Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto

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.

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.

Abstract

純粋な P2P 版電子通貨は金融機関を通すことなく二者間で直接オンライン支払いを可能にする。電子署名は部分的な解決を提供するが、二重使用を防止するために信用できる第三者機関が依然として必要な場合、主要な利点は失われる。我々は P2P ネットワークを用いて二重使用問題の解決策を提案する。そのネットワークは、ハッシュベースの PoW の継続的なチェーンにトランザクションをハッシュすることによりタイムスタンプしており、PoW をやり直さない限り変更することができない記録を形成する。最も長いチェーンは署名されたイベントの一連の証拠としてだけでなく、最も大きい CPU パワーのプールから来ていることの証拠としても機能する。過半数の CPU パワーがネットワークを攻撃することに協力する悪意のあるノードに管理されない限り、最も長いチェーンは攻撃者らのチェーンを凌ぐように生成される。ネットワークは最小限の構造を必要としている。メッセージはベストエフォート原則でブロードキャストされ、ノードは意のままにネットワークから離脱と復帰することができ、離脱していた間に起きたことの証明として最も長い PoW チェーンを受け取ることができる。

1. Introduction

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.

「信頼」の代わりに必要なものは、暗号による証明をベースとした電子支払いシステムであり、第三者機関を必要とせずに当事者間で直接取引を行うことが可能になる。計算的にひっくり返すことが非現実的なトランザクションは、売り手を詐欺から守ることができ、簡単に実装できる第三者に委託されているメカニズムによって買い手も守ることができる。

1. Introduction

In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to 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.

この論文では、トランザクションの時系列の計算的証明を生成する P2P に分散されたタイムスタンプサーバを用いることで二重使用問題の解決策を提案する。システムは誠実なノードたちが攻撃者たちの CPU パワーよりも多くを制御している限りセキュアである。

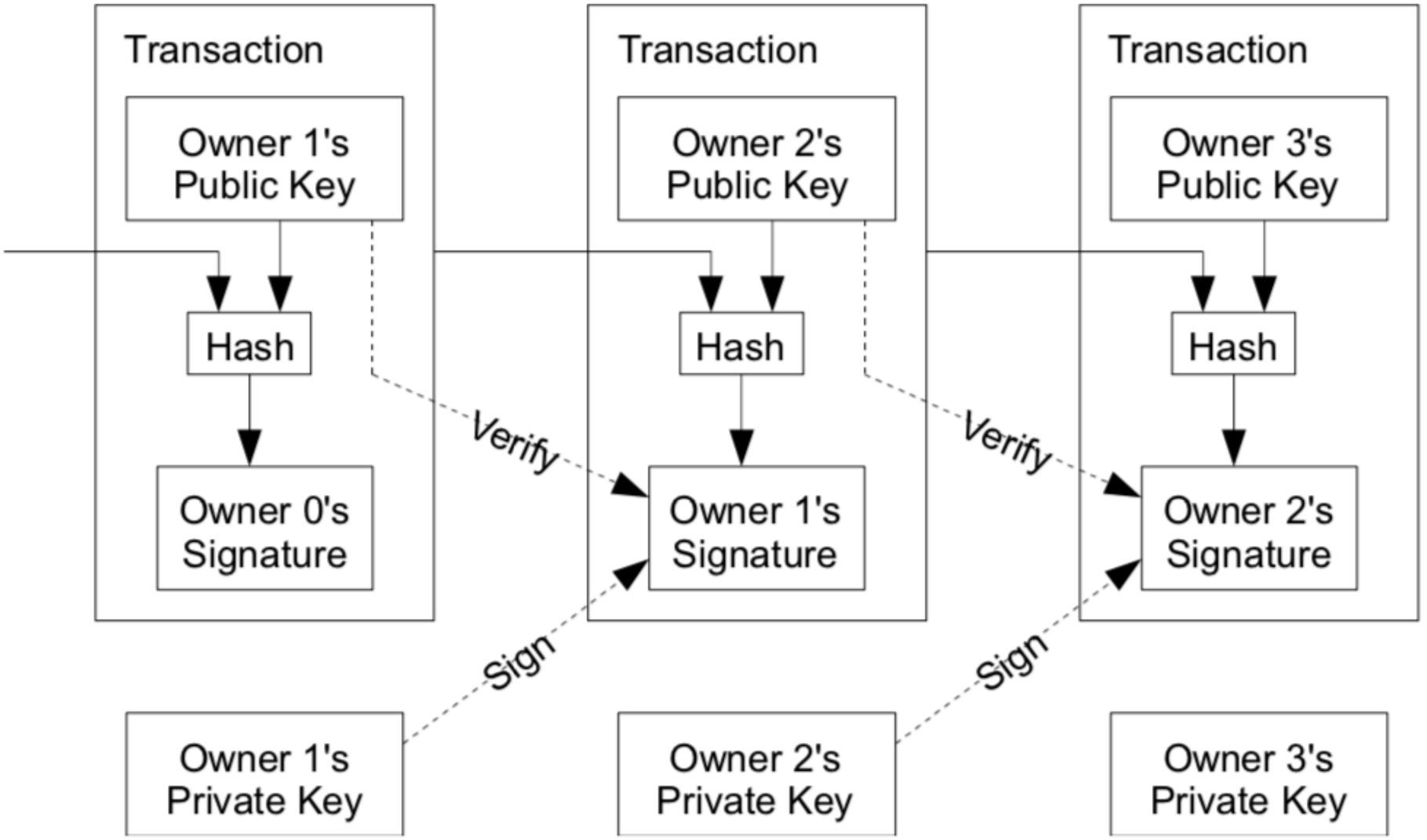
.

2. 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.

我々は電子署名のチェーンとして電子コインを定義する。各所有者は前トランザクションのハッシュ値と次の所有者の公開鍵に電子署名を行い、コインの最後に追加することでコインを次の所有者に移動させることができる。受取人は所有者のチェーンを検証することで電子署名を検証することができる。

.


Transaction
Owner 1's Public Key
Hash
Owner 0's Signature
Transaction
Owner 2's Public Key
Hash
Owner 1's Signature
Transaction
Owner 3's Public Key
Hash
Owner 2's Signature
Verify
Verify
Sign
Sign
Owner 1's Private Key
Owner 2's Private Key
Owner 3's Private Key

2. Transactions

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 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.

問題は受取人が所有者の一人が二重使用をしていないことを検証できないことである。一般的な解決策は、二重使用していないか全てのトランザクションを確認する信頼できる中央当局や造幣局を導入することだ。各トランザクションの後、新たにコインを発行するために、コインは必ず造幣局に戻らなければならない。そして、造幣局から直接発行したコインのみが二重使用されていないと信頼される。この解決策の問題は、システム全体の運命がちょうど銀行のように全てのトランザクションが通過する造幣局として運営している企業に依存することだ。

2. Transactions

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.

以前の所有者が以前のトランザクションに署名しなかったことを知る方法が受取人のために必要である。我々の目的では最も早いトランザクションは 1 つのため、二重使用のためにその後試行されたものは注意する必要がない。トランザクションが存在しないことを確認する唯一の方法は全トランザクションを確認することである。造幣局ベースモデルでは、造幣局は全トランザクションを確認することができ、どれが最初に届いたかを決定することができる。

2. Transactions

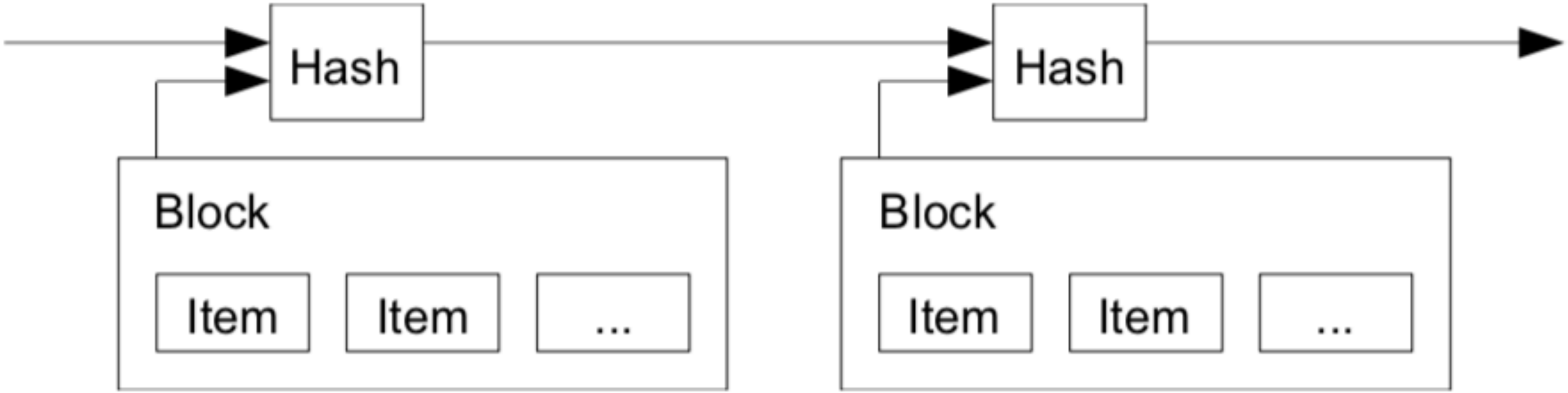
To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

信頼できる機関なしでこの問題を達成するには、トランザクションが公に知らされ、参加者が受け取った順番の単一履歴を同意するシステムが必要である。受取人は、各トランザクションの時点で、過半数ノードがそれが最初に受け取られたことに同意したという証拠を必要とする。

3. 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]. 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.

我々が提案する解決策はタイムスタンプサーバから始まる。タイムスタンプサーバはタイムスタンプされるアイテムのブロックのハッシュ値を取り、新聞や Usenet post のように広範囲に公開する。タイムスタンプサーバは、ハッシュ値を取るためにそのデータがその時に間違いなく存在していたことを証明する。各タイムスタンプはそのハッシュに前タイムスタンプが含まれるため、追加される各タイムスタンプは前タイムスタンプを強化する。


Hash
Hash
Block
Item
Item
...
Block
Item
Item
...

4. Proof-of-Work

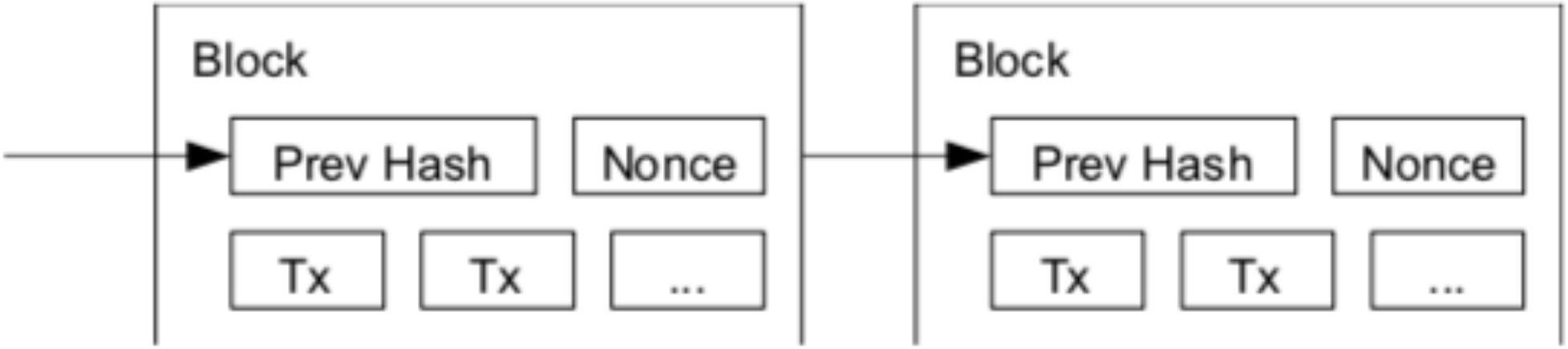
To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof- of-work system similar to Adam Back's Hashcash [6]. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

P2P ベースに分散したタイムスタンプサーバを実装することは Adam Back’s Hashcash と似た PoW システムを使う必要がある。PoW は SHA-256 のようなハッシュ値を取った時に、多くのゼロビットで始まるハッシュ値の走査を含んでいる。その作業はゼロビットの数に対して平均して指数的に増加し、 1 回のハッシュを実行することで検証が可能である。

4. 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]. 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.

P2P ベースに分散したタイムスタンプサーバを実装することは Adam Back’s Hashcash と似た PoW システムを使う必要がある。PoW は SHA-256 のようなハッシュ値を取った時に、多くのゼロビットで始まるハッシュ値の走査を含んでいる。その作業はゼロビットの数に対して平均して指数的に増加し、1 回のハッシュを実行することで検証が可能である。我々のタイムスタンプネットワークでは、ブロックのハッシュ値が要求された数のゼロビットになるまでブロック内のナンスをインクリメントすることによって PoW が実装されている。


Block
Prev Hash
Nonce
Tx
Tx
...
Block
Prev Hash
Nonce
Tx
Tx
...

4. Proof-of-Work

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. 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.

PoW は多数決によって代表者を決める問題も解決する。もし多数決において 1 つの IPアドレスが 1 票を持っていたとすると、多くの IP アドレスが割り当てられたものは誰でも覆すことができる。PoW は基本的に 1 つの CPU が 1 票を持っている。最も多くの PoW エフォートが費やされた最も長いチェーンによって多数決は表される。CPU パワーの過半数が誠実なノードによって制御されている場合、誠実なチェーンが最も早く成長し競争する他のチェーンを打ち負かす。

4. Proof-of-Work

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.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.

過去のブロックを修正するために攻撃者はそのブロック以降の全ての PoW をやり直す必要があり、誠実なノードのチェーンを追い越す必要がある。ハードウェア速度の増加と実行中ノードの関心の変化を補うために、PoW 難易度は時間当たりの平均ブロック数を対象とした平均移動によって決定される。もしとても早くブロックが生成された場合、難易度は増加する。

5. 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.
6) 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.

ネットワークの実行手順は以下の通りである。

1. 新しいトランザクションは全てのノードにブロードキャストする
2. 各ノードは新しいトランザクションをブロック内に集める
3. 各ノードはそのブロックの PoW を見つける
4. ノードが PoW を見つけた時、ブロックを全てのノードにブロードキャストする
5. ブロックに入っている全トランザクションが正当であり二重使用でない場合、ノードはブロックを受け取る
6. ノードは受け取ったブロックのハッシュ値を前ハッシュとして、チェーンの次ブロックを作ることでブロックの受理を表す

5. Network

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.

ノードは常に最も長いチェーンが正しいものと見なしており、それを拡大しようとする。もし2つのノードが異なる次のブロックを同時にブロードキャストした場合、他のノードは最初に受け取ったブランチで動作するが、他のブランチがより長くる場合に備えて他のブランチも保存しておく。この分岐点は次の PoW が発見されどちらかのブランチがより長くなった時になくなる。他のブランチで作業していたノードは長くなったブランチに切り替える。

5. Network

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.

新しいトランザクションのブロードキャストは全てのノードに届くために必ずしも必要ではない。各ノードが多くのノードと繋がっている限り、ブロックが長くなる前にそれを受け取ることができる。ブロックブロードキャストはメッセージ破棄の耐性にもなる。ノードはブロックを受け取らなかった場合、次ブロックを受け取り間のブロックが欠けていることに気づいた時にそれを要求することができる。

6. 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.

慣例により、ブロックの最初のトランザクションはそのブロックの作成者が所持する新しいコインを始める特別なトランザクションである。コインを発行する中央当局が存在しないため、これはネットワークを支えるためのインセンティブをノードに与え、最初にコインを分配する方法を提供する。固定された定数量の新しいコインを追加することは、流通に金を追加するために資源を消費する金鉱夫に似ている。今回のケースではCPU時間と電気が消費される。

6. Incentive

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.

インセンティブはトランザクション手数料で賄うこともできる。トランザクションの出力値が入力値よりも小さい場合、その違いはそのトランザクションを含むブロックのインセンティブ値（手数料）に追加される。流通に事前に決められた量のコインが入ったら、そのインセンティブは全体的に手数料に移行することができ、完全にインフレ料金になる。

6. Incentive

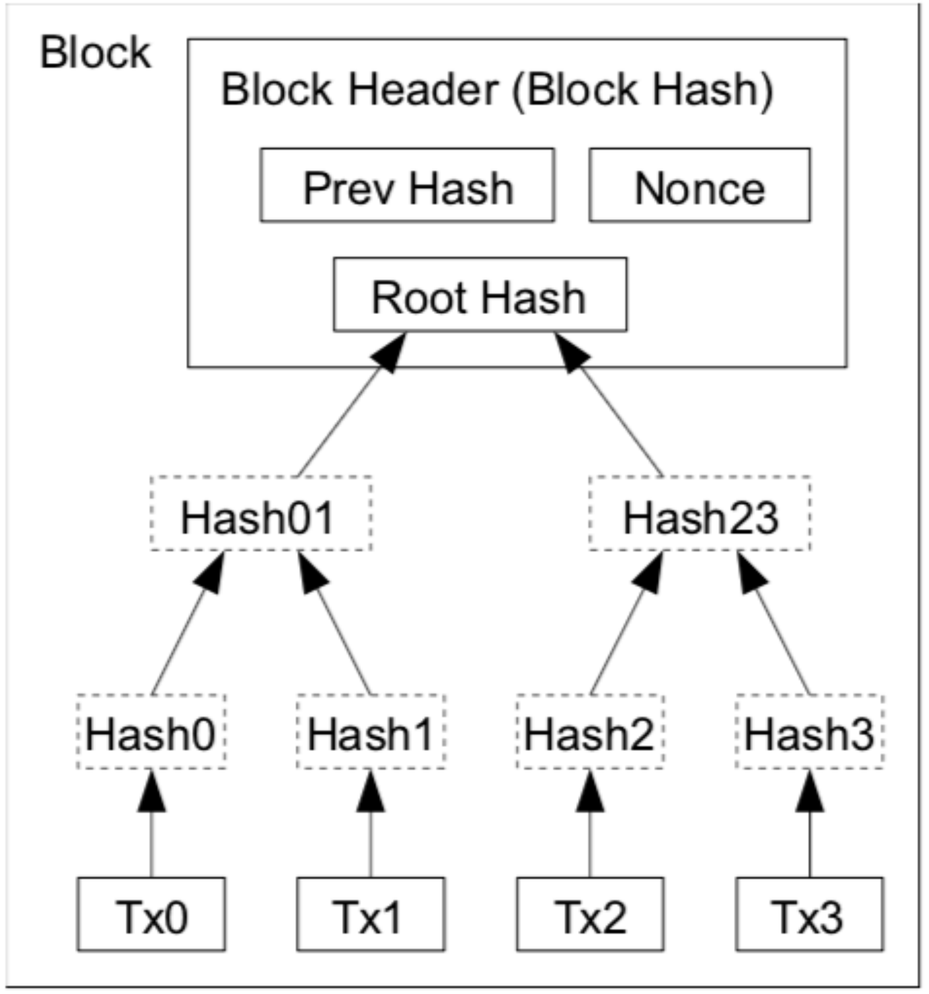
The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to 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.

そのインセンティブはノードたちを誠実であり続ける動機付けになる。もし貪欲な攻撃者が誠実なノードたちよりも多くの CPU パワーを集めることができた場合、攻撃者は支払いを盗んで人々を騙すために使用するか、それを使用して新しいコインを生成するかを選択する必要がある。システムと自身の富の正当性を損なうよりも、他の誰もが結合したよりも多くの新しいコインが彼を支持するようなルールで動作するほうが、より利益があると思うべきである。

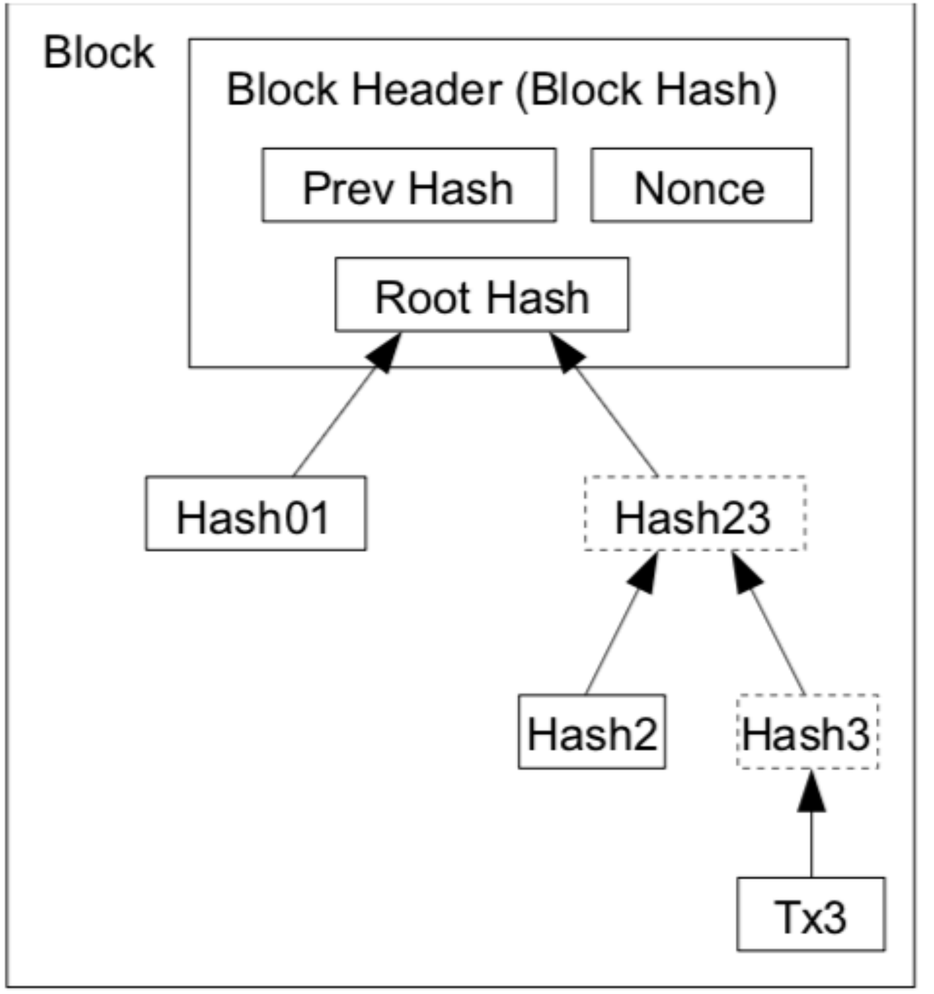
7. Reclaiming Disk Space

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][2][5], 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.

コインの最新トランザクションが一度十分な数のブロックに埋もれたら、それ以前の使用済みトランザクションはディスクイメージを守るために破棄することができる。ブロックのハッシュを壊すことなくこれを促すために、トランザクションたちはブロックヘッダーにルートだけを含むマークルツリーにハッシュされる。古いブロックたちは木の枝を切り取ることによって圧縮することができる。内部ハッシュは保存しておく必要はない。



Transactions Hashed in a Merkle Tree

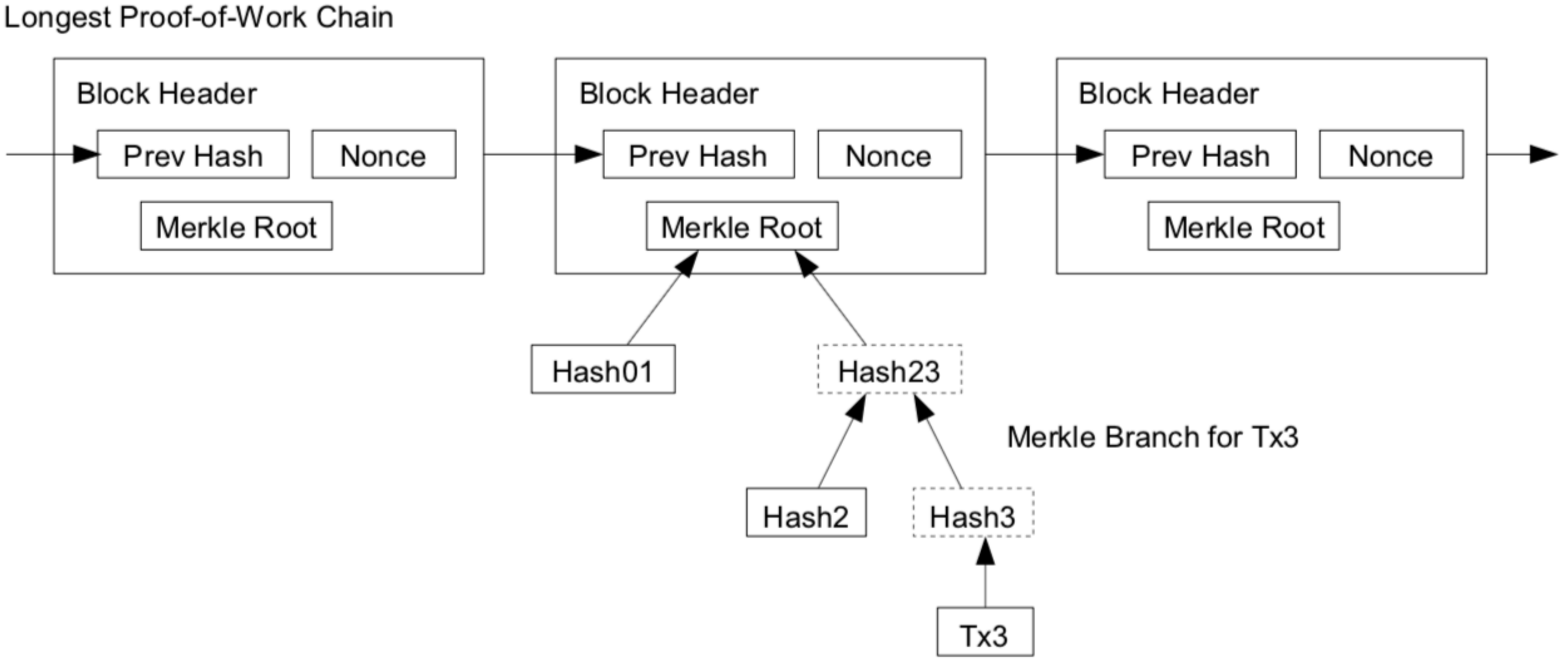


After Pruning Tx0-2 from the Block

8. 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.

支払い検証はネットワークのフルノードを実行せずに可能である。ユーザは最も長い PoW チェーンのブロックヘッダーのコピーだけをもっておく必要がある。これは自分が最も長いチェーンを持っていると確信できるまでネットワークのノードにクエリを投げることで取得でき、そのトランザクションがブロックにリンクするマークルブランチを取得できる。そのトランザクションを自分では確認することができないが、チェーン内の場所にリンクすることでノードがそれを受理していることを確認できる。


Longest Proof-of-Work Chain
Block Header
Prev Hash
Nonce
Merkle Root
Block Header
Prev Hash
Nonce
Merkle Root
Block Header
Prev Hash
Nonce
Merkle Root
Hash01
Hash23
Merkle Branch for Tx3
Hash2
Hash3
Tx3

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

コインを個別に処理することは可能であるが、トランザクション料金をセントごとに分けることは扱いにくいだろう。価値の分離や結合を可能にするためにトランザクションは複数の入力と出力を含んでいる。通常は、大きな前のトランザクションからの単一の入力か、少量を結合させた複数の入力のどちらかであり、出力は1つは支払い、もう1つはお釣りが発生した場合の送信者への返却のためのものである。

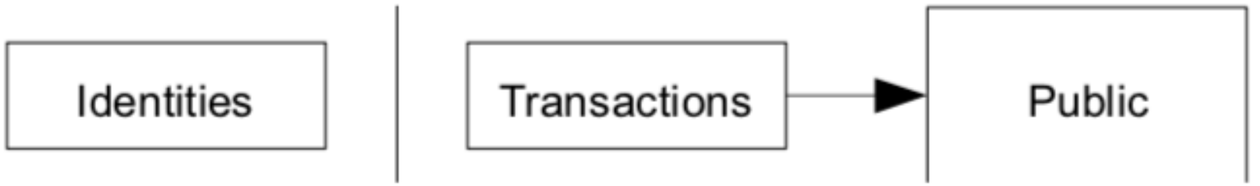
10. Privacy

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.

伝統的な銀行モデルでは、情報へのアクセスを関係団体や信頼できる第三者機関に限定することでプライバシーレベルに達する。全トランザクションを公に知らせる必要性はこのメソッドを妨げるが、他の場所で情報の流れを分断することでプライバシーは維持される。つまり公開鍵を匿名にする。公には誰かが他の誰かにある量を送ったことを確認できるが、そのトランザクションを誰かに結びつけることはできない。

Traditional Privacy Model

New Privacy Model



10. Privacy

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.

追加のファイアウォールとして、各トランザクションごとに共通の所持者にリンクされる新しい鍵ペアを使うべきである。複数入力のトランザクションではそれらの鍵が同じ所持者ということが明らかになってしまうことがリスクである。

12. 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.

我々は信頼への依存することのない電子取引のシステムを提案した。我々は所有権の強い制御を提供する電子署名からできている通常のコインのフレームワークから開始しているが、二重使用を防ぐ方法なしでは不完全である。これを解決するためにトランザクションの公式履歴を記録するために PoW を用いた P2P ネットワークを提案した。トランザクションは誠実なノードが CPU パワーの過半数を制御している場合、攻撃者はそれを変更することは計算的に非現実的となる。ネットワークは、構造化されていないシンプルさで堅牢である。

12. Conclusion

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.

ノードはわずかな調整で一斉に動く。メッセージは特定の場所にルーティングされるわけではなく、ベストエフォート方式で配送されることだけが必要であるため、それらは特定される必要はない。ノードは離脱と復帰を意のままに行うことができ、離脱していた間に何が起きたかの証明として PoW チェーンを受け取ることができる。CPU パワーで投票し、チェーンを拡張することで正当なブロックの受理を表現する。そして、正当でないと判断した場合は拒否することで拒絶を表現する。このコンセンサスメカニズムで必要なルールやインセンティブを遵守させることができる。