Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto

October 31, 2008

[Abstract](https://nakamotoinstitute.org/bitcoin/#abstract)

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

Биткоин: одноранговая система электронных денег

Сатоши Накамото

31 октября 2008 г.

Аннотация

Чисто одноранговая версия электронных денег позволит отправлять онлайн-платежи напрямую от одной стороны к другой, минуя финансовые институты. Цифровые подписи являются частью решения, но основные преимущества теряются, если по-прежнему требуется доверенная третья сторона для предотвращения двойных расходов.

Мы предлагаем решение проблемы двойных расходов с помощью одноранговой сети. Сеть ставит временные метки на транзакции, хешируя их в непрерывную цепочку доказательств-работы на основе хеша, формируя запись, которая не может быть изменена без повторения доказательства-работы (без повторных затрат вычислительной мощности). Самая длинная цепочка служит не только доказательством последовательности наблюдаемых событий, но и доказательством того, что она исходит из самого большого пула вычислительной мощности. Пока большая часть вычислительной мощности контролируется узлами, которые не объединяются для атаки на сеть, они будут генерировать самую длинную цепочку и опережать злоумышленников.

Сама сеть требует минимальной структуры. Сообщения передаются по принципу «максимальных усилий», и узлы могут покидать сеть и повторно присоединяться к ней по своему желанию, принимая самую длинную цепочку доказательств-выполнения-работы в качестве доказательства того, что произошло, пока их не было.

[1. Introduction](https://nakamotoinstitute.org/bitcoin/#introduction)

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to

1. Введение

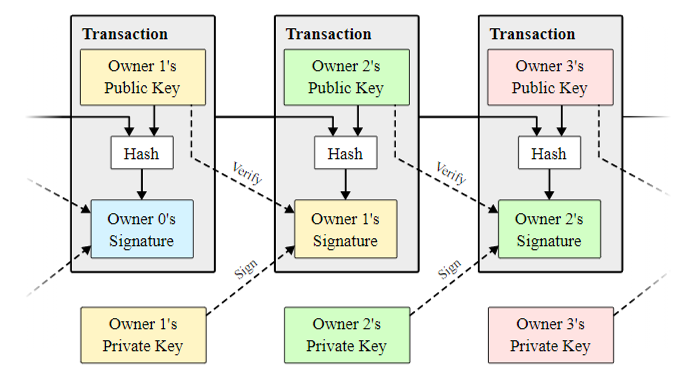
Торговля в Интернете стала полагаться почти исключительно на финансовые институты, выступающие в качестве доверенных третьих сторон для обработки электронных платежей. Хотя система работает достаточно хорошо для большинства транзакций, она по-прежнему страдает недостатками, присущими модели, основанной на доверии. Совершенно необратимые транзакции на самом деле невозможны, поскольку финансовые институты не могут избежать посредничества в спорах. Стоимость посредничества увеличивает транзакционные издержки, ограничивая минимальный практический размер транзакции и исключая возможность проведения маленьких повседневных транзакций, а потеря возможности совершать необратимые платежи за необратимые услуги приводит к более широким издержкам. С возможностью обратимости потребность в доверии увеличивается. Продавцы должны с осторожностью относиться к своим покупателям, требуя от них большего количества информации, чем им было бы в противном случае необходимо. Определенный процент мошенничества считается неизбежным. Этих затрат и неопределенностей с платежами можно избежать, используя физическую валюту, но не существует механизма для осуществления платежей по каналу связи без доверенной стороны.

Что необходимо, так это система электронных платежей, основанная на криптографическом доказательстве вместо доверия, позволяющая любым двум желающим сторонам совершать транзакции напрямую друг с другом без необходимости в доверенной третьей стороне. Транзакции, которые невозможно отменить с вычислительной точки зрения, защитят продавцов от мошенничества, а обычные механизмы условного депонирования могут быть легко реализованы для защиты покупателей. В этой статье мы предлагаем решение проблемы двойной траты с использованием однорангового распределенного сервера временных меток для

generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

[2. Transactions](https://nakamotoinstitute.org/bitcoin/#transactions)

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

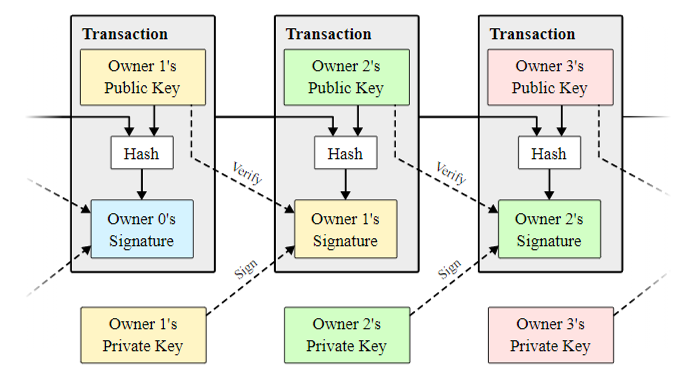


The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the

генерирования вычислительного доказательства хронологического порядка транзакций. Система безопасна до тех пор, пока честные узлы совместно контролируют больше вычислительной мощности, чем любая сотрудничающая группа узлов злоумышленников.

2. Транзакции

Мы определяем электронную монету как цепочку цифровых подписей. Каждый владелец передает монету следующему, подписывая цифровой подписью хэш предыдущей транзакции и открытый ключ следующего владельца и добавляя их в конец монеты. Получатель может проверить подписи, чтобы проверить цепочку владения.



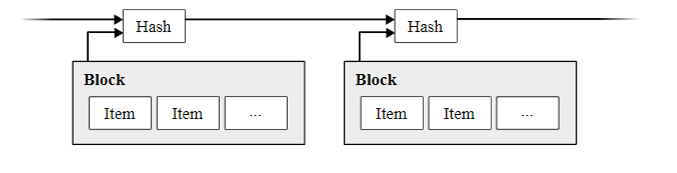
Проблема, конечно, в том, что получатель платежа не может подтвердить, что один из владельцев не тратил монету дважды. Распространенным решением является введение доверенного центрального органа, или эмитента, который проверяет каждую транзакцию на предмет двойных расходов. После каждой транзакции монета должна быть возвращена эмитенту для выпуска новой монеты, и только монеты, выпущенные непосредственно с монетного двора, не могут быть потрачены дважды. Проблема с этим решением заключается в том, что

fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced[[1]](https://nakamotoinstitute.org/bitcoin/" \l "fn1), and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

[3. Timestamp Server](https://nakamotoinstitute.org/bitcoin/#timestamp-server)

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post[[2-5]](https://nakamotoinstitute.org/bitcoin/" \l "fn2). The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

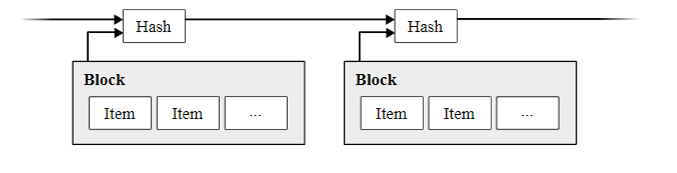


судьба всей денежной системы зависит от компании, выпускающей монеты, и каждая транзакция должна проходить через них, как и банке.

Нам нужен способ, чтобы получатель знал, что предыдущие владельцы не подписывали никаких более ранних транзакций. Для наших целей учитывается самая ранняя транзакция, поэтому нас не волнуют последующие попытки двойного расходования. Единственный способ подтвердить отсутствие транзакции - знать обо всех транзакциях. В модели, основанной на едином эмитенте монет, эмитент знал обо всех транзакциях и решал, какая из них прибыла первой. Чтобы добиться этого без доверенной стороны, транзакции должны быть публично объявлены [1], и нам нужна система для участников, чтобы согласовать единую историю порядка, в котором они были получены. Получателю требуется доказательство того, что во время каждой транзакции большинство узлов согласилось, что она была получена первой.

3. Сервер временных меток

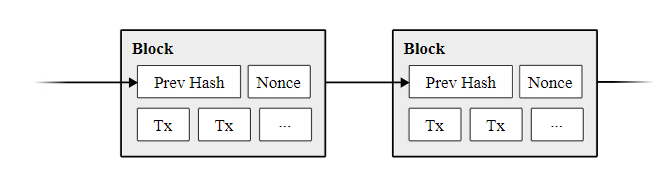
Предлагаемое нами решение начинается с сервера временных меток. Сервер временных меток работает так: берется хеш блока элементов, на которые требуется поставить временные метки, затем этот хеш общедоступно публикуется, например, в газете или сообщении Usenet [2-5]. Отметка времени доказывает, что в указанное время, очевидно, данные должны были существовать, для того, чтобы попасть в хэш. Каждая временная метка включает в свой хэш предыдущую временную метку, образуя цепочку, причем каждая дополнительная временная метка усиливает предыдущие.



[4. Proof-of-Work](https://nakamotoinstitute.org/bitcoin/#proof-of-work)

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash[[6]](https://nakamotoinstitute.org/bitcoin/" \l "fn6), rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.

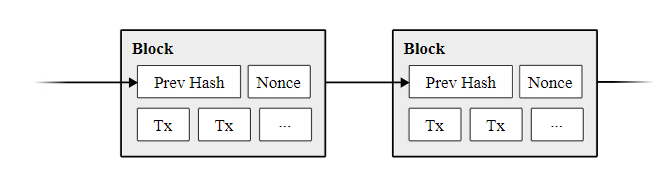


The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote.

4. Доказательство-работы

Чтобы реализовать распределенный сервер временных меток на одноранговой основе, нам нужно будет использовать систему подтверждения работы, скорее аналогичную Hashcash Адама Бэка [6], а не газетам или сообщениям Usenet. Доказательство-работы включает поиск значения числа (решения блока), которое при хешировании, например, с SHA-256, начинается с некоторого количества нулевых битов. Средняя требуемая работа экспоненциальна по количеству требуемых нулевых битов и может быть проверена путем извлечения одиночного хеша.

Для нашей сети с отметками времени мы реализуем доказательство работы, увеличивая решение блока (nonce) до тех пор, пока не будет найдено значение, которое дает хешу блока требуемые нулевые биты. После того, как вычислительная мощность была израсходована, чтобы обеспечить соответствие найденного решения доказательству-выполнения-работы, блок не может быть изменен без повторения работы. Поскольку последующие блоки связаны с предыдущим, работа по изменению блока будет включать повторное выполнение всех блоков после него.



Доказательство-работы также решает проблему определения представительства большинства при принятии решений. Если бы большинство было основано на принципе «один IP-адрес - один голос», это могло бы быть нарушено любым, кто может выделить много IP-адресов. Доказательство работы - это, по сути, один процессор - один голос.

The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

[5. Network](https://nakamotoinstitute.org/bitcoin/#network)

The steps to run the network are as follows:

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a difficult proof-of-work for its block.
4. When a node finds a proof-of-work, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent.

Решение большинства представлено самой длинной цепочкой, в которую вложены самые большие затраты вычислительной мощности для доказательства-работы. Если большая часть вычислительной мощности контролируется честными узлами, честная цепочка будет расти быстрее всех и опережать любые конкурирующие цепочки. Чтобы изменить предыдущий блок, злоумышленник должен будет повторить доказательство-работы блока и всех блоков после него, а затем догнать и превзойти работу честных узлов. Позже мы покажем, что вероятность того, что более медленный злоумышленник догонит, экспоненциально уменьшается по мере добавления последующих блоков.

Чтобы компенсировать увеличение скорости оборудования и изменение интереса к запуску узлов с течением времени, сложность доказательства работы определяется скользящим средним, нацеленным на среднее количество блоков в час. Если они генерируются слишком быстро, сложность возрастает.

5. Сеть

Шаги по запуску сети следующие:

1. Новые транзакции транслируются на все узлы.

2. Каждый узел собирает новые транзакции в блок.

3. Каждый узел работает над поиском сложности доказательства-работы своего блока (имеется ввиду количество нулевых битов в решении).

4. Когда узел находит доказательство-работы (решение блока), он рассылает блок всем узлам.

5. Узлы принимают блок только в том случае, если все транзакции в нем действительны и еще не потрачены.

1. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

[6. Incentive](https://nakamotoinstitute.org/bitcoin/#incentive)

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

6. Узлы выражают свое принятие блока, работая над созданием следующего блока в цепочке, используя хэш принятого блока в качестве предыдущего хеша.

Узлы всегда считают самую длинную цепочку правильной и будут продолжать работать над ее продолжением. Если два узла одновременно транслируют разные версии следующего блока, некоторые узлы могут сначала получить одну или другую. В этом случае они работают с первой полученной веткой, но сохраняют другую ветку на случай, если она станет длиннее. Проблема будет решена, когда будет найдено следующее доказательство работы, и одна ветка станет длиннее; узлы, которые работали на другой веткой, затем переключатся на более длинную.

Сообщения о новых транзакциях не обязательно должны достигать всех узлов. Пока сообщения достигают множества узлов, со временем они попадут в блок. Рассылки блоков также терпимы к выпавшим сообщениям. Если узел не получает блок, он запросит его, когда получит следующий блок, и поймет, что пропустил один.

6. Вознаграждение (стимул)

По соглашению, первая транзакция в блоке - это специальная транзакция, которая выпускает в сеть новую монету, принадлежащую создателю блока. Это добавляет стимул для узлов поддерживать сеть и дает возможность первоначально распределять монеты в обращение, поскольку нет центрального органа для их выпуска. Постоянное добавление определенного количества новых монет аналогично тому, как золотодобытчики тратят ресурсы на добавление золота в обращение. В нашем случае затрачивается время работы процессора и электричество.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

[7. Reclaiming Disk Space](https://nakamotoinstitute.org/bitcoin/#reclaiming-disk-space)

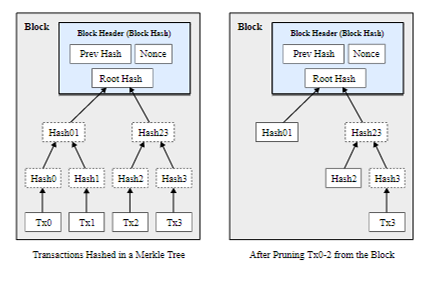
Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree [[7]](https://nakamotoinstitute.org/bitcoin/#fn7)[[2]](https://nakamotoinstitute.org/bitcoin/#fn2)[[5]](https://nakamotoinstitute.org/bitcoin/#fn5), with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.

Поощрение также может финансироваться за счет комиссии за транзакцию. Если выходное значение транзакции меньше входного значения, разница представляет собой комиссию за транзакцию, которая добавляется к стимулирующей стоимости блока, содержащего транзакцию. После того, как заранее определенное количество монет поступило в обращение, поощрение может полностью перейти на комиссию за транзакцию и быть полностью свободным от инфляции.

Вознаграждение может помочь узлам оставаться честными. Если жадный злоумышленник сможет собрать больше вычислительной мощности, чем все честные узлы, ему придется выбирать между использованием ее для обмана людей путем возвращения своих платежей или использованием его для генерации новых монет. Ему должно быть выгоднее играть по таким правилам, которые дают ему новых монет больше, чем всем остальным вместе взятым, а не подрывать систему и ценность своего собственного богатства.

7. Освобождение дискового пространства

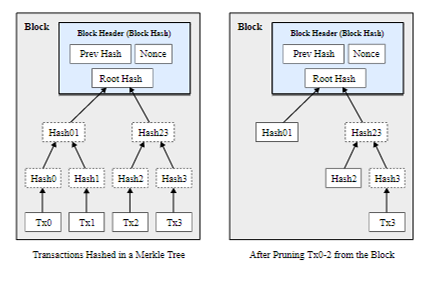
Как только последняя транзакция в монете будет похоронена под достаточным количеством блоков, отправленные транзакции перед ней можно будет отбросить, чтобы сэкономить место на диске. Чтобы облегчить это, не нарушая хеш-функции блока, транзакции хешируются в дереве Меркла [7] [2] [5], причем только корень включается в хеш-код блока. Затем старые блоки можно уплотнить, отрезав ветви от дерева. Внутренние хэши хранить не нужно.



A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, 80 bytes \* 6 \* 24 \* 365 = 4.2MB per year. With computer systems typically selling with 2GB of RAM as of 2008, and Moore's Law predicting current growth of 1.2GB per year, storage should not be a problem even if the block headers must be kept in memory.

[8. Simplified Payment Verification](https://nakamotoinstitute.org/bitcoin/#simplified-payment-verification)

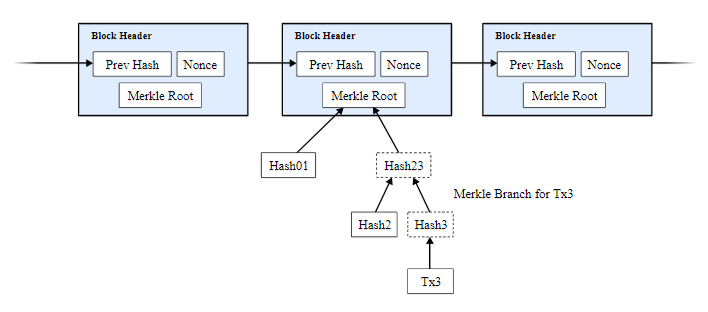
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.



Заголовок блока без транзакций будет около 80 байт. Если предположить, что блоки создаются каждые 10 минут, 80 байт\*6\*24\* 365 = 4,2 МБ/год. Поскольку в 2008 году вычислительные системы (имеется ввиду не компьютер в привычном понимании, а специальная вычислительная система) обычно продаются с 2 ГБ оперативной памяти, а закон Мура предсказывает текущий рост на 1,2 ГБ в год, хранение не должно быть проблемой, даже если заголовки блоков должны храниться в памяти.

8. Упрощенная проверка платежа

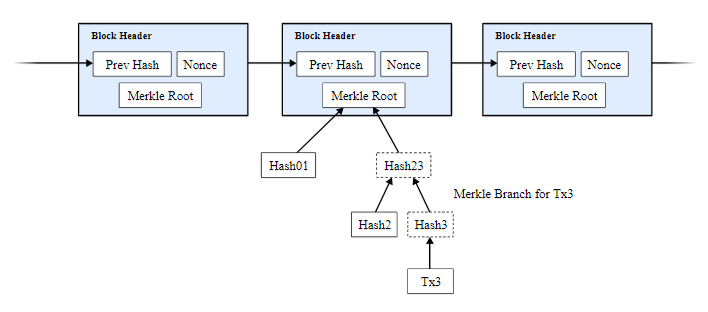
Можно проверять платежи без запуска полного сетевого узла. Пользователю нужно только сохранить копию заголовков блоков самой длинной цепочки подтверждения-выполнения-работы, которую он может получить, запрашивая сетевые узлы, пока не убедится, что у него самая длинная цепочка, и получить ветвь Меркла, связывающую транзакцию с блоком, в котором на нее поставлена временная метка. Он не может проверить транзакцию самостоятельно, но, связав ее с местом в цепочке, он может увидеть, что сетевой узел принял ее, и блоки, добавленные после этого, дополнительно подтверждают, что сеть приняла транзакцию.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

[9. Combining and Splitting Value](https://nakamotoinstitute.org/bitcoin/#combining-and-splitting-value)

Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple

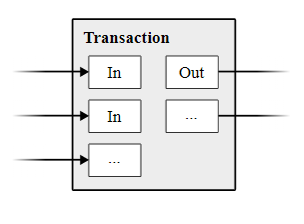


Таким образом, проверка надежна до тех пор, пока честные узлы контролируют сеть, но более уязвима, если сеть взломана злоумышленником. В то время как сетевые узлы могут проверять транзакции для себя, злоумышленник может обходить упрощенный метод сфабрикованными транзакциями до тех пор, пока сможет подавлять сеть. Одна из стратегий защиты от этого - принимать предупреждения от сетевых узлов, когда они обнаруживают недопустимый блок, предлагая программному обеспечению пользователя загрузить полный блок и проверить транзакции для подтверждения несоответствия. Компании, которые часто получают платежи, вероятно, по-прежнему захотят запускать свои собственные узлы для более независимой системы безопасности и более быстрой проверки.

9. Комбинирование и разделение ценности

Хотя можно было бы обрабатывать монеты раздельно, было бы неудобно проводить отдельную транзакцию для каждого цента в переводе. Чтобы осуществить разделение и комбинирование значений, транзакции содержат несколько входов и выходов. Обычно будет либо один ввод из более крупной предыдущей транзакции, либо несколько

inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.

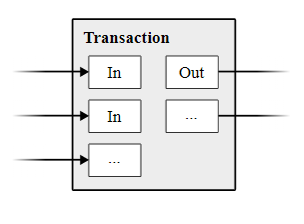


It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

[10. Privacy](https://nakamotoinstitute.org/bitcoin/#privacy)

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.

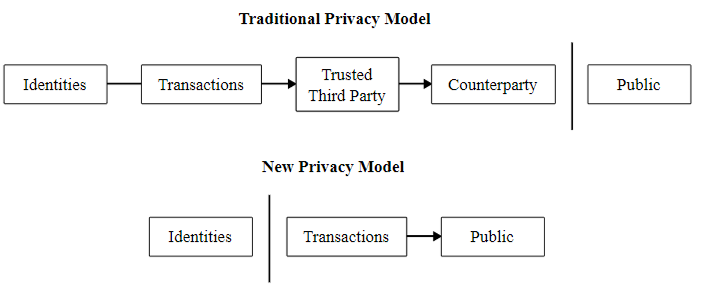
входов, объединяющих меньшие суммы, и не более двух выходов: один для платежа и один для возврата сдачи, если таковая имеется, обратно отправителю.



Следует отметить, что разветвление, когда транзакция зависит от нескольких транзакций, а эти транзакции зависят от многих других, не являются проблемой. Никогда не возникнет необходимость извлекать полную автономную копию истории транзакций.

10. Конфиденциальность

Традиционная банковская модель обеспечивает уровень конфиденциальности за счет ограничения доступа к информации для вовлеченных сторон и доверенной третьей стороны. Необходимость публично объявлять все транзакции исключает этот метод, но конфиденциальность по-прежнему можно поддерживать, прерывая поток информации в другом месте: сохраняя анонимность открытых ключей. Общественность может видеть, что кто-то отправляет сумму кому-то другому, но без информации, связывающей транзакцию с кем-либо. Это похоже на уровень информации, публикуемой биржами, где время и размер отдельных сделок, «биржевая лента», обнародуются, но без указания сторон.

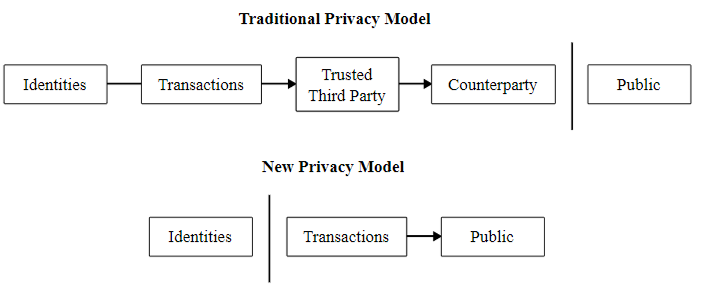


As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

[11. Calculations](https://nakamotoinstitute.org/bitcoin/#calculations)

We consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. Even if this is accomplished, it does not throw the system open to arbitrary changes, such as creating value out of thin air or taking money that never belonged to the attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. An attacker can only try to change one of his own transactions to take back money he recently spent.

The race between the honest chain and an attacker chain can be characterized as a Binomial Random Walk. The success event is the honest chain being extended



В качестве дополнительного брандмауэра, для каждой транзакции следует использовать новую пару ключей, чтобы они не были связаны с общим владельцем. Наличие некоторой связи все еще неизбежно при транзакциях с несколькими входами, которые обязательно показывают, что их входы принадлежали одному и тому же владельцу. Риск заключается в том, что, если будет раскрыт владелец ключа, наличие связи может раскрыть другие транзакции, принадлежащие тому же владельцу.

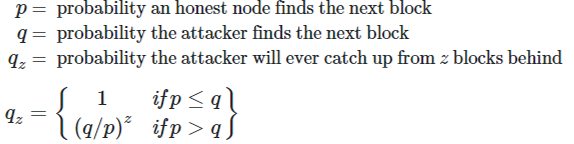
11. Вычисления

Мы рассматриваем сценарий, когда злоумышленник пытается создать альтернативную цепочку быстрее, чем создается честная. Даже если это будет выполнено, это не сделает систему открытой для произвольных изменений, таких как создание стоимости из воздуха или получение денег, которые никогда не принадлежали злоумышленнику. Узлы не будут принимать недействительную транзакцию в качестве оплаты, а честные узлы никогда не примут блок, содержащий их. Злоумышленник может попытаться изменить только одну из своих транзакций, чтобы вернуть недавно потраченные деньги.

Гонку между честной цепью и цепочкой атакующего можно охарактеризовать как биномиальное случайное блуждание. Событие успеха - это расширение честной цепочки

by one block, increasing its lead by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by -1.

The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, as follows[[8]](https://nakamotoinstitute.org/bitcoin/" \l "fn8):

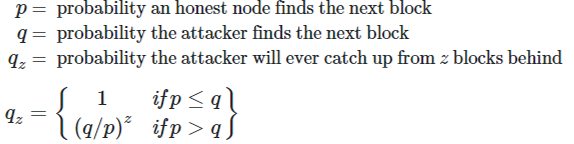


Given our assumption that *p>q*, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he doesn't make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind.

We now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can't change the transaction. We assume the sender is an attacker who wants to make the recipient believe he paid him for a while, then switch it to pay back to himself after some time has passed. The receiver will be alerted when that happens, but the sender hopes it will be too late.

на один блок, увеличивая его опережение на +1, и событием провала является расширение цепочки атакующего на один блок, уменьшая разрыв на -1.

Вероятность того, что злоумышленник наверстает возникший дефицит, аналогична вероятности из проблемы разорения игрока. Предположим, что игрок с неограниченным кредитом начинает с дефицита и потенциально играет бесконечное количество попыток, чтобы попытаться достичь безубыточности. Мы можем рассчитать вероятность того, что он когда-либо достигнет безубыточности, или что злоумышленник когда-либо догонит честную цепочку, следующим образом [8]

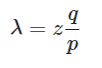


Учитывая наше предположение, что p> q, вероятность экспоненциально падает с увеличением количества блоков, которые атакующий должен догнать. Несмотря на то, что шансы против него, если он не сделает удачный рывок вперед на ранней стадии, его шансы станут исчезающе малы по мере того, как он будет все больше отставать.

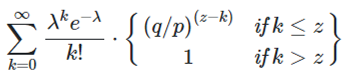
Теперь мы рассмотрим, как долго получатель новой транзакции должен ждать, прежде чем будет достаточно уверен, что отправитель не может изменить транзакцию. Мы предполагаем, что отправитель - это злоумышленник, который хочет заставить получателя на какое-то время поверить в то, что он заплатил ему, а затем поменять транзакцию на выплату самому себе по прошествии некоторого времени. Когда это произойдет, получатель будет предупрежден, но отправитель надеется, что будет слишком поздно.

The receiver generates a new key pair and gives the public key to the sender shortly before signing. This prevents the sender from preparing a chain of blocks ahead of time by working on it continuously until he is lucky enough to get far enough ahead, then executing the transaction at that moment. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

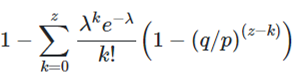
The recipient waits until the transaction has been added to a block and z blocks have been linked after it. He doesn't know the exact amount of progress the attacker has made, but assuming the honest blocks took the average expected time per block, the attacker's potential progress will be a Poisson distribution with expected value:



To get the probability the attacker could still catch up now, we multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:



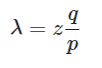
Rearranging to avoid summing the infinite tail of the distribution...



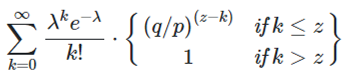
Converting to C code...

Получатель генерирует новую пару ключей и передает открытый ключ отправителю незадолго до подписания транзакции. Это не позволяет отправителю заранее подготовить цепочку блоков, работая над ней непрерывно, пока ему не посчастливится продвинуться достаточно далеко вперед, а затем выполнить транзакцию в этот момент. Когда транзакция отправлена, недобросовестный отправитель начинает секретно работать в параллельной цепочке, содержащей альтернативную версию его транзакции.

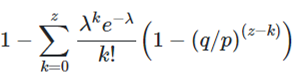
Получатель ждет, пока транзакция не будет добавлена в блок и после нее в цепочку будут добавлены z блоков. Он не знает точного прогресса, достигнутого злоумышленником, но если предположить, что на честные блоки затрачено среднее ожидаемое время на блок, то потенциальный прогресс злоумышленника будет иметь распределение Пуассона с ожидаемым значением:



Чтобы получить вероятность того, что атакующий на данный момент все еще может догнать (честную цепочку), мы умножаем плотность распределения Пуассона для каждого значения прогресса, которого он мог бы достичь, на вероятность того, что с этого момента он сможет догнать честную цепочку:



Перестановка, чтобы избежать суммирования бесконечного остатка распределения ...



Конвертируем в код на C:

#include

double AttackerSuccessProbability(double q, int z)

{

double p = 1.0 - q;

double lambda = z \* (q / p);

double sum = 1.0;

int i, k;

for (k = 0; k <= z; k++)

{

double poisson = exp(-lambda);

for (i = 1; i <= k; i++)

poisson \*= lambda / i;

sum -= poisson \* (1 - pow(q / p, z - k));

}

return sum;

}

Running some results, we can see the probability drop off exponentially with z.

q=0.1

z=0 P=1.0000000

z=1 P=0.2045873

z=2 P=0.0509779

z=3 P=0.0131722

z=4 P=0.0034552

z=5 P=0.0009137

z=6 P=0.0002428

z=7 P=0.0000647

z=8 P=0.0000173

z=9 P=0.0000046

z=10 P=0.0000012

q=0.3

z=0 P=1.0000000

z=5 P=0.1773523

z=10 P=0.0416605

z=15 P=0.0101008

z=20 P=0.0024804

z=25 P=0.0006132

z=30 P=0.0001522

z=35 P=0.0000379

z=40 P=0.0000095

z=45 P=0.0000024

z=50 P=0.0000006

Solving for P less than 0.1%...

P < 0.001

q=0.10 z=5

q=0.15 z=8

q=0.20 z=11

q=0.25 z=15

q=0.30 z=24

q=0.35 z=41

q=0.40 z=89

q=0.45 z=340

#include

double AttackerSuccessProbability(double q, int z)

{

double p = 1.0 - q;

double lambda = z \* (q / p);

double sum = 1.0;

int i, k;

for (k = 0; k <= z; k++)

{

double poisson = exp(-lambda);

for (i = 1; i <= k; i++)

poisson \*= lambda / i;

sum -= poisson \* (1 - pow(q / p, z - k));

}

return sum;

}

Получив некоторые результаты, мы можем увидеть, что вероятность экспоненциально спадает с увеличением z.

q=0.1

z=0 P=1.0000000

z=1 P=0.2045873

z=2 P=0.0509779

z=3 P=0.0131722

z=4 P=0.0034552

z=5 P=0.0009137

z=6 P=0.0002428

z=7 P=0.0000647

z=8 P=0.0000173

z=9 P=0.0000046

z=10 P=0.0000012

q=0.3

z=0 P=1.0000000

z=5 P=0.1773523

z=10 P=0.0416605

z=15 P=0.0101008

z=20 P=0.0024804

z=25 P=0.0006132

z=30 P=0.0001522

z=35 P=0.0000379

z=40 P=0.0000095

z=45 P=0.0000024

z=50 P=0.0000006

Решаем для Р меньше 0.1%...

P < 0.001

q=0.10 z=5

q=0.15 z=8

q=0.20 z=11

q=0.25 z=15

q=0.30 z=24

[12. Conclusion](https://nakamotoinstitute.org/bitcoin/#conclusion)

We have proposed a system for electronic transactions without relying on trust. We started with the usual framework of coins made from digital signatures, which provides strong control of ownership, but is incomplete without a way to prevent double-spending. To solve this, we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity. Nodes work all at once with little coordination. They do not need to be identified, since messages are not routed to any particular place and only need to be delivered on a best effort basis. Nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone. They vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism.

[References](https://nakamotoinstitute.org/bitcoin/#references)

1. W. Dai, ["b-money,"](https://nakamotoinstitute.org/b-money/) <http://www.weidai.com/bmoney.txt>, 1998. [↩](https://nakamotoinstitute.org/bitcoin/#ref1)
2. H. Massias, X.S. Avila, and J.-J. Quisquater, ["Design of a secure timestamping service with minimal trust](https://nakamotoinstitute.org/secure-timestamping-service.pdf)
3. [requirements,"](https://nakamotoinstitute.org/secure-timestamping-service.pdf) In *20th Symposium on Information Theory in the Benelux*, May 1999. [↩](https://nakamotoinstitute.org/bitcoin/#ref2) [↩](https://nakamotoinstitute.org/bitcoin/#ref2-2)

12. Заключение

Мы предложили систему электронных транзакций, не полагаяющуюся на доверие. Мы начали с обычной структуры монет, созданных на основе цифровых подписей, которая обеспечивает строгий контроль за владением, но является неполной без возможности предотвратить двойные траты. Чтобы решить эту проблему, мы предложили одноранговую сеть, использующую механизм доказательства-работой для записи общедоступной истории транзакций, которая быстро становится вычислительно непрактичной для изменения злоумышленником, если честные узлы контролируют большую часть мощности ЦП. Сеть сильна в своей неструктурированной простоте. Узлы работают одновременно с небольшой координацией. Им не нужно быть идентифицированными, поскольку сообщения не направляются в какое-либо конкретное место, а должны быть доставлены только с максимальной эффективностью. Узлы могут покидать сеть и снова присоединяться к ней по своему желанию, принимая цепочку доказательства-работой как доказательство того, что произошло, пока они отсутствовали. Они голосуют вычислительной мощностью своего процессора, выражая свое согласие с действительными блоками, работая над их расширением, и отклоняя недействительные блоки, отказываясь работать с ними. С помощью этого механизма консенсуса можно обеспечить соблюдение любых необходимых правил и стимулов.

Список литературы

1. W. Dai, ["b-money,"](https://nakamotoinstitute.org/b-money/) <http://www.weidai.com/bmoney.txt>, 1998. [↩](https://nakamotoinstitute.org/bitcoin/#ref1)
2. H. Massias, X.S. Avila, and J.-J. Quisquater, ["Design of a secure timestamping service with minimal trust](https://nakamotoinstitute.org/secure-timestamping-service.pdf)
3. [requirements,"](https://nakamotoinstitute.org/secure-timestamping-service.pdf) In *20th Symposium on Information Theory in the Benelux*, May 1999. [↩](https://nakamotoinstitute.org/bitcoin/#ref2) [↩](https://nakamotoinstitute.org/bitcoin/#ref2-2)
4. S. Haber, W.S. Stornetta, ["How to time-stamp a digital document,"](https://nakamotoinstitute.org/time-stamp-digital-document.pdf) In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991. [↩](https://nakamotoinstitute.org/bitcoin/#ref2)
5. D. Bayer, S. Haber, W.S. Stornetta, ["Improving the efficiency and reliability of digital time-stamping,"](https://nakamotoinstitute.org/improving-time-stamping.pdf) In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993. [↩](https://nakamotoinstitute.org/bitcoin/#ref2)
6. S. Haber, W.S. Stornetta, ["Secure names for bit-strings,"](https://nakamotoinstitute.org/secure-names-bit-strings.pdf) In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997. [↩](https://nakamotoinstitute.org/bitcoin/#ref2) [↩](https://nakamotoinstitute.org/bitcoin/#ref5)
7. A. Back, ["Hashcash - a denial of service counter-measure,"](https://nakamotoinstitute.org/hashcash.pdf) <http://www.hashcash.org/papers/hashcash.pdf>, 2002. [↩](https://nakamotoinstitute.org/bitcoin/#ref6)
8. R.C. Merkle, ["Protocols for public key cryptosystems,"](https://nakamotoinstitute.org/public-key-cryptosystems.pdf) In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980. [↩](https://nakamotoinstitute.org/bitcoin/#ref7)
9. W. Feller, ["An introduction to probability theory and its applications,"](https://nakamotoinstitute.org/introduction-probability-theory-vol-i.pdf) 1957. [↩](https://nakamotoinstitute.org/bitcoin/#ref8)
10. S. Haber, W.S. Stornetta, ["How to time-stamp a digital document,"](https://nakamotoinstitute.org/time-stamp-digital-document.pdf) In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991. [↩](https://nakamotoinstitute.org/bitcoin/#ref2)
11. D. Bayer, S. Haber, W.S. Stornetta, ["Improving the efficiency and reliability of digital time-stamping,"](https://nakamotoinstitute.org/improving-time-stamping.pdf) In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993. [↩](https://nakamotoinstitute.org/bitcoin/#ref2)
12. S. Haber, W.S. Stornetta, ["Secure names for bit-strings,"](https://nakamotoinstitute.org/secure-names-bit-strings.pdf) In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997. [↩](https://nakamotoinstitute.org/bitcoin/#ref2) [↩](https://nakamotoinstitute.org/bitcoin/#ref5)
13. A. Back, ["Hashcash - a denial of service counter-measure,"](https://nakamotoinstitute.org/hashcash.pdf) <http://www.hashcash.org/papers/hashcash.pdf>, 2002. [↩](https://nakamotoinstitute.org/bitcoin/#ref6)
14. R.C. Merkle, ["Protocols for public key cryptosystems,"](https://nakamotoinstitute.org/public-key-cryptosystems.pdf) In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980. [↩](https://nakamotoinstitute.org/bitcoin/#ref7)
15. W. Feller, ["An introduction to probability theory and its applications,"](https://nakamotoinstitute.org/introduction-probability-theory-vol-i.pdf) 1957. [↩](https://nakamotoinstitute.org/bitcoin/#ref8)

Глоссарий:

1. Digital signatures

2. proof-of-work

3. hash

4. distributed timestamp server

5. incrementing

6. moving average

7. Merkle Tree

8. public keys

9. Binomial Random Walk

10. Gambler's Ruin problem

11. Poisson distribution

12. Poisson density

Примечание: я не нашел в тексте каких-то особо редких технических фраз или терминов, т.к. эта статья специально написана для широкого круга людей и по задумке автора должна была быть максимально понятной и простой.