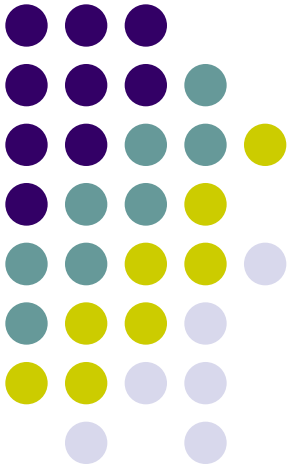
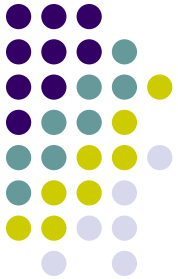
Введение в распределенные методы обработки информации

Лекция №8

Технология Блокчейн

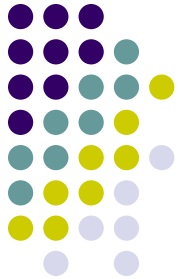


Немного истории



Bitcoin стал результатом развития идей, заложенных в существовавшие ранее криптовалюты.
Впервые он появляется на сцене в ноябре 2008-го, когда пользователь под псевдонимом Сатоши Накамото (Satoshi Nakamoto) публикует статью, описывающую систему новой валюты.
В январе 2009го система начинает функционировать, а её популярность сначала растет медленно, но вскоре переходит в экспоненциальный рост.
Накамото исчезает в апреле 2011, и его личность до сих пор остается загадкой.

Хэш-функция



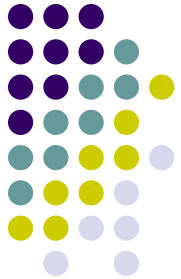
Хеш-функцией (hash function) называется математическая или иная *функция*, которая для строки произвольной длины вычисляет некоторое целое *значение* или некоторую другую строку фиксированной длины.

Хеш-*функция*:

- на вход получает любую строку
- на выходе выдает строку фиксированного размера
- эффективно вычисляема

Смысл хеш-функции состоит в определении характерного признака прообраза – значения хеш-функции.
Это *значение* обычно имеет определенный фиксированный размер, например, 64,128 или ***256 бит***.

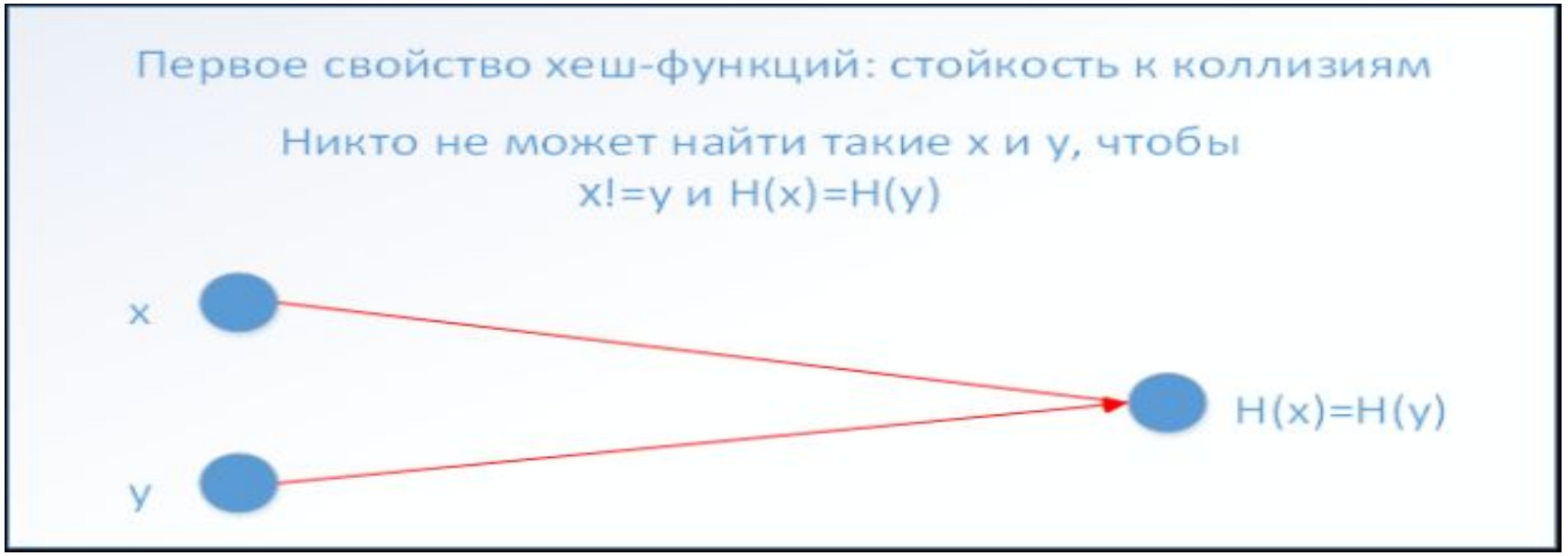
Криптографическая хэш-функция



Основные требования, предъявляемые к криптографическим хеш-функциям:

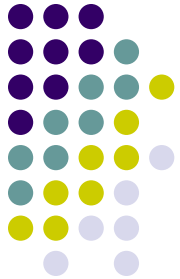
1. Устойчивость к коллизиям
2. Необратимость
3. Открытость к вычислению

Устойчивость к коллизиям



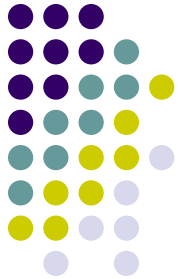
Невозможно подобрать различные значения x и y, для которых значения хэш-функции совпадают.

Устойчивость к коллизиям



Устойчивую к коллизиям хэш-функцию можно использовать в качестве краткого содержания, то есть как своего рода профиля сообщения.
Если значения хэша одинаковые, то предполагаем, что одинаковыми были и изначальные значения.
Потому что, если бы кто-то знал, что x и y, имеющие один *хэш-код*, — разные, то это, конечно же, была бы *коллизия*.
А поскольку это неизвестная *коллизия*, тогда, зная, что значения хэша одинаковы, можно предположить, что значения на входе одинаковы.
Это позволяет использовать хэш в качестве краткого содержания передаваемого сообщения.

Необратимость



Если известен результат хэш-функции, то нет никакого способа определить, какое *значение* X было на входе.

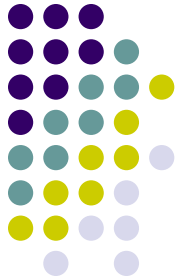
Если хэш-*функция* необратима, то не должно быть случая, когда *значение* X можно легко вычислить, зная H(X)

Обеспечение необратимости в криптографии:

Необходимо взять *значение* X и объединить его со значением r, которое выбрано случайным образом.

H (r|X) означает, что берутся все биты значения r и за ними помещаются все биты значения x. То есть, если берется хэш значения r с хэшем значения x, из этого объединения невозможно вернуть x.

Открытость для сложного вычисления

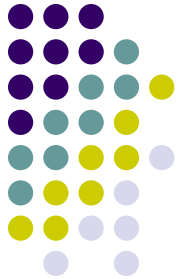


Для любого возможного значения на выходе **y**, если **k** выбрано случайным образом, очень трудно, практически невозможно найти такой **x**, чтобы хэш конкатенации k и x был равен y (**H(k | x) = y**).

Свойство открытости для сложного вычисления подразумевает, что не существует лучшей стратегии решения, чем простой подбор случайных значений **x**. Поэтому, если нужно сформулировать сложную задачу, необходимо генерировать **k** в подходящем случайном порядке.

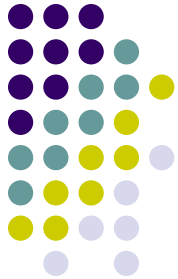
Эта *вычислительная задача* будет рассмотрена в далее в контексте биткоин майнинга.

Хэш-функция, которую использует биткоин - ***SHA-256***



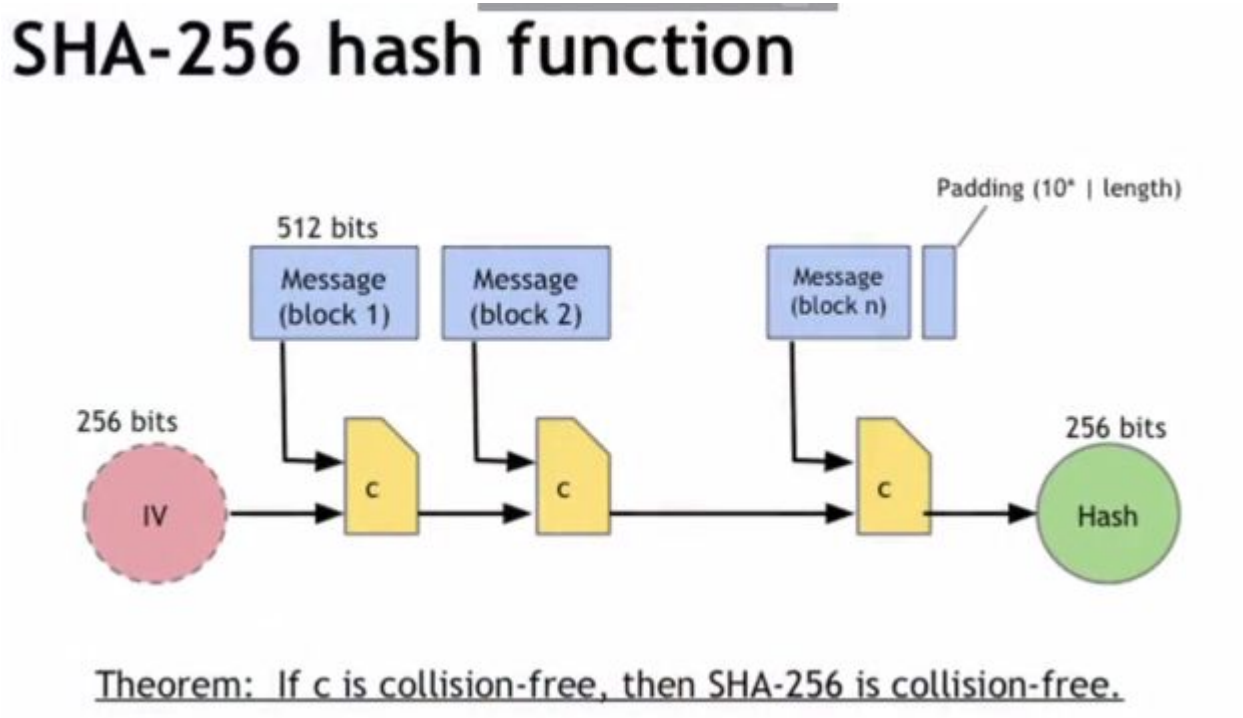
Берется сообщение, которое необходимо хэшировать, и разбивается на блоки размером 512 *бит*
Сообщение не должно быть кратно размеру блока, поэтому в конце добавляется *дополнение* (*padding*).
В конце дополнения будет *поле* длиной 64 *бит*, которое обозначает длину сообщения в битах.
А перед дополнением будет стоять один *бит*, за которым следует некоторое количество нулевых *бит*, чтобы заполнить блок до конца.
Когда *длина* сообщения станет кратной блоку размером 512 *бит*, оно разбивается на эти блоки.

Хэш-функция, которую использует биткоин - ***SHA-256***

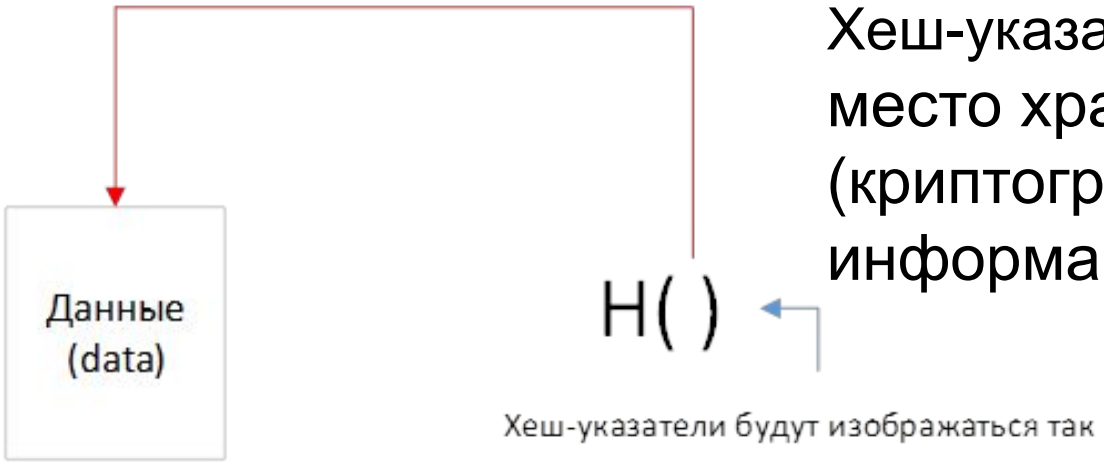


Потом выполняется *вычисление*.
Оно начинается с 256-битного значения, называемого IV. Это номер, который можно просмотреть в стандартном документе. Берется IV и первый блок сообщения, получается 768 *бит*, и пропускается через функцию сжатия. На выходе получается 256 *бит*.
То же самое делается со следующими 512 битами сообщения – этот цикл продолжается до конца.
Каждая *итерация* сжимает последующий 512-битный блок сообщения и включает его в *логический* результат. Результатом вычисления станет хэш, размером 256 *бит*. При этом нет проблемы доказать, что если *функция* обладает свойством стойкости к коллизиям, то вся хэш-*функция* также будет обладать стойкостью к коллизиям.

Хэш-функция, которую использует биткоин



Хеш-указатели и структуры данных



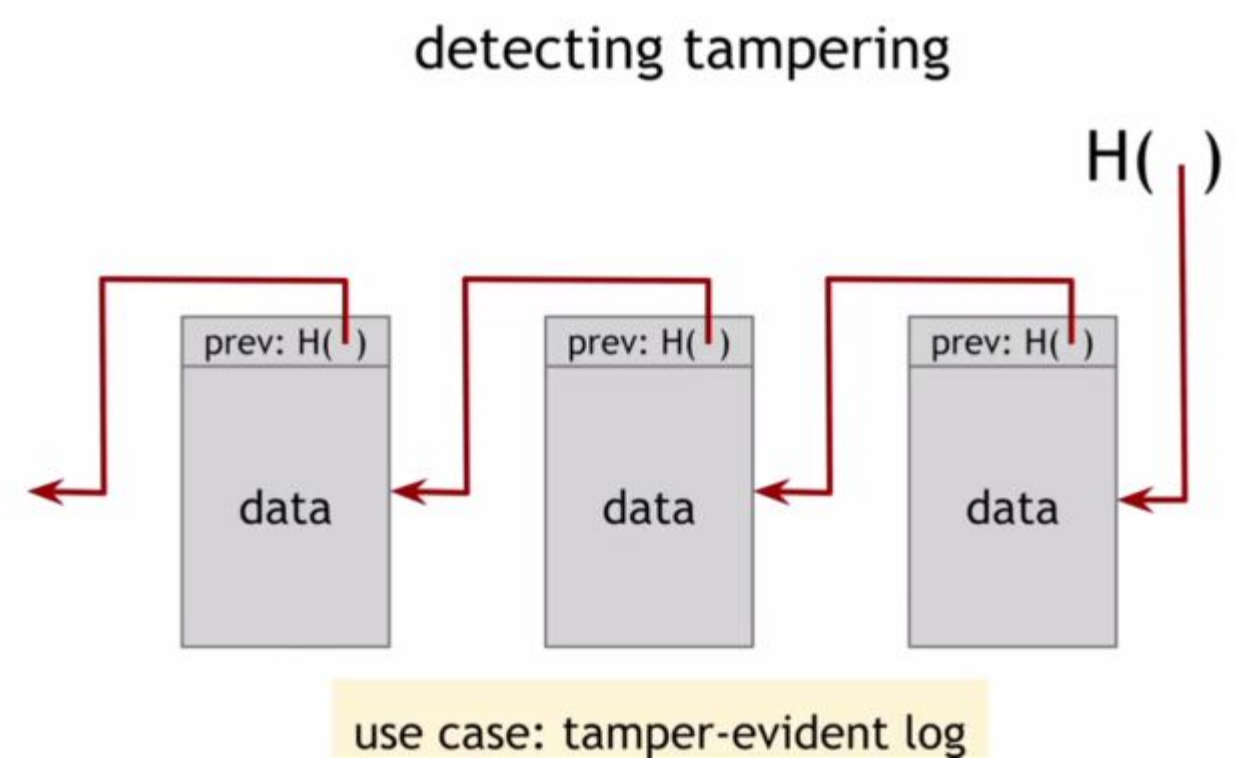
Хеш-указатель – это указатель на место хранения информации и (криптографический) хэш этой информации

Имея хэш-указатель, можно

1. запросить информацию, на которую он указывает обратно
2. верифицировать то, что хэш не изменился, как следствие, не изменилась информация.

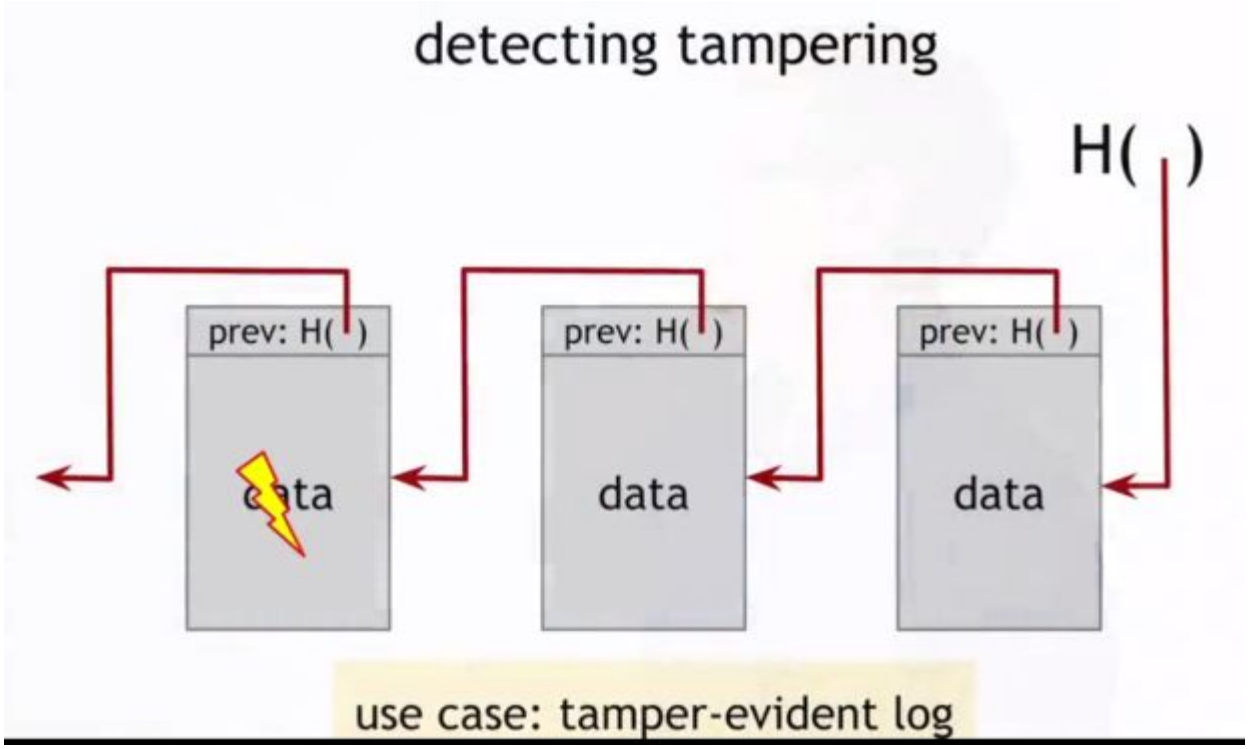
хэш-указатель показывает, где что-то расположено и каково его значение

Цепочка блоков (блокчейн)



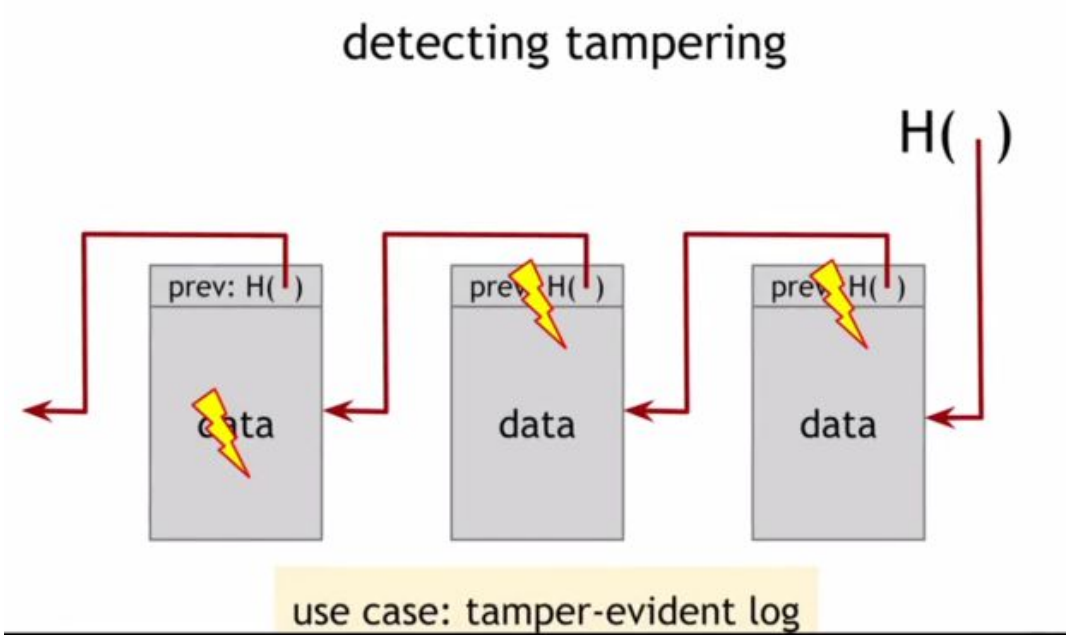
Блокчейн – связанный *список*, состоящий из серии блоков.
Каждый блок содержит данные и хэш-*указатель* на предыдущий блок в списке.
Начало списка сохраняется как обычный хэш-*указатель*.

Пример использования – журнал контроля изменений



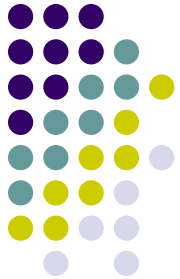
Злоумышленник изменил содержимое блока, находящегося в середине цепочки. Изменится хэш всего этого блока и не будет совпадать с сохраненным ранее. Изменения будут обнаружены

Пример использования – журнал контроля изменений



Если злоумышленник хочет подделать данные в какой-либо точке данной цепочки, то, чтобы сохранить согласованность, ему придется изменять хэш-указатели вплоть до начала.

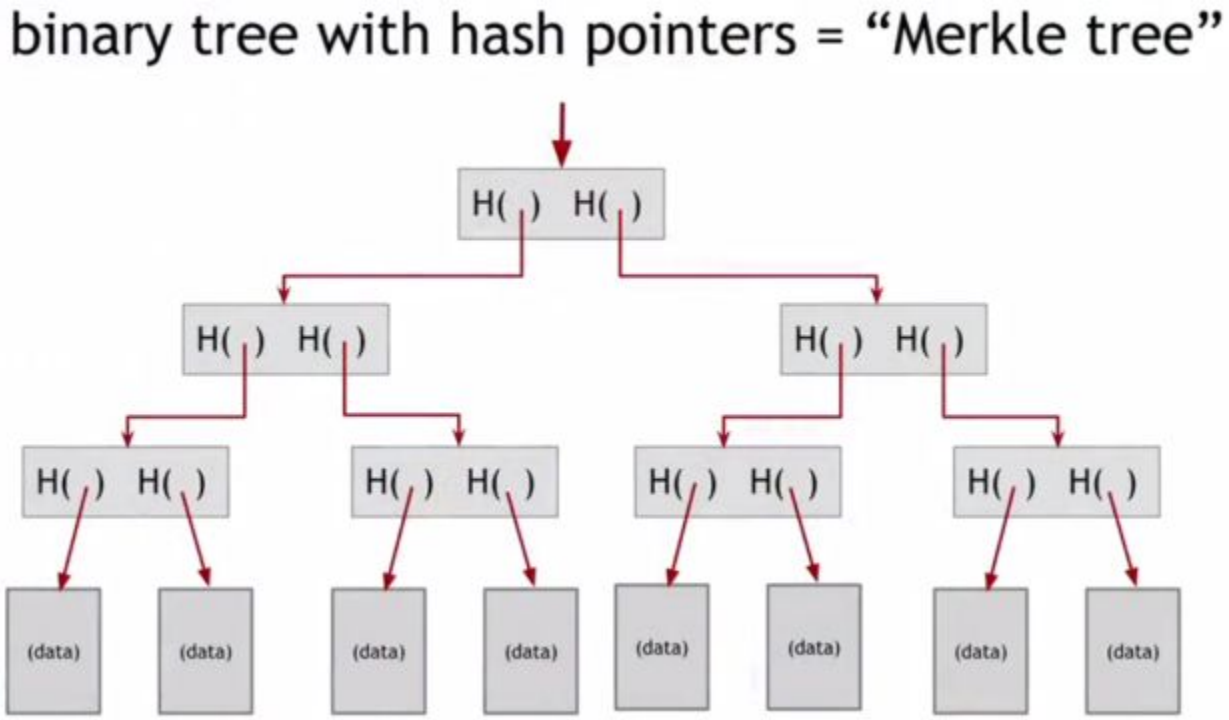
Пример использования – журнал контроля изменений



Журнал контроля изменений – цепочка блоков, в начале которой имеется особый блок, который можно назвать генезис-блоком.

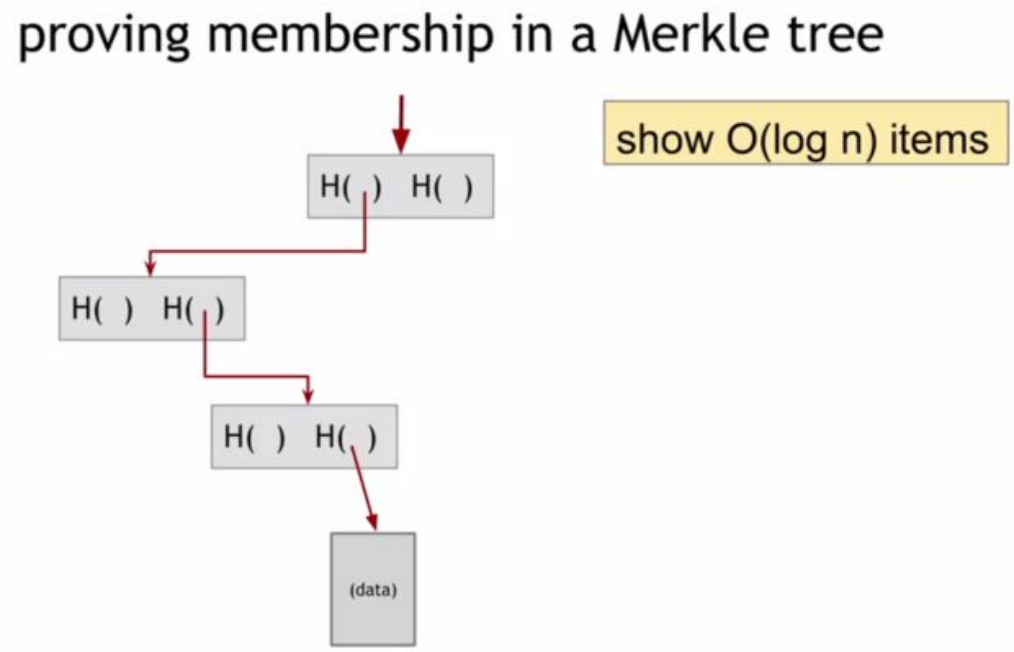
Сохранение хэш-указателя генезис-блока является средством контроля изменений всего списка от начала до конца.

Дерево Меркла



Любая попытка изменить любую часть данных внизу будет замечена благодаря сохраненному хэш-указателю наверху

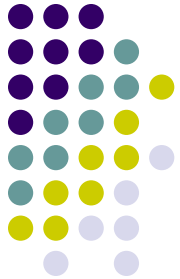
Доказательство вхождения в дерево Меркла



Если кто-то хочет доказать, что конкретный блок данных является членом этого дерева Меркла, нужно будет предоставить только хэш корня.

Даже при очень большом числе блоков данных в дереве Меркла, можно проверить *достоверность* их принадлежности за относительно короткое время.

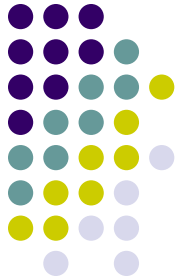
Цифровая подпись



Цифровая (электронная) подпись (digital signature) – уникальное числовое дополнение к передаваемой информации, позволяющее проверить ее авторство.

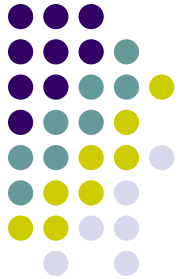
Электронная (цифровая) подпись (ЭЦП) представляет собой последовательность бит фиксированной длины, которая вычисляется определенным образом с помощью содержимого подписываемой информации и секретного ключа.

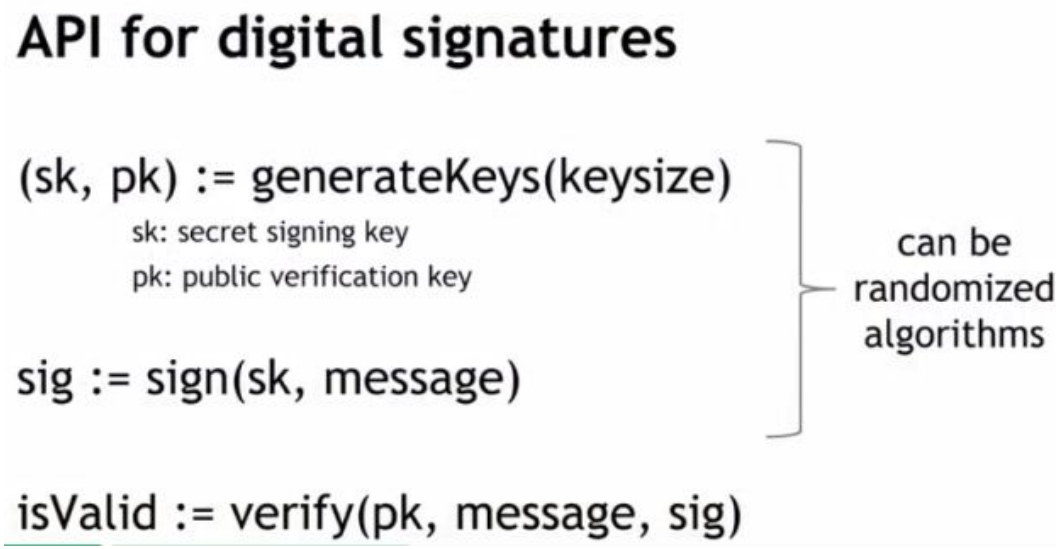
Требования к цифровой подписи



1. Подписать может только один человек, но все могут проверить. Точно так же, как с подписью на бумаге.
2. Подпись связана с определенным документом, ее нельзя вырезать и вставить в другой документ.
3. Подпись не просто подпись, она означает согласие или одобрение конкретного документа.
4. После того, как документ подписан, его невозможно изменить
5. От поставленной подписи невозможно отказаться, то есть лицо, подписавшее документ, не сможет потом утверждать, что не ставило подпись.

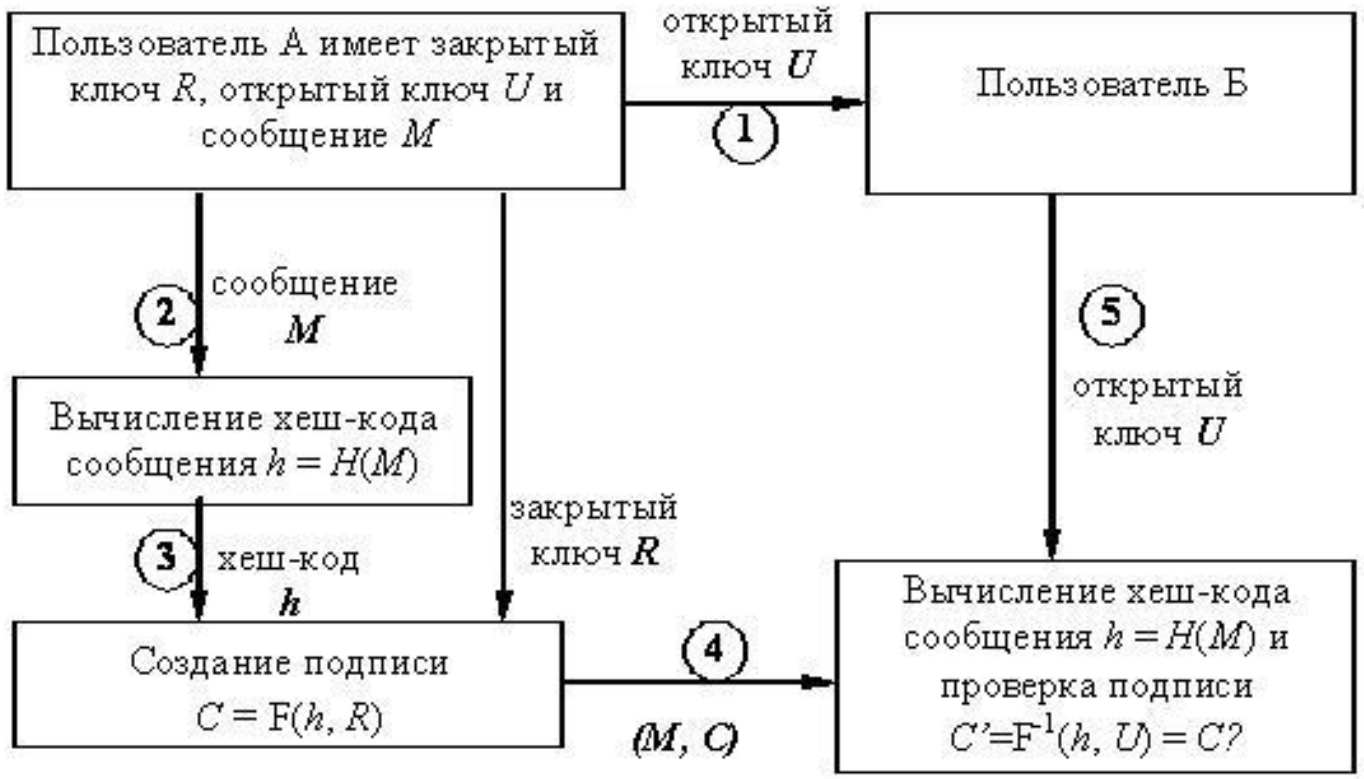
API цифровых подписей



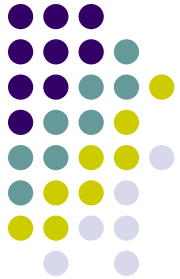


1. Генерация ключей
2. Операция подписи
3. Верификация – проверка подписи

Схема создания и проверки цифровой подписи



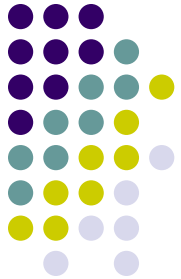
Открытые ключи как идентификаторы личности



Чтобы подписывать сообщения, необходимо знать *закрытый ключ* (sk), соответствующий открытому ключу (pk).
Проставление подписи фактически аналогично некоторому заявлению от имени этого открытого ключа.
Это заявление может сделать только *личность*, которая владеет закрытым ключом.
Соответственно, *открытый ключ* в данном случае является идентификатором личности.

Иными словами, открытый проверочный *ключ* из схемы цифровой подписи можно приравнять к идентификатору человека, действия или системы.

Как создать новый идентификатор

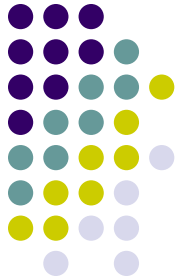


Создаем новую случайную пару ключей (sk, pk):

- pk – публичное "имя", которое можно использовать (обычно лучше использовать хэш от pk)
- sk позволяет "говорить от имени" идентификатора

- создатель ключей контролирует личность, потому что только он знает закрытый ключ
- никто не сможет узнать, кто является владельцем ключа
- можно создать новую личность, которая выглядит случайной, и которую может контролировать только владелец закрытого ключа

Идентификаторы в системе биткоин



Децентрализованное управление идентификацией:

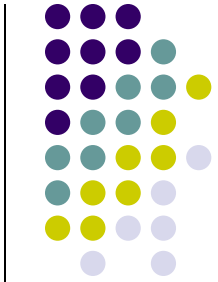
- любой может создать новый идентификатор в любое время в любом количестве
- нет центрального места координации

Идентификаторы в системе биткоин называют "адресами".

Конфиденциальность:

Адреса не имеют прямой привязки к реальной личности.
Но наблюдатель может связать действия адреса в определенное времяи сделать нужные выводы

Простая криптовалюта



Хотим создать криптовалюту - Инфокоин

Владелец валюты - Алиса

Алиса хочет переслать Бобу 1 инфокоин:
Как ?

1. Пишет сообщение: «Я, Алиса, даю Бобу 1 инфокоин»
2. Подписывает сообщение, используя цифровую подпись своим закрытым ключом
3. Публикует сообщение в сети

Валюта – комбинация бит, представляющая в цифровом виде сообщение+ цифровая подпись

Простая криптовалюта – нулевое приближение

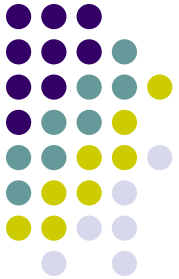
Любой человек в мире, включая Боба может, используя открытый ключ, удостовериться, что:

1. Алиса действительно подписала сообщение
2. Никто другой не мог написать это сообщение, соответственно Алиса не сможет отказаться от того, что это действительно она передала один инфокоин Бобу
3. Никто не может подделать это сообщение
4. Можно скопировать сообщение, после того, как Алиса его опубликовала, но это нельзя сделать с нуля

Основные преимущества протокола Инфокоин:

- подтверждение действий Алисы
- ограниченная защита от подделки

Простая криптовалюта

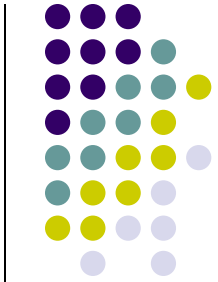


Проблема №1

Алиса может посылать множество таких сообщений Бобу. Пусть
Боб получил 10 подписанных Алисой сообщений: «Я, Алиса, даю Бобу 1 инфокоин»
Что это значит?

1. Боб получил 10 инфокоинов
2. Алиса случайно, по ошибке сгенерировала 10 одинаковых сообщений.
3. Алиса хочет обмануть Боба

Простая криптовалюта



Решение проблемы №1

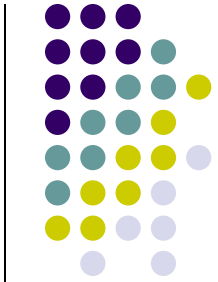
Нужно сделать сообщение уникальным, присвоив ему метку или номер.

Алиса посылает сообщения:

1. «Я, Алиса, даю Бобу 1 инфокоин, серийный номер 8740348»
2. «Я, Алиса, даю Бобу 1 инфокоин, серийный номер 8740349»
3. «Я, Алиса, даю Бобу 1 инфокоин, серийный номер 874050»

Боб и все остальные будут знать, что речь идет о разных инфокоинах.

Простая криптовалюта

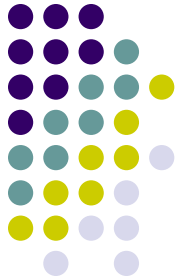


Обеспечение решения проблемы №1

Нужен источник серийных номеров.
Источником может быть

1. Банк:
    - обеспечивает выпуск серийных номеров
    - отслеживает, кто чьи инфокоины получил
    - проверяет легитимность транзакций
2. Создание коллективного банка:
    - каждый владелец инфокоинов хранит копию реестра, содержащего сведения, кто, кому и сколько инфокоинов перечислил
    - Все будут вести полный «гроссбух», содержащий всю информацию о всех транзакциях инфокоинов

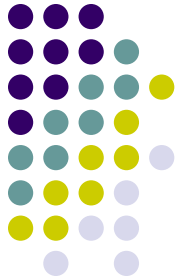
Простая криптовалюта



Проблема №2

Алиса может попытаться обмануть Боба, путем двойного использования инфокоинов (проблема двойной траты).
Алиса пишет два сообщения с одинаковыми серийными номерами двум разным людям .
Если сообщения приходят практически одновременно, то оба человека могут подтвердить получение денег и опубликовать это в сети.

Простая криптовалюта – первое приближение

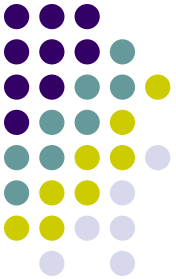


Алиса:

1. Пишет сообщение: «Я, Алиса, даю Бобу 1 инфокоин, серийный номер 123456»
2. Пишет второе сообщения Чарли: «Я, Алиса, даю Чарли 1 инфокоин, серийный номер 123456»
3. Чарли и Боб проверяет сообщение (предположим практически в одно время), и публикуют сообщение вместе с подтверждением в сети

Проблема: как все остальные смогут обновить свой «гроссбух»? Не может быть простого способа достичь согласованной общей книги транзакций. И даже если каждый сможет согласовать последовательный способ обновления своих цепочек блоков, все еще существует проблема, что либо Боб, либо Чарли будут обмануты.

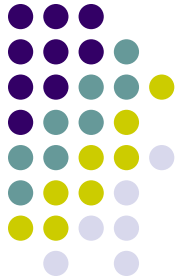
Простая криптовалюта



Решение проблемы №2

Когда Алиса посылает инфокоин Бобу, он не должен проверять это в одиночку.
Боб, получив сообщение должен опубликовать его в сети, и попросить всех проверить легитимность транзакции.
Если все коллективно подтвердят, что никто больше не получал сообщение от Алисы с таким номером, транзакция считается легитимной и все обновят свои гроссбухи
Если люди увидят, что сообщение с одинаковым серийным номером было послано двум разным людям они сообщат об этом и транзакция будет отвергнута.

Простая криптовалюта

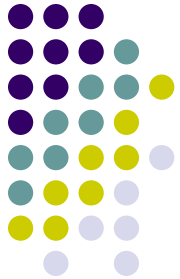


Решение проблемы №2 – более детально

Алиса посылает сообщения:

1. «Я, Алиса, даю Бобу 1 инфокоин, серийный номер 123456»
2. Боб удостоверяет, что 1 инфокоин принадлежит Алисе
3. Боб публикует сообщение Алисы в сети и просит подтверждения
4. Те, кто в сети проверяют, что Алиса действительно владеет 1 инфокоином с серийным номером 123456
5. Как только достаточное количество людей подтвердят, что транзакция легитимна, все обновляют гроссбухи.

Proof-of-work

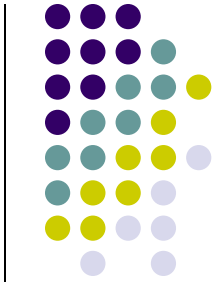


Проблема №3

Алиса может попытаться обмануть всю сеть, путем двойного использования инфокоинов (проблема двойной траты), если она:

1. Сможет создать огромное число фальшивых пользователей в сети, создав ложные идентификаторы личностей (публичные ключи)
2. Ложные личности смогут быстро подтвердить, что транзакции и для Боба и для Чарли легитимны.

Proof-of-work



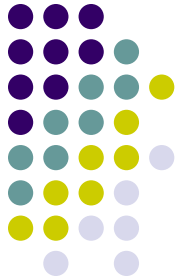
Решение проблемы №3

Работа по подтверждению
В основе лежат две идеи:

1. Сделать работу по подтверждению транзакций вычислительно сложной
2. Назначить награду за осуществление валидаций

Тогда подтверждение транзакций потребует долгой работы больших вычислительных мощностей, что будет дорогим удовольствием для потенциальных обманщиков
Награда заставит пользователей сети включаться в работу по валидации транзакций несмотря на ее вычислительную сложность

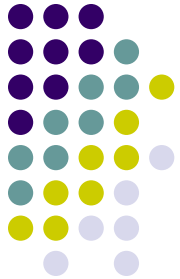
Proof-of-work



Решение проблемы №3 – более детально

Алиса посылает и публикует сообщение :
«Я, Алиса, даю Бобу 1 инфокоин, серийный номер 123456»
Как только остальные участники сети получают это сообщение, они добавляют его к другим подобным еще не подтвержденным сообщениям.
Пусть, Дэвид имеет следующую очередь неподтвержденных сообщений:
«Я, Том, даю Сью 1 инфокоин, серийный номер 1201174»
«Я, Сидни, даю Синтии 1 инфокоин, серийный номер 1295618»
«Я, Алиса, даю Бобу 1 инфокоин, серийный номер 123456»
Дэвид, просмотрев свой блокчейн, решает, что все сообщения валидны и хочет опубликовать это подтверждение.
Но прежде чем сделать это он должен решить некоторую сложную вычислительную задачу – proof-of-work.
Без решения задачи, его подтверждение не будет учитываться.

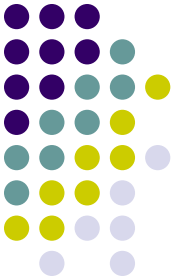
Proof-of-work



Решение проблемы №3 – более детально

Какую работу должен совершить Дэвид?
Пусть ***h*** – фиксированная Хэш функция, известная всем в сети Bitcoin использует ***SHA 256***, но можно использовать любую криптографическую Хэш функцию
Обозначим очередь сообщений Дэвида как ***I***
Пусть Дэвид добавит число x (nonce) к I и хэширует комбинацию.
Задача, которую должен решить Дэвид – найти такое значение x, что после добавления x к I и хэширования, хэш будет начинаться с большого количества нулей.
Трудность задачи можно варьировать, меняя количество нулей в выходном хэше.
Задача затрудняется еще и тем, что значение выходного хэша меняется случайным образом, который трудно предсказать.
Потребуется перебрать достаточно большое количество значений x, чтобы получить необходимое значение хэша.

Proof-of-work

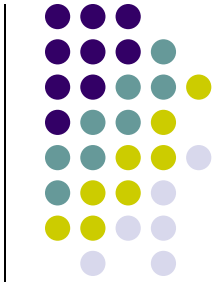


Фактически, протокол биткойнов получает довольно тонкий уровень контроля над трудностью задачи, используя небольшое изменение в значении nonce.

Вместо начальных нулей биткойн протокол требует, чтобы хэш заголовка блока был меньше или равен числу, известному как ***«target».***

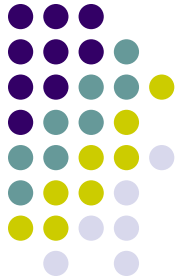
Значение ***«target»*** автоматически корректируется, чтобы гарантировать, что подтверждение биткойн-блока будет занимает в среднем около десяти минут.

Proof-of-work



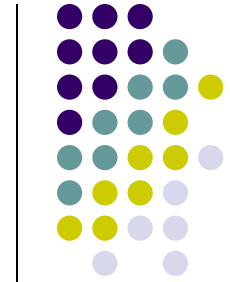
Пусть Дэвид проделал необходимую работу, нашел нужное значение x
Затем он опубликовал всю очередь подтвержденных сообщений и значение x
Другие участники сети могут проверить значение x.
После проверки они обновляют свои «гроссбухи», включив подтвержденную цепочку сообщений
Чтобы идея работы для подтверждения была успешной, участники должны иметь стимул для проведения соответствующей трудной вычислительной работы.
В нашем случае мы можем наградить участников работы по подтверждению некоторым количеством инфокоинов.

Proof-of-work для Биткоин



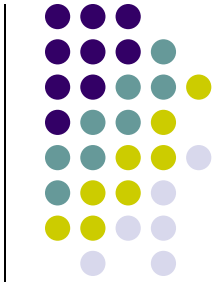
В протоколе биткойнов этот процесс проверки называется «майнинг».
Для каждого проверяемого блока транзакций успешный майнер получает биткойн в качестве награды.
Первоначально награда составляла 50 биткойнов.
Но для каждых 210 000 проверенных блоков (примерно один раз в четыре года) вознаграждение сокращается на половину.
Это сокращение в два раза будет продолжаться каждые четыре года до 2140 года н.э.
В этот момент награда за майнинг опустится ниже 10 ^ {- 8} биткойнов на блок.

Proof-of-work для Биткоин



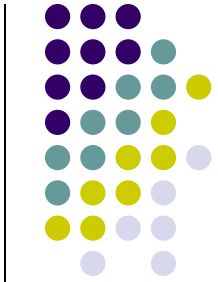
10 ^ {- 8} биткойны - фактически минимальная единица Биткойна и известна как сатоши.
Поэтому в 2140 году н.э. общий объем биткойнов перестанет расти. Однако это не устранит стимул для подтверждения транзакций. Биткойн также позволяет отложить некоторое количество криптовалюты качестве платы за транзакцию в качестве «налога на транзакцию», которая идет на оплату работы майнера
С начала налог на транзакцию был равен нулю, но по мере того, как Биткойн приобретал популярность, транзакционные сборы постепенно повышались и в настоящее время являются существенным дополнительным вознаграждением за добычу блока.

Proof-of-work для Биткоин



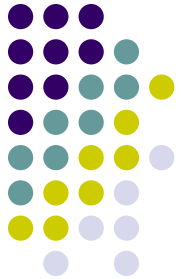
Можно рассматривать proof-of-work как конкурентную борьбу за одобрение транзакций. Каждая одобренная запись требует существенных затрат вычислительной мощности.
Шанс майнера на победу в конкурсе (грубо и с некоторыми оговорками) равен доле общей вычислительной мощности, которую он контролирует.
Если майнер контролирует один процент вычислительной мощности, используемой для проверки транзакций биткойнов, то у него есть примерно один процент шансов на победу.

Proof-of-work для Биткоин



Таким образом, учитывая, что вычислительные затраты велики, у нечестного майнера, вероятно, будет только относительно небольшой шанс для нечестной игры, если он не будет расходовать огромное количество вычислительных ресурсов.
Конечно, хотя обнадеживает тот факт, что у нечестной стороны есть лишь относительно небольшой шанс испортить цепочку блоков, этого недостаточно, чтобы дать нам уверенность в валюте.
В частности, мы еще не окончательно рассмотрели вопрос о двойных тратах.

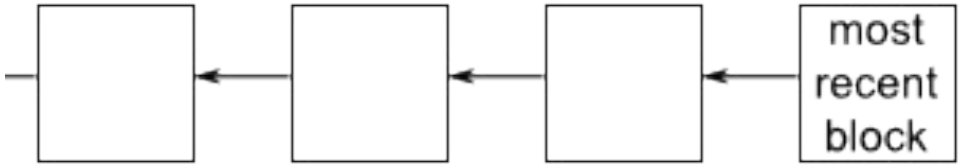
Порядок записей для Inforcoin



Хотелось бы, в идеале, чтобы сеть Infocoin согласовала порядок совершения транзакций. Если этого не сделать, то в любой момент времени может быть неясно, кому принадлежит infocoin. Для установления порядка потребуем, чтобы новые блоки всегда включали указатель на последний блок, проверенный в цепочке, в дополнение к списку транзакций в блоке. (Указатель фактически является только хэшем предыдущего блока).

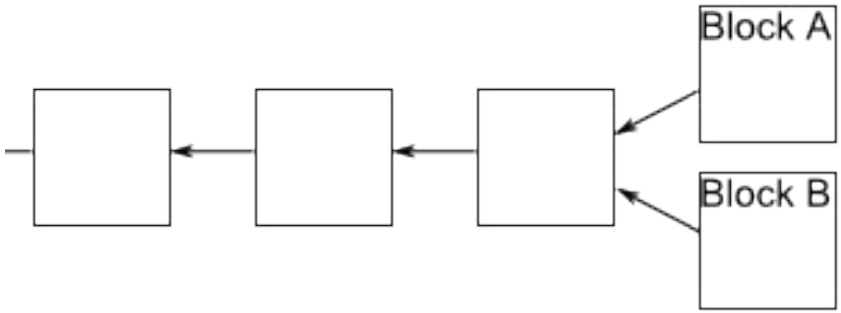
Порядок записей для Inforcoin

Таким образом, обычно цепочка блоков представляет собой просто линейную цепочку транзакций, один за другим, причем последующие блоки содержат каждый указатель на непосредственно предшествующий блок:

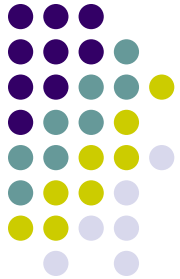


Порядок записей для Inforcoin – возможность «вилки»

Иногда в цепочке блоков появляется вилка.
Это может произойти, например, если два майнера случайно проверили блок транзакций одновременно - оба передали свой недавно проверенный блок в сеть, а участники сети обновили свою цепочку блоков, причем по-разному.



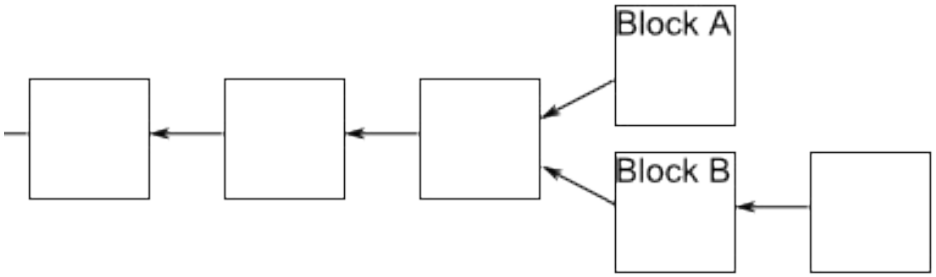
Порядок записей для Inforcoin – возможность «вилки»



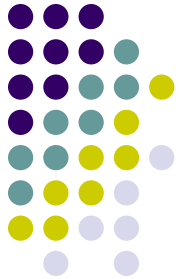
Это приводит именно к проблеме, которую мы пытаемся избежать - уже не ясно, в каком порядке произошли транзакции, и может быть неясно, кто владеет этими инфоконами.
Есть простая идея, которая может быть использована для удаления любых вилок.
Правило таково: если возникает вилка, люди в сети отслеживают обе ветки.
Но в любой момент времени майнеры работают с самой длинной веткой в их копии цепочки блоков.

Порядок записей для Inforcoin – возможность «вилки»

Предположим, например, что у нас есть вилка, в которой одни майнеры сначала получают блок A, а другие - сначала получают блок B.
Те майнеры, которые получают блок A сначала, будут продолжать добычу вдоль этой ветик, в то время как другие вдоль ветки B.
Предположим, что шахтеры, работающие на B, успешно валидировали следующий блок этой ветки:



Порядок записей для Inforcoin – возможность «вилки»

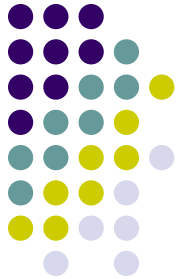


После получения новостей о том, что это произошло, майнеры, работающие на веткне A, заметят, что ветка B теперь длиннее и переключится на работу над этой вилкой. Работа над веткой A прекратится, и все будут работать над одной и той же линейной цепочкой, а блок A можно игнорировать.

Любые оставшиеся транзакции в A все еще ожидающие подтверждения, будут задействована на ветке B, и поэтому все транзакции в конечном итоге будут проверены.

Точно так же может случиться так, что майнеры, работающие на вилке A, первыми добудут новый блок. В этом случае работа над веткой B быстро прекратится, и снова у нас будет одна линейная цепочка.

Задача биткойн-майнеров



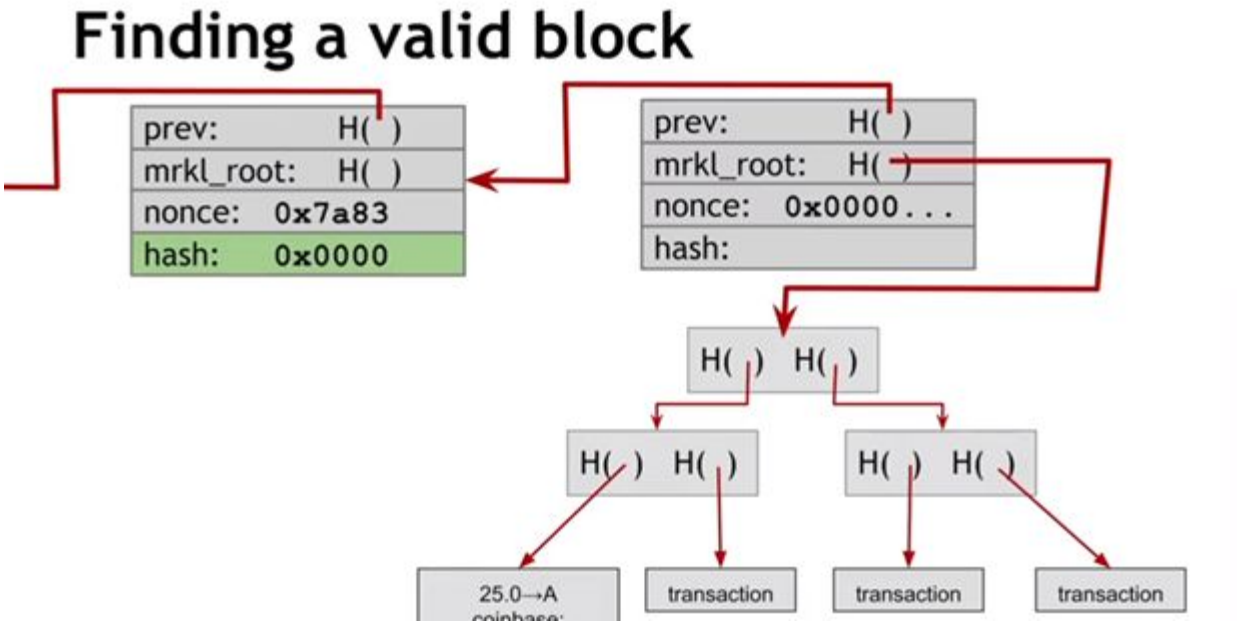
Майнинг существует, чтобы осуществлять подтверждение транзакций.
Остальное: вознаграждение и *поиск* блоков, – это всего лишь стимул, чтобы побудить майнеров осуществлять подтверждение, так как это необходимо для функционирования биткойн в качестве валюты.
Майнинг - это не схема быстрого обогащения, а очень длинный *путь* к заработку.

Задача биткойн-майнеров

Для того, чтобы стать биткойн-майнером нужно:

1. присоединиться к сети и стать Биткойн-нодой
2. начать перехватывать и подтверждать все новые транзакции и блоки, поддерживать текущую цепочку блоков
3. чтобы создать новый блок, путем вычисления случайного одноразовый кода (*nonce*), который позволит блоку считаться подтвержденным.
4. после нахождения блока, все остальные майнеры должны признать этот блок, и начать майнить, продолжая цепочку
5. и если все это совершится, то майнер получит вознаграждение.

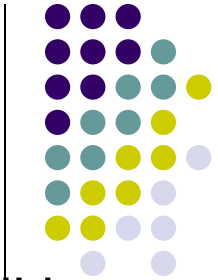
Процесс поиска достоверного блока



Существует цепочка блоков (*block* chain), где каждый заголовок блока указывает на предыдущий заголовок блока в цепочке.

В каждом блоке есть *дерево* Меркла, которое представляет собой бинарное *дерево* на основе хэша всех транзакций, включенных в блок.

Процесс поиска достоверного блока



1. майнер – собирает все транзакции, которые были получены от пула, ожидающего блоки, в *дерево Меркла*.
2. создает блок с правильным заголовком, который указывает на предыдущий блок
3. пытается найти случайный код(*nonce*), чтобы хэш заголовка блока начинался с необходимого количества нулей

Майнер может начать со случайного одноразового кода (*nonce*), состоящего только из нулей (32-битное *целое число*) и, попробовав, получит неверный хэш.

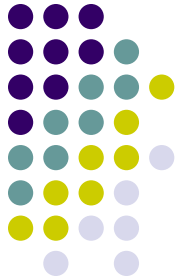
Тогда он перейдет следующему коду (*nonce*) с номером 1, и хэш также будет неверным.

Затем код (*nonce*) номер 2, хэш также окажется неверным.

Майнер будет пытаться использовать каждое возможное *значение* для этого 32-битного целого числа, и, возможно, ни в одном случае хэш не будет верным.

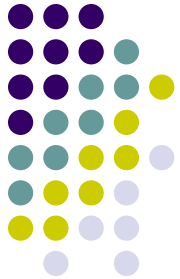
Тогда ему потребуется внести изменения.

Процесс поиска достоверного блока



Комиссионная *транзакция* (**coinbase *transaction***) существует там, где майнеры фактически создают новые монеты и выдвигают свои *права* на них.
В ней существует дополнительный п*араметр* одноразового кода (***nonce***), содержащий только нули.
После того, как майнер исчерпал все возможные одноразовые коды (***nonce***) в заголовке блока, он использует *дополнительный код* (***extra nonce***) в комиссионной транзакции, который изначально содержит только нули.
Он перейдет к дополнительному коду (***extra nonce***) в комиссионной транзакции и изменит его на 1, и затем начнет *поиск* кодов (***nonce***) в заголовке блока заново.

Процесс поиска достоверного блока

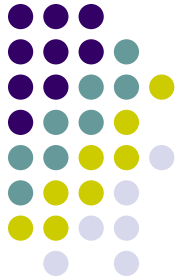


При изменении этого параметра в комиссионной транзакции, все *дерево* транзакций Меркла меняется. Таким образом, изменения будут распространяться до конца.

Изменение дополнительного кода (*extra nonce*) в комиссионной транзакции обойдется гораздо дороже, чем изменение одноразового кода (*nonce*) в заголовке, *по* этой причине майнер делает это как внешний цикл. А внутренний цикл, в котором он действительно трудится, – изменение кода (*nonce*) в заголовке блока.

Но если в очередной раз попытка не сработает, майнеру придется перебирать возможные значения дальше.

Процесс поиска достоверного блока

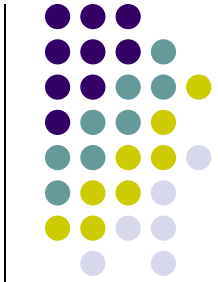


Поиск ***nonce*** – это действительно сложный вычислительный процесс, так что подавляющее большинство кодов (*nonce*), которые проверяет майнер, не срабатывают.

В конечном счете, если майнер посвятит этому достаточно времени, он найдет правильную комбинацию дополнительного кода (*extra nonce*) в комиссионной транзакции и одноразового кода (*nonce*) в заголовке, и сгенерирует блок с хэшем, который начинается с достаточного количества нулей, чтобы считаться достоверным.

Тогда он захочет объявить об этом так быстро, насколько может, и будет надеяться, что сможет извлечь из этого выгоду.

Сложность нахождения *nonce*

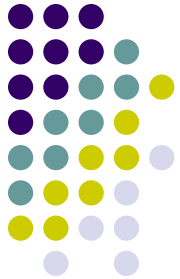


Хэш любого подтвержденного блока должен быть менее 256 *бит*. Мы используем хэш-функцию (*SHA-256*), в которой, *по* крайней мере первые 64 бита хэша любого достоверного блока должны быть нулями.

В целом сложность на настоящий момент составляет 2^{66} = 84 квинтильона = $84x10^{18}$.

Приближение к этому числу, которое можно представить, – это население Земли в квадрате. Другими словами, если бы каждый человек на Земле был бы сам *по* себе своей планетой Земля с 7 миллиардами человек на ней, то общее число людей было бы близко к этому числу.

Сложность майнинга



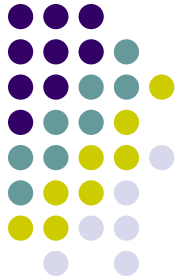
Сложность майнинга пересчитывается каждые 2 недели на основе эффективности работы майнеров в течение предыдущих 2 недель.

Берется время, затрачиваемое майнерами на *поиск* предыдущих 2016 блоков и делится на 2 недели. Затем полученное *отношение* умножается на предыдущее *значение* сложности, чтобы получить следующее *значение* сложности.

Таким образом, *значение* сложности регулярно пересчитывается, чтобы поддерживать постоянной среднюю скорость создания блоков (примерно 1 блок в 10 минут).

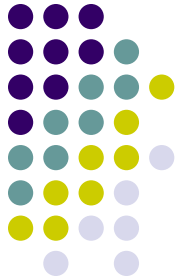
Это константа, которая была установлена в самом начале развития Биткойн, и каждые 2 недели сложность переустанавливается, чтобы гарантировать, что это свойство поддерживается.

Сложность майнинга



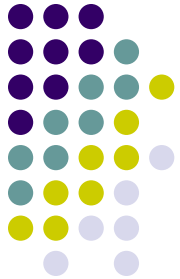
С течением времени сложность майнинга увеличивается. Это не обязательно постоянное линейное или экспоненциальное увеличение.
Сложность зависит от многих параметров: активности на рынке, количества новых майнеров, обменого курса и т.п.
На текущий момент основным фактором является увеличение вычислительных мощностей майнеров в сети. Чем больше людей занимаются хэшированием, тем быстрее создаются новые блоки, и, соответственно, сложность должна будет корректироваться так, чтобы на создание новых блоков снова затрачивалось 10 минут.

Аппаратное обеспечение майнинга



Первое поколение майнинга – майнинг биткойнов позиционировался как возможный для осуществления на компьютерах общего назначения с центральным процессором (ЦП) общего назначения.
ЦП искал коды (*nonce*) *по* линейному закону, вычислял *SHA-256* с помощью программного обеспечения (*ПО*) и проверял, является ли результат достоверным блоком.

Аппаратное обеспечение майнинга



Как быстро это будет происходить на компьютере общего назначения?

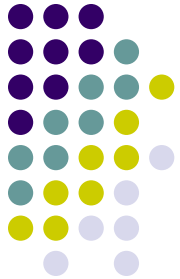
Если настольный ПК высокого класса, то возможно вычислять около 2^{24} хэшей за секунду, что примерно будет составлять 20 мегагерц.

Хотя герц часто применяется *по* отношению к скорости процессора, основная идея герца заключается в обозначении того, что одна операция выполняется много раз в секунду.

Допустим, хэши вычисляются со скоростью 20 мегагерц, тогда оздание одного блока займет более ста тысяч лет с такой скоростью.

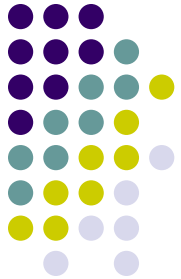
Если сегодня заняться майнингом на ПК общего назначения, *поиск* блока займет более 140000 лет.

Аппаратное обеспечение майнинга



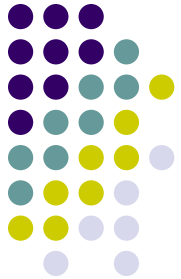
Второе поколение аппаратного обеспечения майнинга связано с попытками использовать графический *процессор* (*GPU*).
Графические процессоры разработаны для обеспечения чрезвычайной параллельности, что помогает при майнинге биткойнов, так как можно одновременно распараллелить и вычислить множество хэшей для разных кодов (*nonce*), чтобы найти подходящий *nonce*.
Помимо этого видеокарты имеют специализированную конвейерную архитектуру для увеличения пропускной способности, что также позволяет увеличить скорость расчетов.
Примерно в 2010 году появилась первая реализация, написанная на языке OpenCL.

Аппаратное обеспечение майнинга



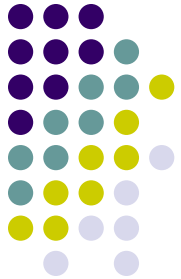
Высокопроизводительные видеокарты с множеством грубых наладок могут предоставить до 200 мегагерц. Это приблизительно составляет 2^{27} хэшей в секунду. Это на порядок выше производительности с использованием ЦП.
Тем не менее, даже если прибавить к этой видеокарте еще 100 видеокарт при текущем уровне сложности уйдет 174 года на *поиск* блока.

Аппаратное обеспечение майнинга



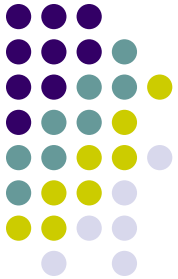
Примерно в 2011 году майнеры начали использовать программируемые пользователем вентильные матрицы (ППВМ).
ППВМ определены как аппаратура, которая может изготавливаться *по* специальным требованиям заказчика или перенастраиваться владельцем, в отличие от заводского микропроцессора, чьи характеристики фиксированы и неизменны.

Аппаратное обеспечение майнинга



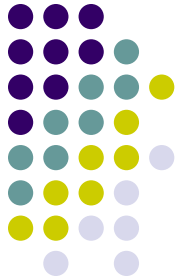
ППВМ предлагают более высокую *производительность*, чем видеокарты. В частности, при выполнении операций, которые не требуют приложения больших усилий.
Если все *операции* точно определены, можно заставить охлаждение работать лучше с ППВМ.
В отличие от видеокарт, существенно меньше функций тратится впустую.
Несколькими ППВМ можно управлять с одного центрального устройства.
Если правильно использовать ППВМ, можно достигнуть производительности около гигагерца.
С такой производительностью и с условием объединения 100 ППВМ, всё равно потребуется 25 лет для создания блока.

Аппаратное обеспечение майнинга



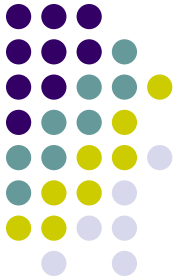
Сегодня скорость вычисления – миллиард хэшей в секунду.
На сегодняшний день в сфере майнинга биткойнов доминирует *интегральная схема* специального назначения (*ASIC*).
Она представляет собой чип, который был разработан и сконструирован с нуля исключительно для майнинга биткойнов.

Аппаратное обеспечение майнинга



В Интернете много предложений *ASIC* с указанием всех необходимых характеристика, таких как *производительность*, *стоимость*, *энергопотребление*, которые помогут сделать выбор.
Однако самым главным критерием отбора является скорость доставки оборудования потребителю.
Большинство *ASIC* имеют довольно значимые оговорки, которые обязывают покупателя предварительно заказать их, даже до того как они будут доступны, и они не дают никаких гарантий *по* поводу даты доставки.

Аппаратное обеспечение майнинга



Сейчас наступила эра профессионального майнинга. Во всем мире появляются профессиональные центры майнинга.

Для организации процессионального майнинг центра необходимы три составляющие:

1. дешевое электричество
2. хорошее сетевое подключение, чтобы получать информацию о новых созданных блоках
3. прохладный климат, чтобы не выкладывать огромные суммы за охлаждение оборудования.

Такие страны как Грузия и Исландия пользуются популярностью для создания майнинг-центров.