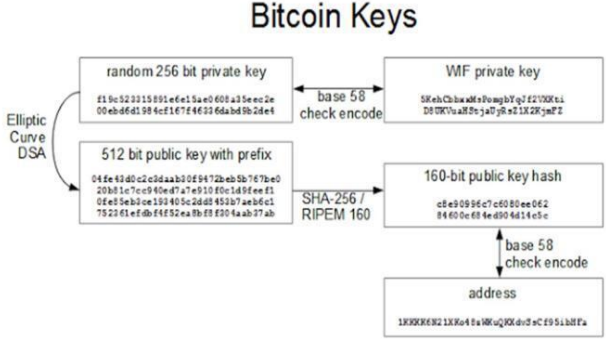
Как вы помните, создание пары публичный-приватный ключей является легким, и вы можете использовать это для своей анонимности или конфиденциальности.

Программное обеспечение кошелька дает вам простой интерфейс, который показывает вам, сколько монет в вашем кошельке.

И когда вы хотите потратить биткойны, кошелек обрабатывает детали того, какие ключи использовать и как создать новый адрес и т. д.

Чтобы тратить или получать биткойны, вам также нужен способ обмена адресом с другой стороной – адресом, которому должны быть отправлены биткойны.

Существует два основных способа кодирования адресов, которые могут быть переданы от получателя к отправителю: это передача текстовой строки или передача QR-кода.



Чтобы закодировать адрес в виде текстовой строки, мы берем биты ключа и преобразуем их из двоичного числа в число base 58.

Что означает base58?

«base» означает число символов, которые вы используете для представления числа.

Base	Characters
2 (binary)	01
10	0123456789
16 (hexadecimal)	0123456789abcdef
58	123456789ABCDEFGH JKLMN PQRSTUVWXYZabcdefghijk mnopqrstuvwxyz

alphanumeric = 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
base58 = 123456789ABCDEFGH JKLMN PQRSTUVWXYZabcdefghijk mnopqrstuvwxyz

В своей повседневной жизни мы пользуемся числами base10.

Чем больше символов у вас в базе, тем меньше их вам нужно использовать для представления больших чисел. Таким образом, чем больше база, тем короче число.

Почему используется base58?

58 – это усеченное количество символов алфавитно-цифрового алфавита, где есть 62 символа.

В base58 удалены плохие символы, такие как 0, O, L и I, которые легко можно спутать.

Таким образом, base58 имеет два преимущества:

Эта база дает большой набор символов, чтобы представлять большие числа в более коротком формате.

И эта база не содержит неудобные символы, чтобы вы не

ошибались при расшифровке.

1000bbcc(base16) = 268483532(base10)

```
      0 Остаток 23    X   584  +
   58)23 Остаток 42   X   583  +
   58)1376 Остаток 2  X   582  +
  58)79810 Остаток 46 X   581  +
 58)4629026 Остаток 24 X  580  +
58)268483532              <----- Начать здесь
```

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

23=Q, 42=j, 2=3, 46=o, 24=R

1000bbcc16=Qj3oR58

Стоимость	символ	Стоимость	символ	Стоимость	символ	Стоимость	символ
0	1	1	2	2	3	3	4
4	5	5	6	6	7	7	8
8	9	9	A	10	B	11	C
12	D	13	E	14	F	15	G
16	H	17	J	18	K	19	L
20	M	21	N	22	P	23	Q
24	R	25	S	26	T	27	U
28	V	29	W	30	X	31	Y
32	Z	33	a	34	b	35	c
36	d	37	e	38	f	39	g
40	h	41	i	42	j	43	k
44	m	45	n	46	o	47	p
48	q	49	r	50	s	51	t
52	u	53	v	54	w	55	x
56	y	57	z				

Например, чтобы перевести десятичное число в 58-ричное, нужно все время делить на 58, брать остаток и сопоставлять его номеру символа базы.

Таким образом, чтобы закодировать адрес в виде текстовой строки, мы берем биты публичного ключа и преобразуем их из двоичного числа в число base 58.

Затем мы используем набор из 58 символов для представления каждой цифры в качестве символа.

Однако такого ручного метода передачи адресов с помощью строк можно избежать с помощью QR-кода.

Таким образом, второй способ кодирования адреса биткойнов – это QR-код, или двумерный штрих-код.

Преимущество QR-кода заключается в том, что вы можете сфотографировать его с помощью смартфона, а программное обеспечение для кошелька может автоматически конвертировать штрих-код в последовательность бит, которая представляет соответствующий биткойн-адрес.

Хранение биткойнов на вашем компьютере или другом устройстве – аналогично тому, как перекладывать деньги из кошелька или в кошелек.

Это называется «горячее хранением».

Это удобно, но также рискованно.

С другой стороны, есть «холодное хранение», которое не подключено к Интернету, и которое заархивировано.

Такое хранение более безопасно, но, конечно, не так удобно.

Чтобы иметь отдельное горячее и холодное хранилище, вам нужно иметь отдельные секретные ключи для каждого из них – иначе монеты в холодном хранилище будут уязвимы, если горячее хранилище будет взломано.

Также вам нужно перемещать монеты туда и обратно между этими двумя хранилищами, используя публичные ключи, которые соответствуют адресам этих хранилищ.

Холодное хранилище не подключено к сети, поэтому горячее хранилище и холодное хранилище не смогут подключаться друг к другу в любом месте.

Но хорошая новость заключается в том, что холодное хранилище не обязательно должно быть онлайн для получения монет – потому что, так как горячее хранилище знает адрес холодного хранилища, оно может в любой момент отправить монеты в холодное хранилище.

В любое время, когда количество денег в вашем горячем кошельке становится слишком большим, вы можете перенести их в холодное хранилище, не подвергая его риску подключением к сети.

В следующий раз, когда холодное хранилище подключится к сети, оно сможет обновить информацию о цепочке блоков, а затем холодное хранилище сможет делать все что хо-

чет с этими монетами.

Но есть небольшая проблема, когда дело доходит до управления адресами холодного хранилища.

С одной стороны, для конфиденциальности и других причин мы хотим иметь возможность получать каждую сумму денег по отдельному адресу со своим секретным ключом.

Поэтому всякий раз, когда мы переносим сумму денег из горячего хранилища в холодное хранилище, мы хотели бы использовать для этой цели свежий холодный адрес.

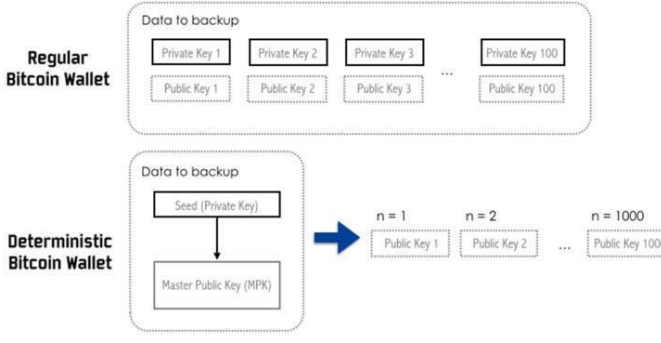
Но так как холодное хранилище не в сети, мы должны иметь какой-то способ, чтобы горячее хранилище узнала об этих адресах.

Тупым решением было бы генерация большой партии адресов одновременно холодным хранилищем и отправка их горячему хранилищу, чтобы в дальнейшем использовать их по одному.

Недостатком является то, что мы должны периодически подключать холодное хранилище, чтобы передавать дополнительно адреса.

Однако более эффективным решением является использование иерархического кошелька.

Это позволяет холодному хранилищу использовать неограниченное количество адресов, и горячему хранилищу знать об этих адресах, с помощью короткой одноразовой связью между двумя хранилищами. Но это требует немного криптографического обмана.



Прежде всего, когда мы говорили о генерации ключей и цифровых подписях, мы рассмотрели функцию, называемую generateKeys, которая генерирует открытый ключ (который действует как адрес) и секретный ключ.

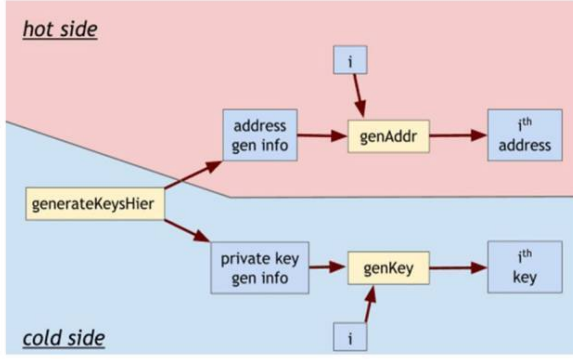
В иерархическом кошельке генерация ключей работает по-другому.

Вместо того, чтобы генерировать единственный адрес, мы генерируем то, что будем называть информацией о генерации адреса, а вместо приватного ключа генерируем то, что мы будем называть информацией о генерации секретного ключа.

Учитывая информацию о генерации адреса, мы можем генерировать последовательность адресов: мы применяем

функцию генерации адреса, которая принимает в качестве входных данных информацию о генерации адресов и любое целое число n и генерирует n-й адрес в последовательности.

Аналогичным образом мы можем генерировать последовательность приватных ключей с использованием информации о генерации секретного ключа.



Криптографическая магия заключается в том, что для каждого i, адрес и секретный ключ соответствуют друг другу, то есть i-й секретный ключ может использоваться для траты биткойнов с i-го адреса так же, как если бы пара была сформирована прежним способом.

Таким образом, теперь у нас есть последовательность пар ключей.

Другим важным криптографическим свойством здесь является безопасность: информация о генерации адресов не передает никакой информации о закрытых ключах.

Это означает, что безопасно предоставить информацию о генерации адресов кому угодно, и этот кто-угодно не сможет сгенерировать i-й ключ.

Теперь, не все существующие схемы цифровой подписи могут быть изменены для поддержки генерации иерархических ключей.

Но хорошей новостью является то, что схема цифровой подписи, используемая Bitcoin, ECDSA, поддерживает иерархическое генерирование ключей, что позволяет использовать этот трюк.

То есть, холодное хранилище может сгенерировать произвольное количество ключей, а горячее хранилище сгенерирует соответствующие адреса.

Холодное хранилище создает и сохраняет информацию о генерации секретного ключа и информацию о генерации адреса.

Оно создает одноразовую передачу информации о генерации адреса горячему хранилищу.

Горячее хранилище генерирует новый адрес последовательно каждый раз, когда оно хочет отправить монеты холодному хранилищу.

Когда холодное хранилище снова подключается в сеть, оно последовательно генерирует адреса и проверяет цепочку

блоков для транзакций на эти адреса, пока не достигнет адреса, который не получил никаких монет.

Оно также может последовательно генерировать приватные ключи, если оно хочет отправить монеты обратно горячему хранилищу или потратить их другим способом.

Теперь давайте поговорим о различных способах хранения холодной информации – будь то один или несколько ключей, или хранение информации о генерации ключей.

Первый способ – хранить эту информацию на каком-либо устройстве и помещать это устройство в сейф.

Это может быть портативный компьютер, мобильный телефон или планшет, или флэш-накопитель.

Главное – выключить устройство и заблокировать его, чтобы, если кто-то захочет его украсть, ему придется взломать заблокированное хранилище.

Второй способ, который мы можем использовать, называется мозговым кошельком.

Это способ контролировать доступ к биткойнам, используя только секретную кодовую фразу.

Это позволяет избежать необходимости использования жестких дисков, бумаги или любого другого долговременного механизма хранения.

Это может быть особенно полезно в ситуациях, когда у вас могут украсть носитель для хранения ключей, например, когда вы путешествуете.

Мозговой кошелек основан на предсказуемом алгоритме

для конвертации ключевой фразы в публичный и приватный ключи.

Например, вы можете хэшировать кодовую фразу подходящей хеш-функцией для получения приватного ключа, а с учетом приватного ключа, публичный ключ можно получить стандартным способом.

Кроме того, объединив это с иерархическим кошельком, мы можем генерировать последовательность адресов и приватных ключей из кодовой фразы, что позволяет создать полноценный кошелек.

Тем не менее, злоумышленник также может получить все приватные ключи в мозговом кошельке, если он сможет угадать кодовую фразу.

Как и всегда в компьютерной безопасности, мы должны предположить, что злоумышленник знает процедуру, которую вы использовали для генерации ключей, и только ваша кодовая фраза обеспечивает безопасность.

Таким образом, злоумышленник может использовать различные фразы и генерировать адреса, проверяя их; если он найдет какие-либо неизрасходованные транзакции в цепочке блоков по любому из этих адресов, он может немедленно потратить их для себя.

Злоумышленник может не знать, кому принадлежали монеты, и эта атака не требует взлома конкретной машины.

Угадывание кодовых фраз мозговых кошельков не направлено на конкретных пользователей, и не оставляет сле-

дов.

Кроме того, в отличие от задачи угадывания пароля электронной почты, которая может быть ограничена вашим почтовым сервером, с мозговыми кошельками, злоумышленник может загрузить список адресов с неиспользованными монетами и пробовать неограниченное число потенциальных фраз.

Обратите внимание, что здесь злоумышленнику не нужно знать, какие адреса соответствуют мозговым кошелькам.

Это называется автономным взломом пароля.

Очень сложно придумать кодовую фразу, которую легко запомнить, но которую сложно угадать.

Одним из надежных способов генерации кодовой фразы является автоматическая процедура выбора случайного 80-битного числа и конвертация этого числа в кодовую фразу так, что разные числа приводят к разным фразам.

На практике также разумно использовать преднамеренно медленную функцию для получения приватного ключа из кодовой фразы (что называется растяжкой ключей), чтобы гарантировать, что атакующему потребуется как можно больше времени для подбора фраз.

Базовый подход состоит в том, чтобы взять быструю криптографическую хеш-функцию, такую как SHA-256, и вычислить, например, 2 в 20 степени ее итераций, умножая вычислительную нагрузку атакующего на коэффициент 2 в 20 степени.

Конечно, если такое вычисление будет слишком медленным, оно начнет раздражать самого пользователя, поскольку его устройство должно быть способно вычислить эту функцию в любое время, когда пользователь захочет потратить монеты из своего мозгового кошелька.

Если кодовая фраза кошелька станет недоступной – скажем, вы ее забыли, не записали и не можете угадать – тогда монеты потеряны навсегда.

Третий вариант холодного хранения – это то, что называется бумажным кошельком.

Мы можем распечатать ключ на бумаге, а затем положить эту бумагу в безопасное место.

Очевидно, что безопасность этого метода так же хороша или плоха, как и физическая безопасность используемой бумаги.

Обычно бумажные кошельки кодируют как публичный, так и приватный ключ двумя способами: как 2D штрих-код и как строку base58.

Так же, как и с мозговым кошельком, хранение небольшого количества информации достаточно, чтобы воссоздать кошелек.

Четвертый способ холодного хранения, с помощью которого мы можем хранить информацию в автономном режиме, состоит в том, чтобы поместить его в какое-либо устройство, защищенное от доступа.

Либо мы вводим ключ в устройство, либо устройство ге-

нерирует ключ; в любом случае, устройство сконструировано таким образом, что оно не может выдавать ключ, то есть для него отсутствует функция чтения.

Вместо этого устройство подписывает транзакции с помощью ключа и делает это, когда мы, скажем, нажимаем кнопку или даем ему какой-то пароль.

Одно из преимуществ заключается в том, что, если устройство потеряно или украдено, мы это узнаем, и ключ может быть украден, только если устройство будет украдено.

Это отличается от хранения вашего ключа на ноутбуке.

Таким образом, люди могут использовать комбинацию из четырех этих методов для хранения своих ключей.

Для горячего хранения, когда хранится большое количество биткойнов, придумываются новые схемы безопасности, чтобы защитить их, и мы поговорим немного об одной из этих более сложных схем далее.

Деление ключей

До сих пор мы рассматривали различные способы хранения и управления приватными ключами, которые управляют биткойнами, но мы всегда хранили ключ в одном месте – в сейфе, в программном обеспечении или на бумаге.

Это дает нам одну точку отказа.

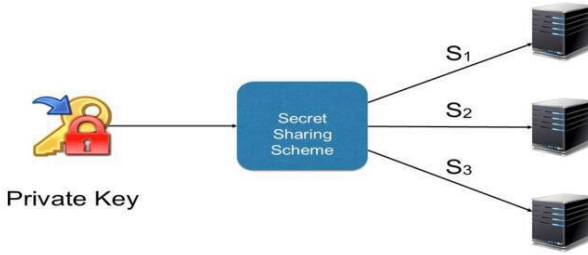
Если что-то пойдет не так с этим местом хранения, тогда у нас возникнут проблемы.

Мы могли бы создавать и хранить резервные копии ключей, что снижает риск потери или повреждения ключа, но увеличивает риск кражи.

Этот компромисс кажется фундаментальным.

Можем ли мы взять часть данных и сохранить их таким образом, чтобы доступность и безопасность увеличивались одновременно?

Очень хорошо, что ответ «да», и это еще один трюк, который использует криптографию, который называется делением секрета.



Идея заключается в следующем: мы хотим разделить наш приватный ключ на некоторое количество частей N.

Мы хотим сделать это таким образом, чтобы, если мы получим какое-либо количество K этих частей, мы сможем восстановить оригинальный секрет, но, если нам дадут меньше, чем K частей, тогда мы не сможем узнать что-либо об оригинальном секрете.

Учитывая это строгое требование, просто «разрезать» секрет на куски не сработает, потому что даже одна часть дает некоторую информацию о секрете.

Нам нужно что-то умнее.

N=2 K=2

S 128-битное число

R 128-битное число

Две части: R и (S XOR R)

Предположим, что N = 2 и K = 2.

Это означает, что мы генерируем 2 части на основе секрета, и нам нужны обе части, чтобы иметь возможность восстановить секрет.

Назовем наш секрет S, который является просто большим (скажем, 128-битным) числом.

Мы могли бы генерировать 128-битное случайное число R и сделать две части равными R и (S побитовое исключающее ИЛИ R).

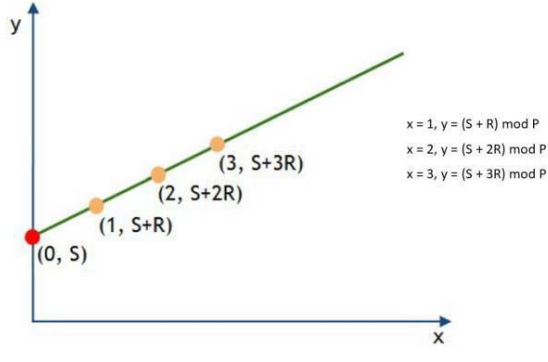
По сути, мы «зашифровали» бы S одноразовым ключом R, и мы сохранили бы ключ (R) и зашифрованный текст (S ИЛИ R) в разных местах.

Ни ключ, ни зашифрованный текст сами по себе ничего не говорят о секрете.

Но, учитывая две части, мы просто собираем их вместе, чтобы восстановить секрет.

Этот трюк работает до тех пор, пока N и K будут одинаковыми – нам просто нужно будет генерировать N-1 разных случайных чисел R для первых N-1 частей, а последней частью будет секрет S – операция ИЛИ – со всеми остальными N- 1 частями.

Но если N больше K, это уже не работает, и нам нужна некоторая алгебра.



Посмотрите на слайд.

Здесь мы должны сначала сгенерировать точку (0, S) по оси Y, а затем нарисовать линию со случайным наклоном через эту точку.

Затем мы создаем точки на этой линии, сколько захотим.

Получается, что это разделение секрета S на N – количество созданных нами точек и K = 2.

Почему это работает?

Во-первых, если мы получим две из созданных точек, мы можем провести через них линию и посмотреть, где она пересечет ось Y.

Это даст нам секрет S.

С другой стороны, если у нас есть только одна точка, она ничего не говорит о секрете S, потому что наклон линии случайный.

Каждая линия в этой точке равно вероятна, и все они пересекают ось Y в разных точках.

Есть только одна тонкость.

Мы берем большое простое число P.

Так, чтобы секрет S был между 0 и P-1, включительно.

Далее мы генерируем случайное значение R, также между 0 и P-1, и создаваемые нами точки

x = 1, y = (S + R) mod P – остаток от деления

x = 2, y = (S + 2R) mod P

x = 3, y = (S + 3R) mod P

и так далее.

Секрет соответствует точке x = 0, y = (S + 0 * R) mod P, которая равна x = 0, y = S.

Таким образом, это способ сделать деление секрета с K = 2 и любым значением N.

Если N = 4, вы можете разделить свой приватный ключ на 4 части и поместить их на 4 разных устройства, чтобы, если кто-то украдет какое-либо из этих устройств, они ничего не узнают о вашем ключе.

С другой стороны, даже если два из этих устройств будут уничтожены, вы сможете восстановить ключ, используя два других устройства.

Как и было обещано, мы увеличили доступность и безопасность.

Но мы можем сделать лучше: мы можем делать деление секрета с любыми N и K, если K не больше N.

Чтобы посмотреть, как это сделать, вернемся к фигуре.

Причина, по которой мы использовали линию вместо некоторой другой кривой, состоит в том, что линия является многочленом степени 1.

Это означает, что для восстановления линии нам нужно не менее двух точек.

Если бы мы хотели сделать K = 3, мы бы использовали параболу, которая представляет собой квадратичный многочлен или многочлен степени 2.

Equation	Degree	Shape	Random parameters	Number of points (K) needed to recover S
$(S + RX) \bmod P$	1	Line	R	2
$(S + R_1X + R_2X^2) \bmod P$	2	Parabola	R_1, R_2	3
$(S + R_1X + R_2X^2 + R_3X^3) \bmod P$	3	Cubic	R_1, R_2, R_3	4

Для построения квадратичной функции необходимы три точки.

Мы можем использовать приведенную таблицу, чтобы понять, что происходит.

Существует формула, называемая интерполяцией Лагранжа, которая позволяет восстановить многочлен степени K-1 из любых K точек на его кривой.

Поэтому, в результате всего этого у нас есть способ хранить любой секрет в виде N частей, чтобы мы были в безопасности, даже если злоумышленник узнает K-1 частей из них.

И в то же время мы можем спокойно утерять N-K частей.

Между прочим, ничего из этого не является специфическим для Биткойна.

Вы можете тайно делить свои пароли прямо сейчас и раздавать части своим друзьям или размещать их на разных устройствах.

Но никто не делает этого с секретами, такими как пароли.

Во-первых, из-за потери удобства.

А во-вторых, потому что существуют другие механизмы безопасности для важных онлайн-учетных записей, например, двухфакторная безопасность с использованием проверки SMS.

Но для Bitcoin, если вы храните свои ключи локально, у вас нет других способов обеспечить безопасность.

Невозможно ограничить доступ к адресу биткойна с помощью SMS-сообщения.

Ситуация отличается от онлайн-кошельков, о которых мы поговорим позже.

Но не принципиально – это просто переносит проблему в другое место.

В конце концов, провайдер онлайн-кошелька должен будет каким-то образом предотвратить одну точку отказа при хранении ключей.

И все равно есть проблема с делением секрета: если мы возьмем ключ, и разделим его.

Когда мы захотим использовать ключ для подписи, нам все равно нужно объединить части и пересчитать первоначальный секрет, чтобы иметь возможность подписать с этим ключом.

Точка, в которой мы объединяем все части, по-прежнему остается одной уязвимой точкой, где злоумышленник может украсть ключ.

Криптография также может решить и эту проблему: если части хранятся на разных устройствах, есть способ генерировать подписи Bitcoin децентрализованным способом, не восстанавливая при этом приватный ключ на каком-то одном устройстве.

Это называется «пороговой подписью».

Здесь наилучшим вариантом использования является кошелек с двухфакторной защитой, который соответствует случаю N = 2 и K = 2.

Скажем, вы настроили свой кошелек на разделение ключей между вашим настольным компьютером и телефоном.

Затем вы можете инициировать оплату на своем настольном компьютере, который создаст частичную подпись и отправит ее на ваш телефон.

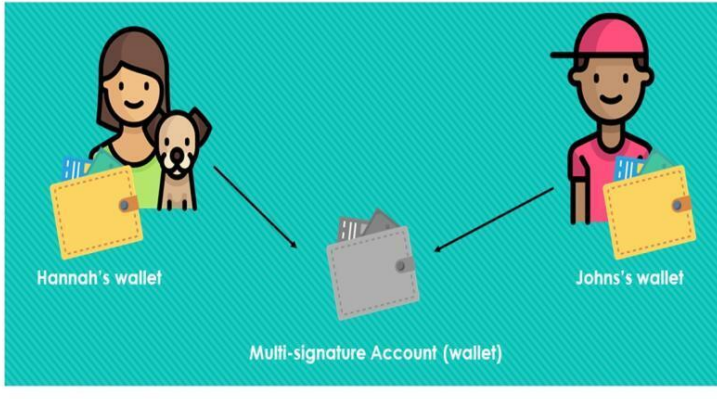
После этого ваш телефон уведомит вас о платежных реквизитах – получателе, сумме и т. д. – и запросит подтверждение.

Если вы подтвердите данные, ваш телефон завершит подпись, используя свою долю закрытого ключа и передаст транзакцию в цепочку блоков.

Если на вашем компьютере был вирус, который пытался украсть ваши биткойны, он может инициировать транзакцию, которая отправляет деньги на адрес хакера, но затем вы

получите уведомление на своем телефоне для транзакции, которую вы не разрешали, и вы не станете ее подтверждать.

Также существует совершенно другой вариант, позволяющий избежать единственной точки отказа: это мульти подпись, про которую мы говорили раньше.



Вместо того, чтобы брать один ключ и разделять его, сценарий Bitcoin напрямую позволяет вам определить, что контроль над адресом должен быть разделен между разные ключи.

Эти ключи затем могут храниться в разных местах, а подписи производятся отдельно.

Конечно, завершенная подписанная транзакция будет создана в конечном итоге на каком-то устройстве, но даже если

злоумышленник будет контролировать это устройство, все, что он сможет сделать, это предотвратить трансляцию транзакции в сеть.

Он не сможет создавать валидные мульти подписи транзакций без участия других устройств.

В качестве примера предположим, что Эндрю, Том, Эд, Джозеф и Стивен, являются соучредителями компании.

И компания имеет много биткойнов.

Мы могли бы использовать мульти подпись для защиты нашего большого хранилища биткойнов.

Каждый из нас сгенерирует пару ключей, и мы будем защищать наше холодное хранилище с помощью 3-из-5 мульти-подписи, что означает, что трое из нас должны подписать, чтобы создать действительную транзакцию.

В результате мы знаем, у нас относительно обеспечена безопасность, при условии, что пятеро из нас хранят ключи отдельно и защищают их по-разному.

Злоумышленнику пришлось бы взломать три из пяти ключей.

Если один или даже двое из нас станут предателями, они не могут украсть монеты компании, потому что для этого нужно как минимум три ключа.

В то же время, если один из нас потеряет ключ, остальные могут перенести монеты на новый адрес и повторно создать ключи.

Другими словами, мульти-подпись помогает управлять

большим количеством холодных монет таким способом, который является относительно безопасным и требует действий нескольких людей, прежде чем что-либо произойдет.

Резюмируем.

Пороговые подписи – это криптографический метод, позволяющий использовать один ключ, разделять его на части, хранить их отдельно и подписывать транзакции без восстановления ключа.

Мульти-подписи – это функция скрипта Bitcoin, с помощью которой вы можете указать, что управление адресом разделяется между несколькими независимыми ключами.

Хотя между ними существуют некоторые различия, они повышают безопасность, избегая одиночных точек отказа.

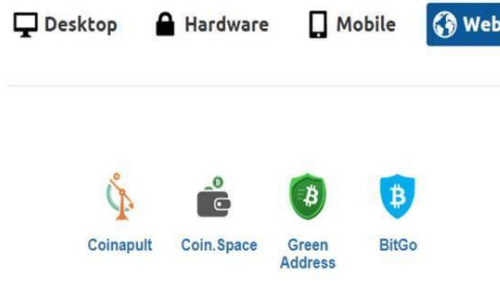
Интернет-кошельки и обменники

До сих пор мы говорили о том, как вы можете хранить и управлять своими биткойнами.

Теперь мы поговорим о том, как вы можете использовать услуги сторонних сервисов которые помогают вам в этом.

Первое, что вы можете сделать, это использовать онлайн-кошелек.

Онлайн-кошелек похож на локальный кошелек, которым вы можете управлять самостоятельно, за исключением того, что информация хранится в облаке, и вы получаете к ней доступ с помощью веб-интерфейса на вашем компьютере или с помощью приложения на вашем смартфоне.



Некоторые популярные онлайн-кошельки показаны на слайде.

Что особенно важно с точки зрения безопасности, так это то, что сайт предоставляет код, который работает в вашем браузере или приложении, а также хранит ваши ключи.

В идеале сайт будет шифровать ваши ключи под паролем, который только вы знаете, но, конечно, вы должны доверять этому сайту, чтобы он это делал.

Вы должны доверять коду сайта, чтобы не потерять ваши ключи или ваш пароль.

У онлайн-кошелька есть определенные преимущества.

Большим преимуществом является то, что это удобно.

Вам не нужно ничего устанавливать на вашем компьютере, чтобы иметь возможность использовать онлайн-кошелек

в своем браузере.

На вашем телефоне вам, возможно, просто нужно будет установить приложение, которому не нужно будет загружать цепочку блоков.

Оно будет работать на нескольких устройствах – у вас может быть один кошелек, доступ к которому вы можете получить на своем настольном компьютере и на своем телефоне, при этом реальный кошелек будет находиться в облаке.

С другой стороны, у онлайн-кошелька есть проблемы с безопасностью.

Если сайт или люди, которые управляют сайтом, оказываются злоумышленниками или каким-то образом скомпрометированы, ваши биткойны находятся в беде.

В идеале, сайт управляется профессионалами по безопасности, которые лучше обучены или, возможно, более старательны, чем вы, при обеспечении безопасности.

Поэтому вы можете надеяться, что они выполняют эту работу лучше и что ваши монеты на самом деле в большей безопасности, чем если бы вы хранили их сами.

Теперь обсудим обменники биткойнов.

Чтобы понять обменники биткойнов, давайте сначала поговорим о том, как банки работают в традиционной экономике.

Вы даете банку немного денег как депозит – и банк обещает вернуть вам эти деньги позже.

Разумеется, на самом деле, банк фактически не берет ва-

ши деньги и не кладет их в ящик в задней комнате.

Весь банк – это обещание, что, если вы потребуете деньги, банк вернет их.

Банк, как правило, берет деньги и помещает их в другое место, то есть инвестирует.

Возможно, банк хранит немного денег в резерве, чтобы гарантировать, что он сможет удовлетворить требование о снятии средств.

Многие банки обычно используют что-то вроде частичного резервирования, когда на всякий случай они хранят определенную долю всех депозитов до востребования.

Теперь, обменники биткойнов – это компании, которые, по крайней мере, с точки зрения пользовательского интерфейса, похожи на банки.

Они принимают депозиты биткойнов и, как банк, обещают вернуть их по требованию позже.

Вы также можете положить на депозит традиционную валюту, такую как доллары и евро в обменник, совершив перевод с вашего банковского счета.

Также обменник позволяет выполнять различные банковские операции.

Вы можете совершать и получать платежи по биткойну, обменять биткойны на валюту или наоборот.

Как правило, они делают это, находя клиента, который хочет купить биткойны за доллары, и другого клиента, который хочет продать биткойны за доллары и соединяют их.

Другими словами, они пытаются найти клиентов, готовых принять противоположные позиции в транзакции.

Если цена взаимоприемлема, они завершают эту транзакцию.

Важно отметить, что, когда эта транзакция происходит с участием меня и другого клиента одного и того же обменника, транзакция на самом деле не фиксируется в цепочке блоков биткойнов.

Обменнику не нужно привлекать цепочку блоков, чтобы перенести биткойны или доллары с одного счета на другой.

Есть плюсы и минусы использования обменников.

Одним из главных плюсов является то, что обменники помогают подключить экономику биткойнов к традиционной валютной экономике, чтобы легко обменивать деньги.

Однако при этом возникают те же риски, как и те, когда вы сталкиваетесь с банками, и эти риски делятся на три категории.

Первым риском является риск ликвидности банка.

Когда много людей одновременно хотят вернуть свои деньги.

Поскольку банк поддерживает только частичное резервирование, он может не справиться с одновременным снятием всех средств.

Это своего рода паническое поведение, когда, как только появляется слух, что банк или биржа могут быть в беде, тогда люди толпой пытаются снять деньги, и вы получаете своего

рода лавину.

Второй риск заключается в том, что владельцы банков могут просто быть мошенниками, создающими пирамиду.

Это схема, когда кто-то убеждает людей давать им деньги в обмен на прибыль в будущем, но затем фактически берет их деньги и использует их для выплаты прибыли людям, которые внесли деньги раньше.

Третий риск – это взлом, риск того, что кто-то, возможно, даже сотрудник обменника, сумеет взломать обменник.

Так как обменники хранят информацию о ключах, которая контролирует большое количество биткойнов, они должны быть очень осторожны в отношении безопасности их программного обеспечения и их процедур – как они управляют своим холодным и горячим хранилищем.

Если что-то пойдет не так, ваши деньги могут быть украдены из обменника.

Все это на самом деле происходило.

Были обменники, которые потерпели неудачу из-за эквивалента нехватки банковской ликвидности.

Были обменники, которые закрылись из-за того, что они являлись на самом деле мошенниками, и были обменники, которых взломали.

Самым известным примером этого, является компания Mt. Gox, которая была когда-то самым крупным обменником биткойнов, и в итоге она оказалась неплатежеспособной, неспособной выплатить деньги, которые она была должна.

Эта компания была японской компанией, и в итоге она объявила о банкротстве и кинула много людей.

Сравнивая это с банками, мы не видим такое количество закрытия банков в большинстве развитых стран, и частично это связано с регулированием.

Правительства регулируют традиционные банки по-разному.

Первое, что делают правительства, – это требуют наличие минимального резерва.

В США доля депозитов до востребования, которые банки должны иметь в ликвидной форме, обычно составляет 3-10 %, так что они могут справится со всплеском изъятий вкладов, если это произойдет.

Во-вторых, правительства часто регулируют типы инвестиций и методы управления капиталом, которые банки могут использовать.

Цель состоит в том, чтобы обеспечить инвестирование активов банков в инструменты с относительно низким риском.

Конец ознакомительного фрагмента.