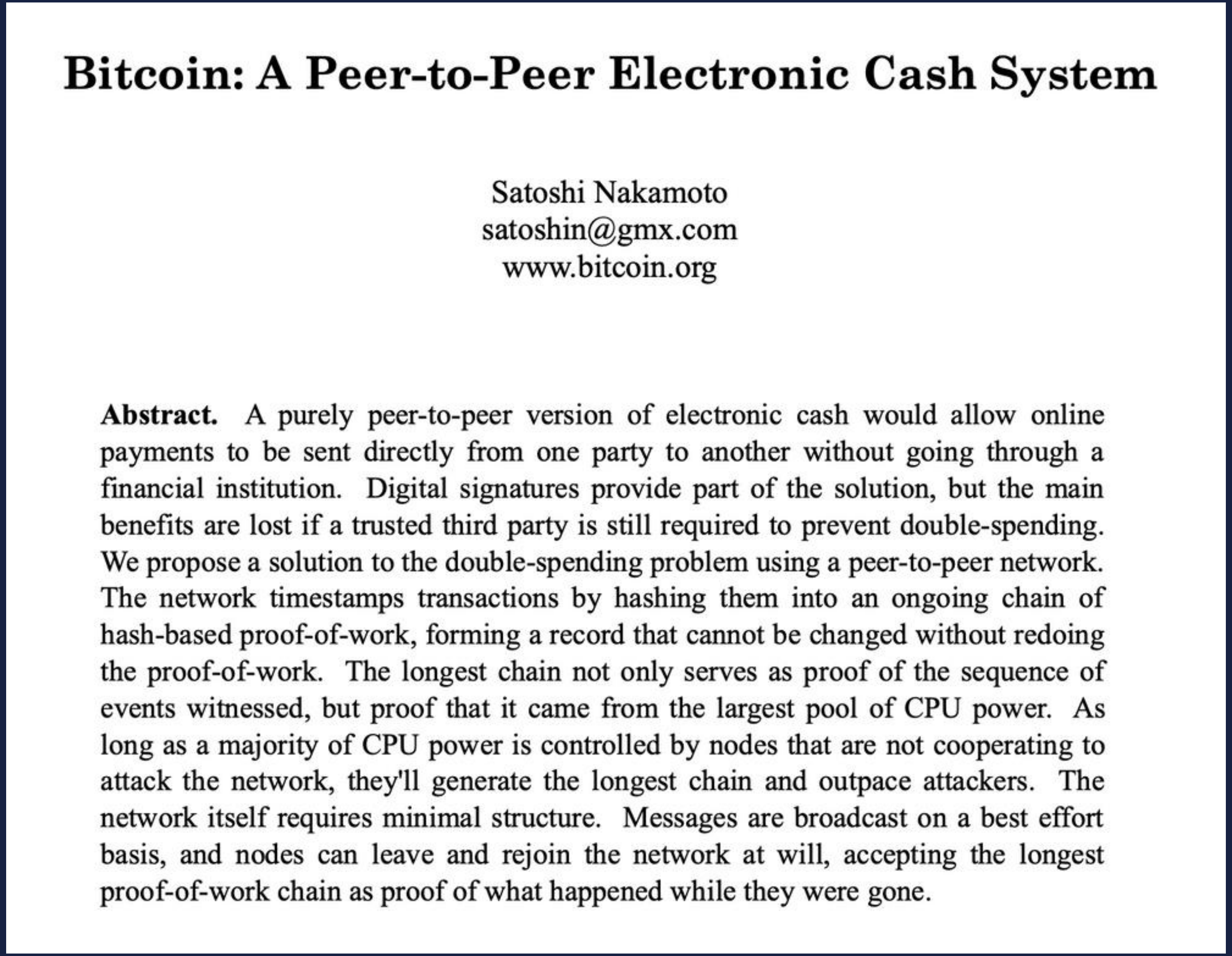
# *Bitcoin*

株式会社Neo Breakthrough
山科 優希

# Bitcoin Whitepaperを読む

## 概要


**Bitcoin: A Peer-to-Peer Electronic Cash System**

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

**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.

*"完全なP2P電子通貨の実現により、金融機関の介在無しに、利用者同士の直接的なオンライン決済が可能となるだろう”*

特徴的なのはBitcoin以前のP2P技術を用いた代表的なシステムのNapsterと比較して、ピュアP2P型である点。
*1960*年代前半に米国の*Paul Baran*が発表した論文*”On Distributed Communications Networks”*内のネットワーク分類の図中*”Distributed”*（分散型）に該当。

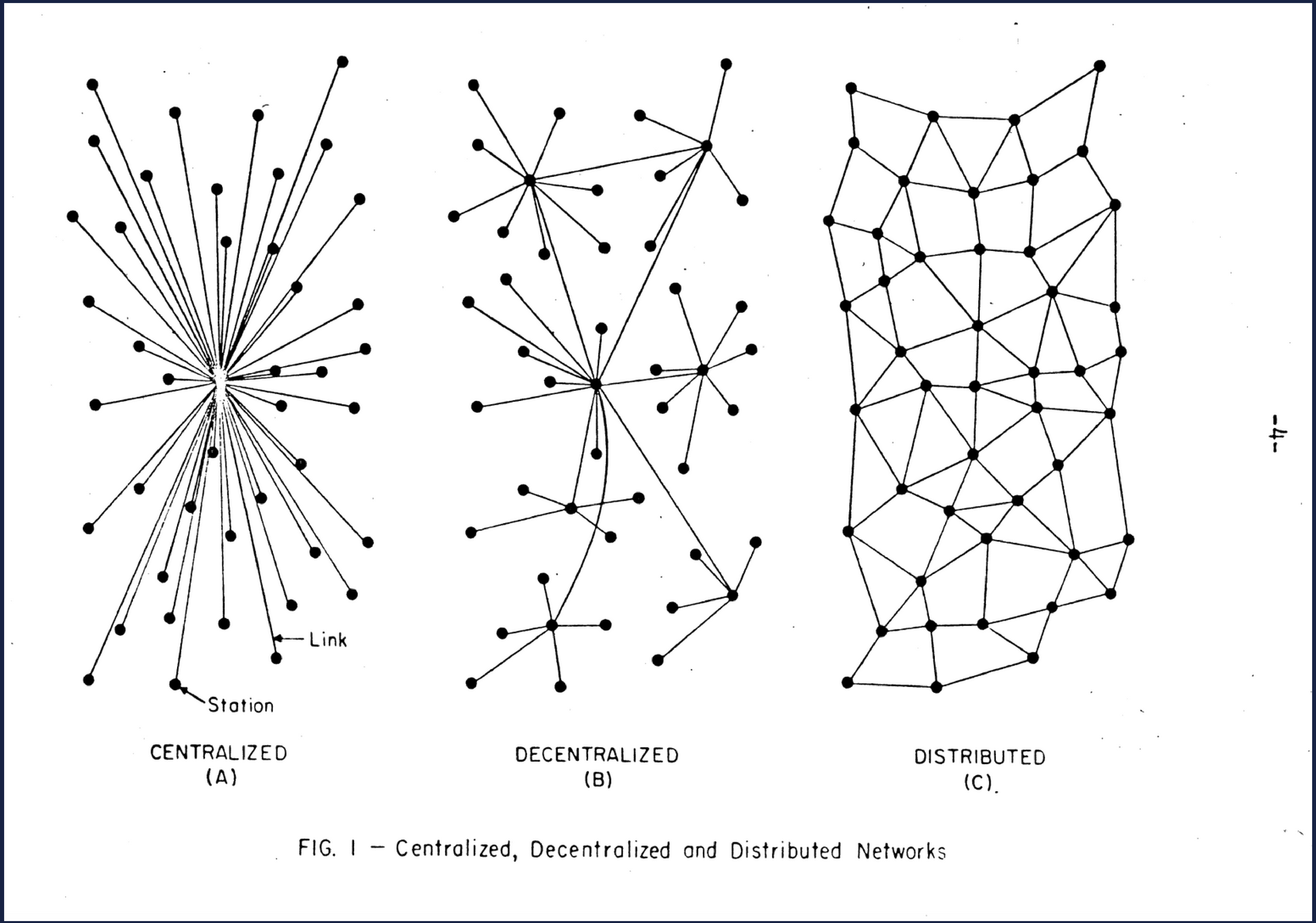
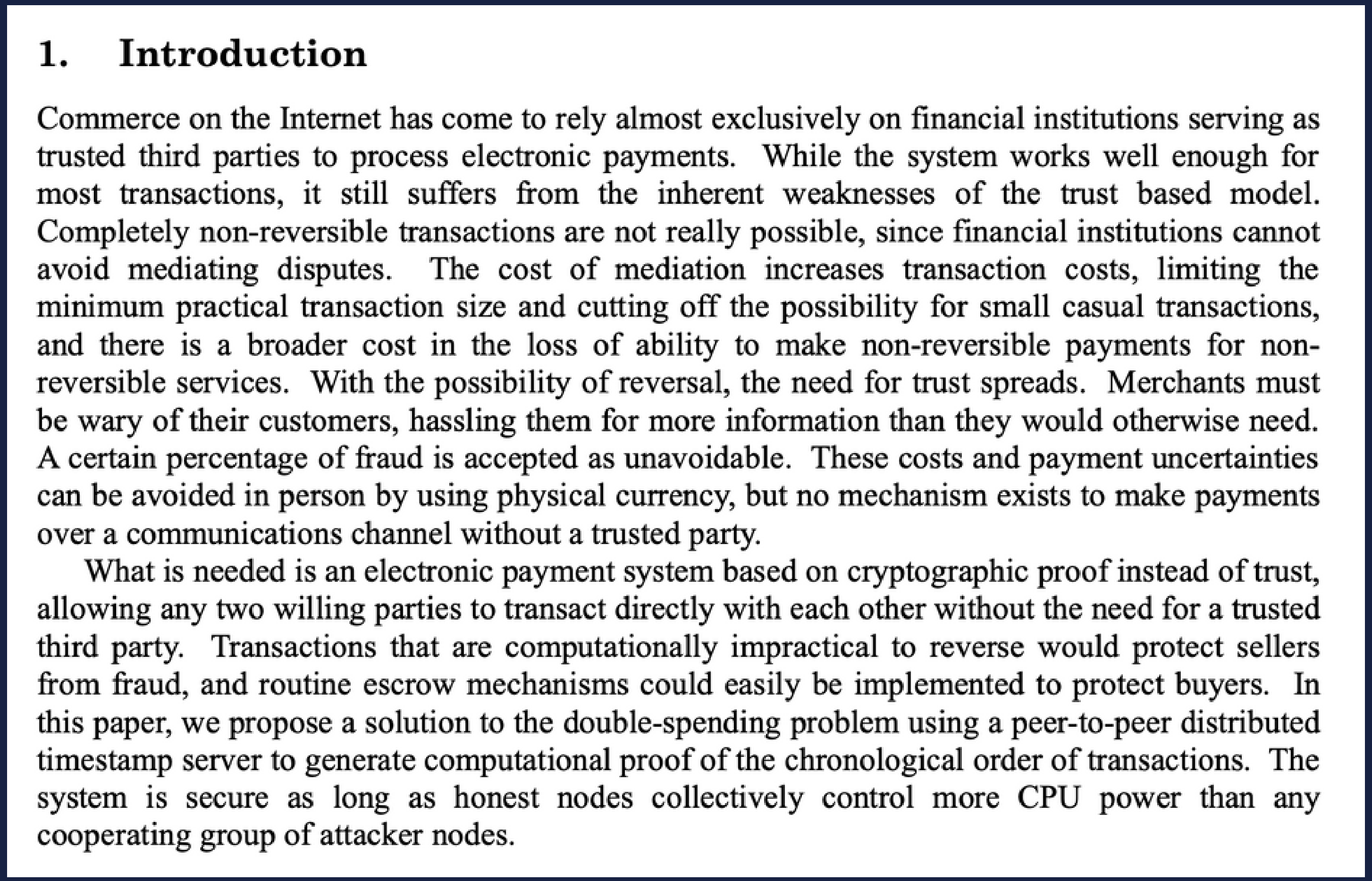


FIG. I — Centralized, Decentralized and Distributed Networks

出典：https://www.rand.org/content/dam/rand/pubs/papers/2005/P2626.pdf

# *1*章：序論



**1. 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 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.

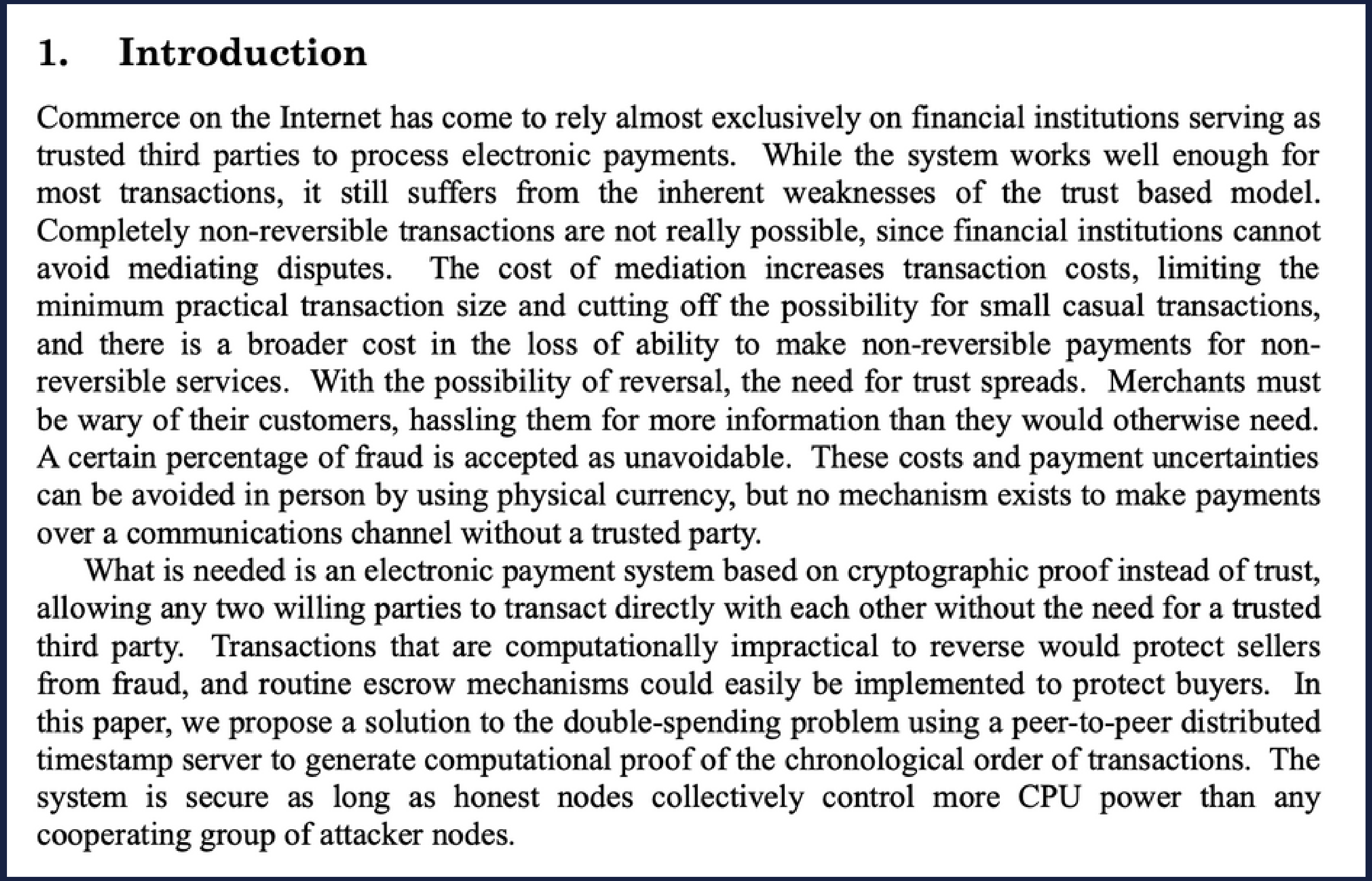
"現在のインターネット上の商取引は殆ど例外なく、電子取引を処理する信用の置ける第三者の金融機関に依存している”

金融機関をTTPとしており、これは電子時刻印（1991年）のタイムスタンプサービスに該当。

*Satoshi*は*TTP*にトラストが生じる部分の脆弱性を指摘してこのような説明をしている。

（2001年のNick Szaboの論文に”Trusted Third Parties are Security Holes”（TTPはセキュリティホールである）という論文があり、上記引用文の指摘根拠となっているのではないか。）

# *1*章：序論



## 1. 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.

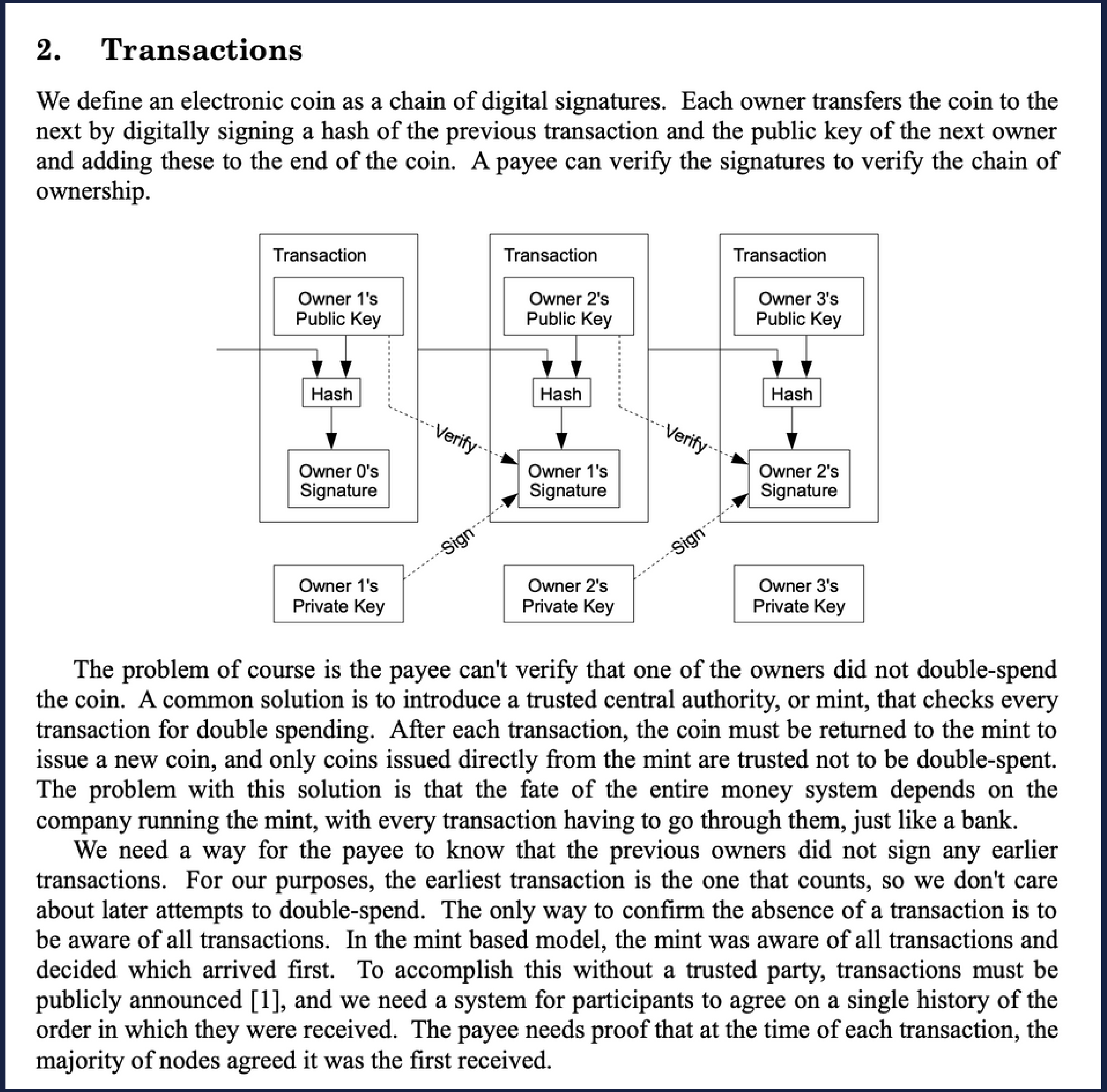
What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, 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 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.

"必要なのは信用ではなく、暗号学的証明に基づいた電子取引システムであり、これにより信用の置ける第三者を介さずに、利用者間の直接取引が可能となる”

→*TTP*となる存在はないが、実質的に暗号学的証明をトラストしている。
Trust-lessなシステムに見えるが暗号学的証明をトラストしているが
故に、”Trust-lessなTrust”と表現される。

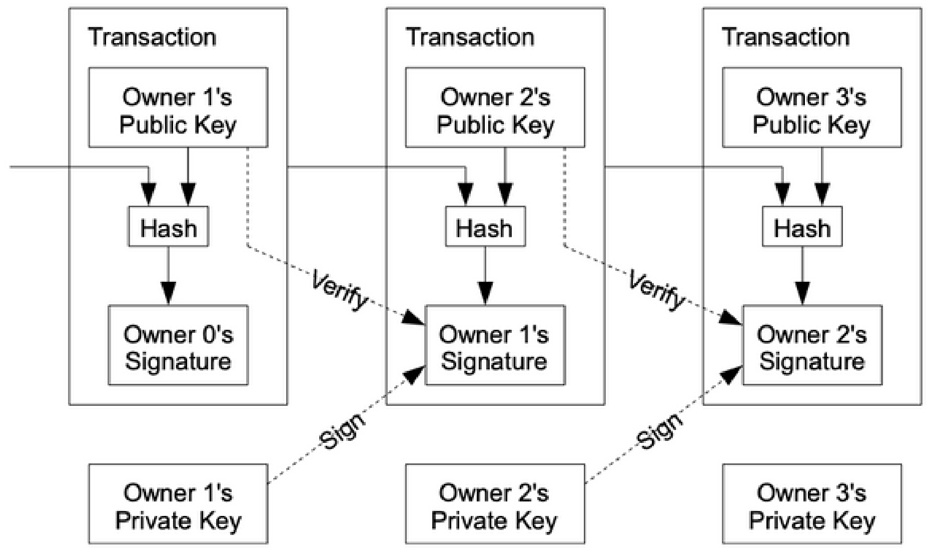
”本論文では取引が時系列に行われたかについて、計算に基づいた証明を生成する*P2P*分散型タイムスタンプサーバを使用し、二重支払い問題の解決策を提案する”

# 2章：トランザクション



**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.



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.

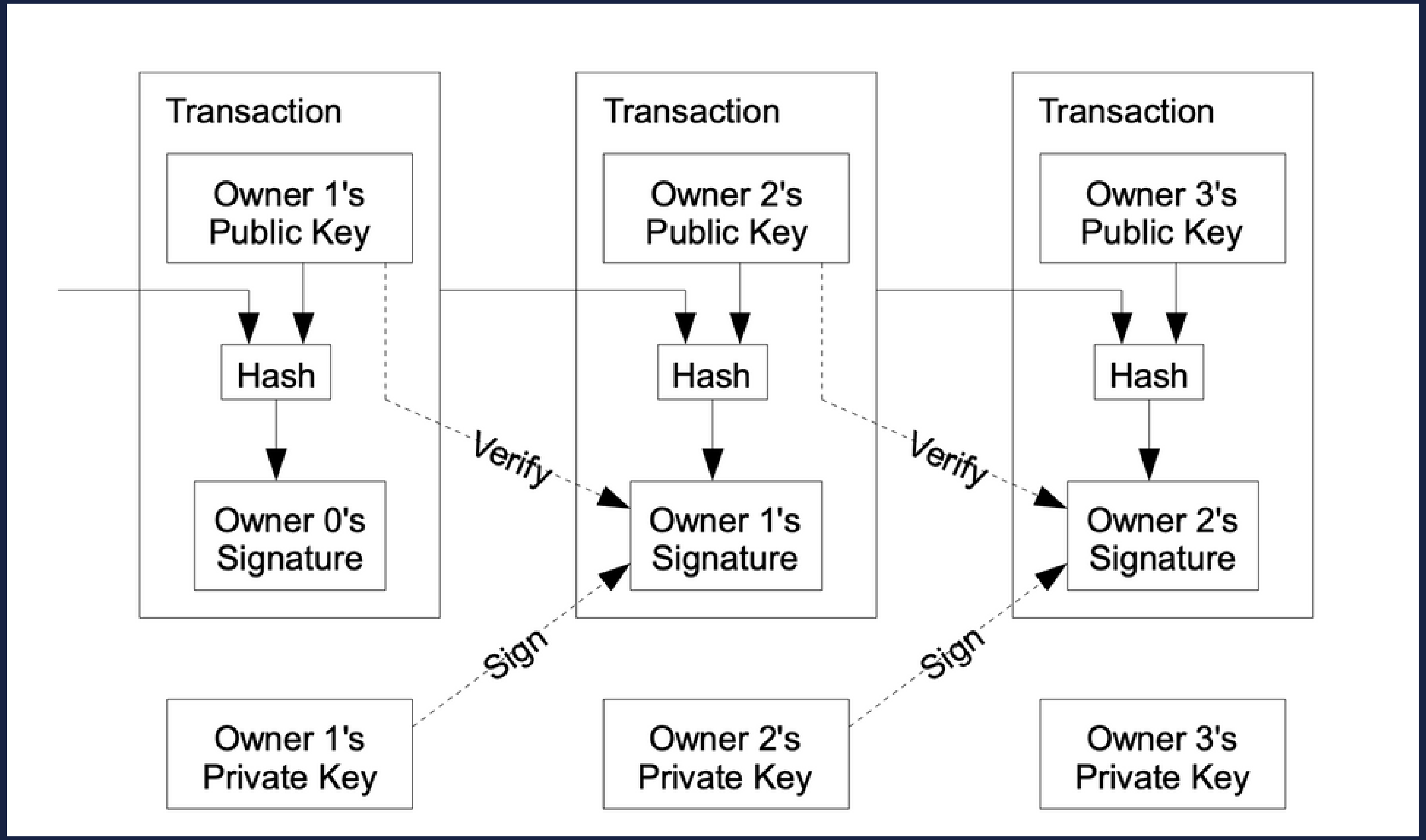
We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. 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], 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.

"各コインの所有者は、自分のトランザクションと次の所有者の公開鍵をハッシュ化したものに電子署名を行 い、これらを電子通貨の末尾に加えることによって、次の所有者にコインを転送する”

”受取人は一連の電子署名を検証することで、過去の所有権を確認できる”

金融機関等のTTPがトランザクションを検証しているため、既存金融システムでは二重支払い問題を解決できている。
トランザクションを公開し、ネットワーク参加者によって検証され合意形成を成すことで解決を試みる。

# 2章：トランザクション

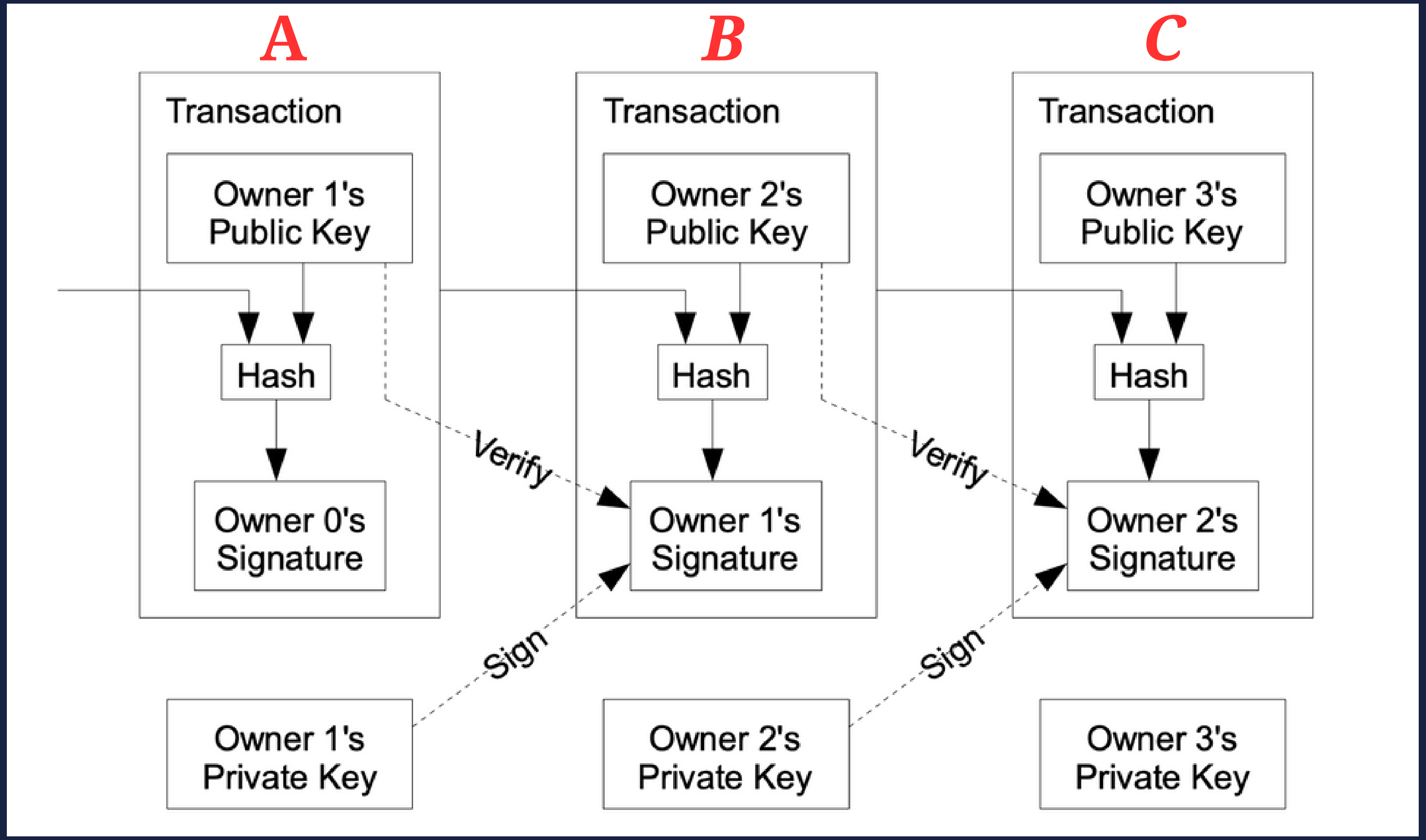


”本論文では、コインをチェーンのように連続したデジタル署名と定義する”

上記文章が左図に表されている。
”チェーン”という言葉がいわゆる『ブロックチェーン』の元になっていると思いがちだが、ここでいう”チェーン”はデジタル署名のチェーンである。

デジタル署名と類似した言葉『電子署名』
これらの違いは『電子署名』は文書の信頼性を保証する技術の総称で、デジタル署名は電子署名の中でも公開鍵暗号方式を用いたより高度な技術。
デジタル署名によって信頼性保証に加えて非改ざん性も保証できる。

# 2章：トランザクション



【簡易的な説明】

前提：Private KeyとPublic Keyは対になっている。

トランザクションAにはOwner1のPublicKeyがあり、

これはトランザクションBにあるOwner1の署名を

検証している。

*PrivateKey1*で偽の署名をした場合、*Owner1*の*PublicKey*

による検証で偽物だと判明した場合トランザクション*B*は

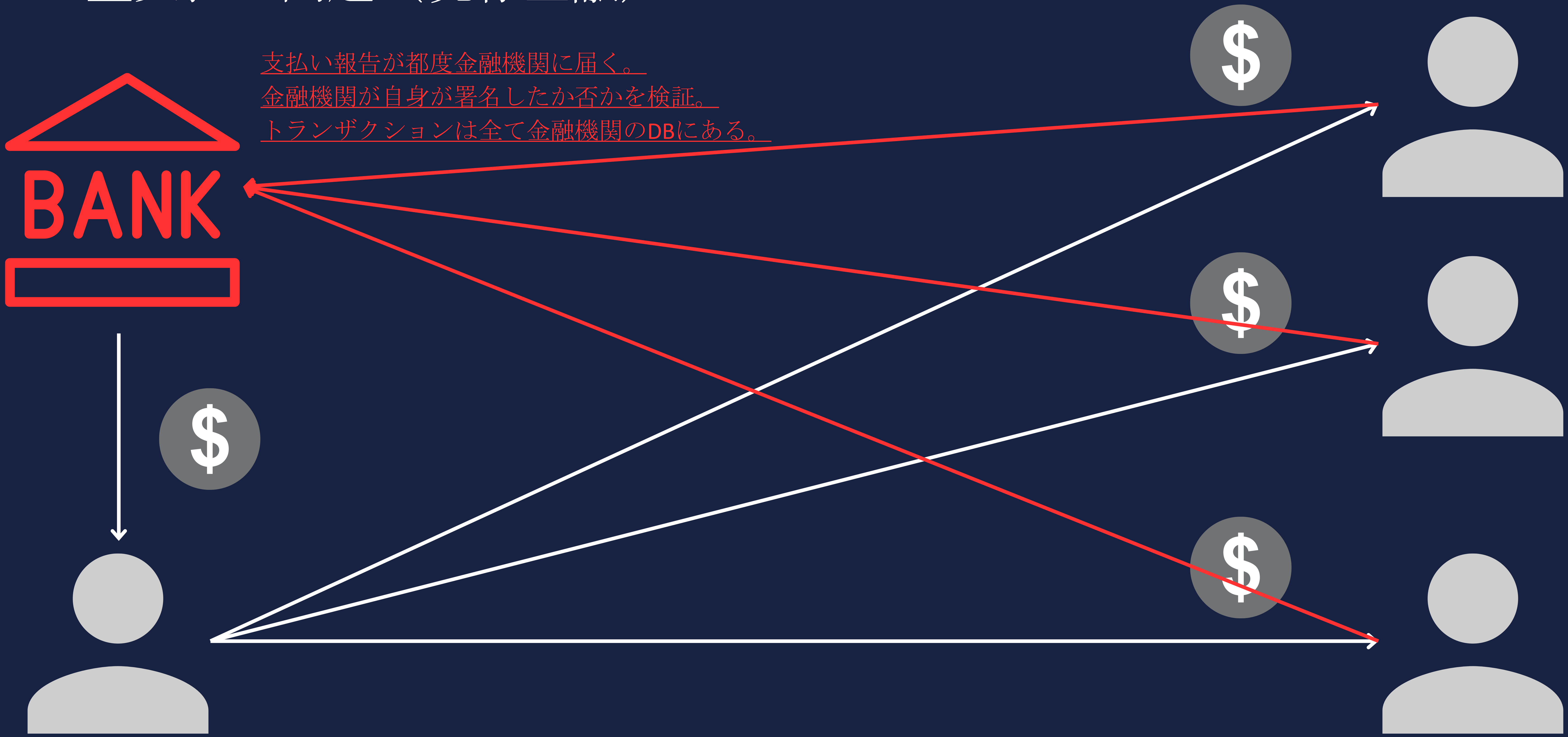
認められない。

このデジタル署名がチェーンとして繋がることによって

コインの譲渡が実現する。

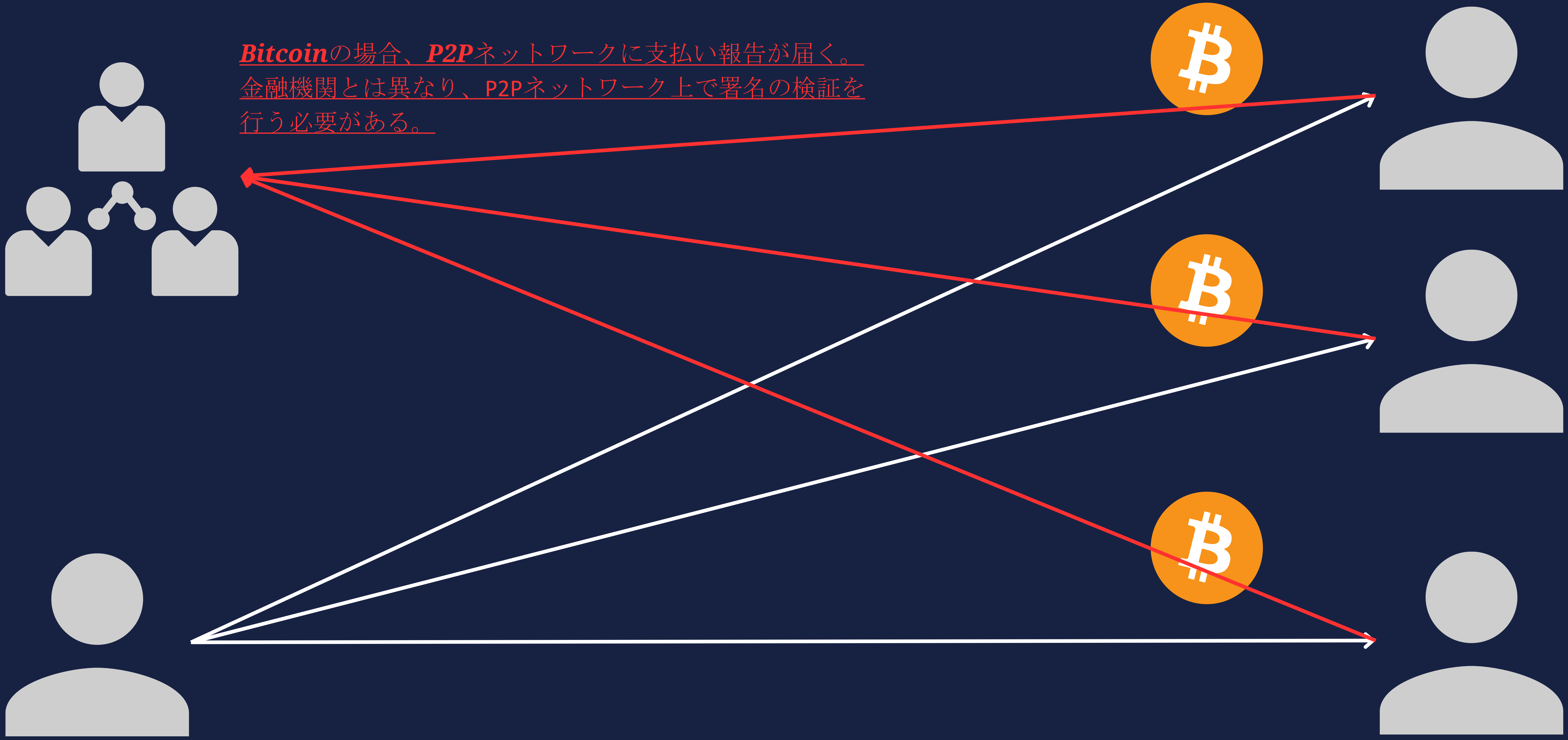
## 二重支払い問題（既存金融）

支払い報告が都度金融機関に届く。
金融機関が自身が署名したか否かを検証。
トランザクションは全て金融機関のDBにある。

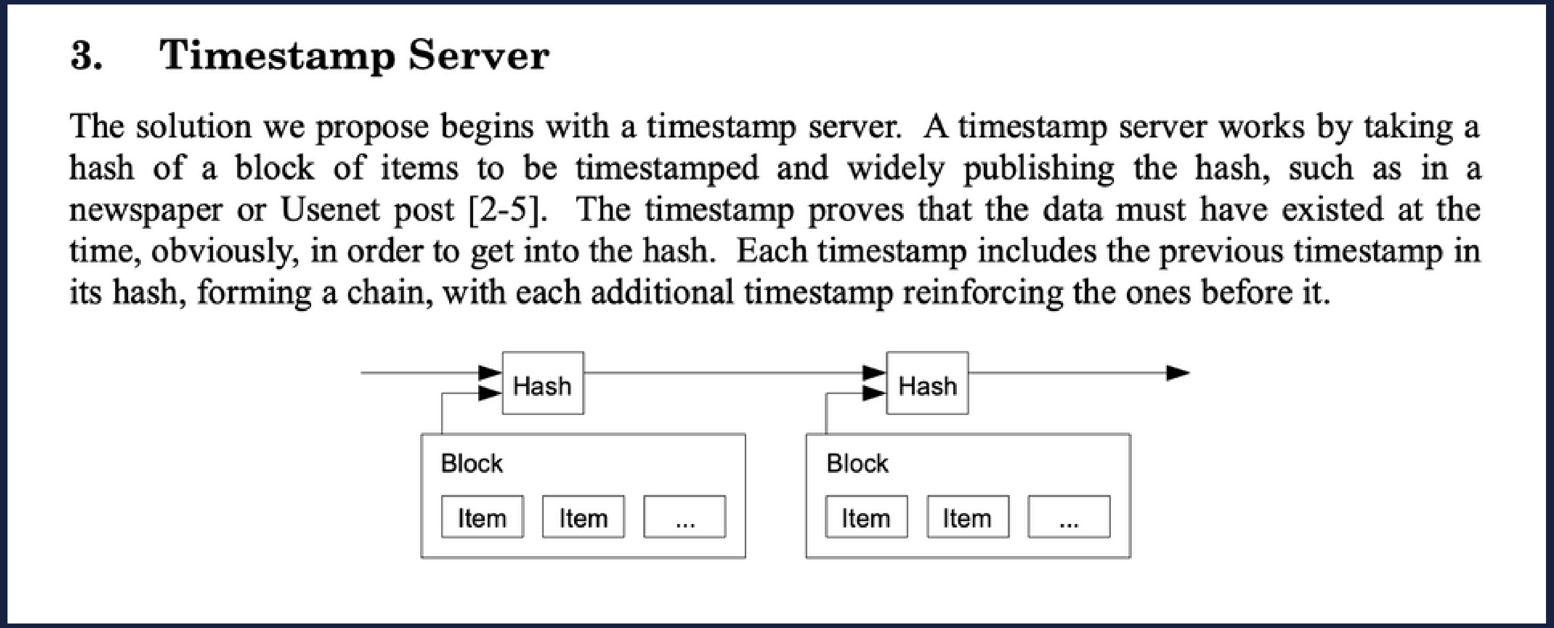


## 二重支払い問題（Bitcoin）

***Bitcoin***の場合、***P2P***ネットワークに支払い報告が届く。
金融機関とは異なり、P2Pネットワーク上で署名の検証を
行う必要がある。



# 3章：タイムスタンプサーバ



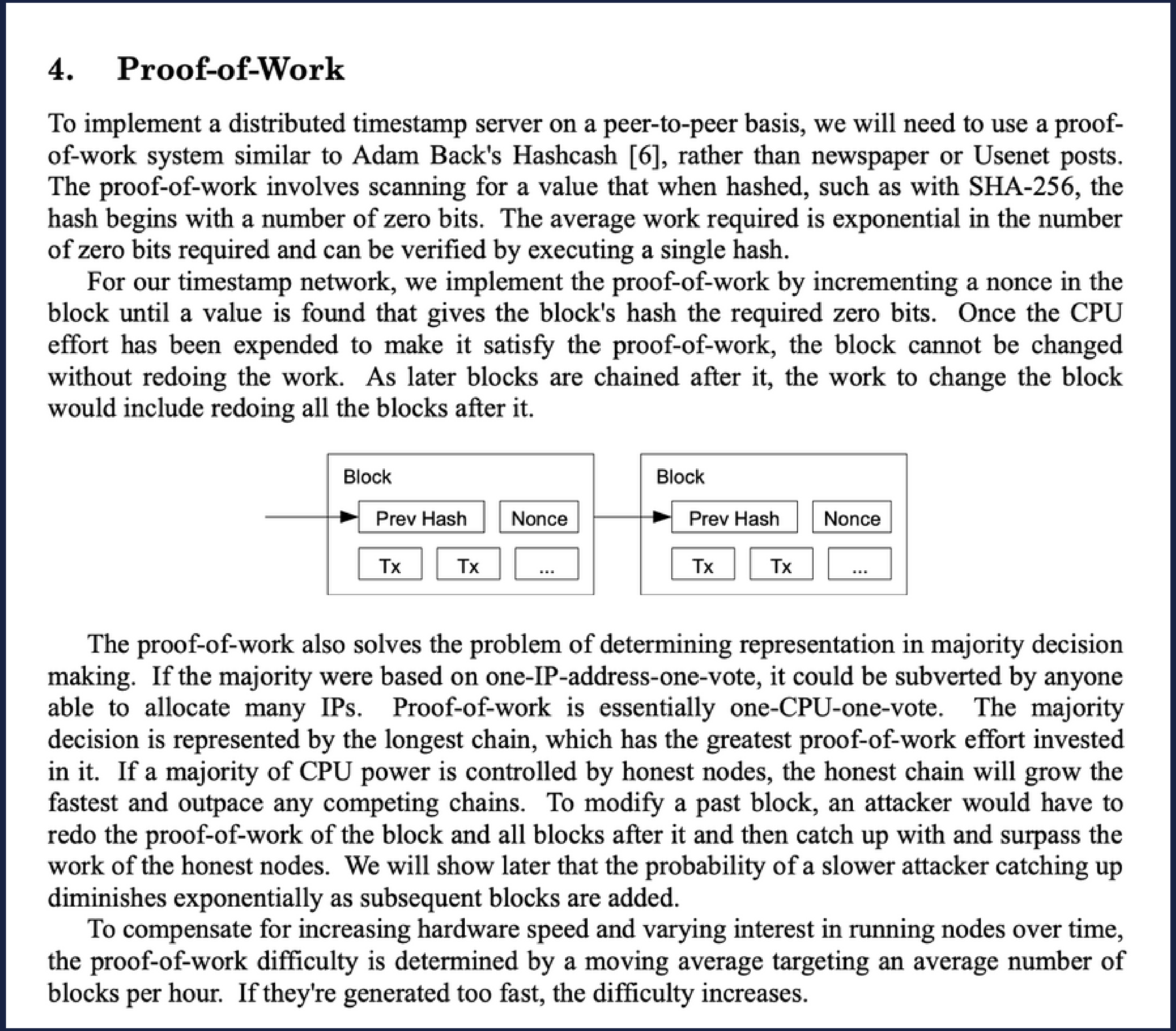
**3. Timestamp Server**

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.

”各タイムスタンプは前のタイムスタンプを含めてハッシュ化されてチェーンを形成し、それ以降にも各タイムスタンプが追加されることでそれ以前のタイムスタンプを強化している”

いわゆるLinkingのことで、Bitcoin以前からある技術。
*3*章では*Bitcoin*以前と同様のタイムスタンプの話をしており、*4*章以降で*TTP*無しのタイムスタンプについて説明されている。

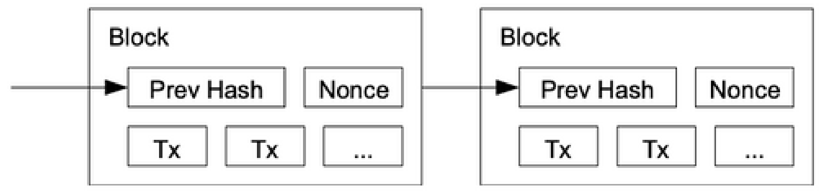
# *4章：Proof-of-Work*



**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], 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.



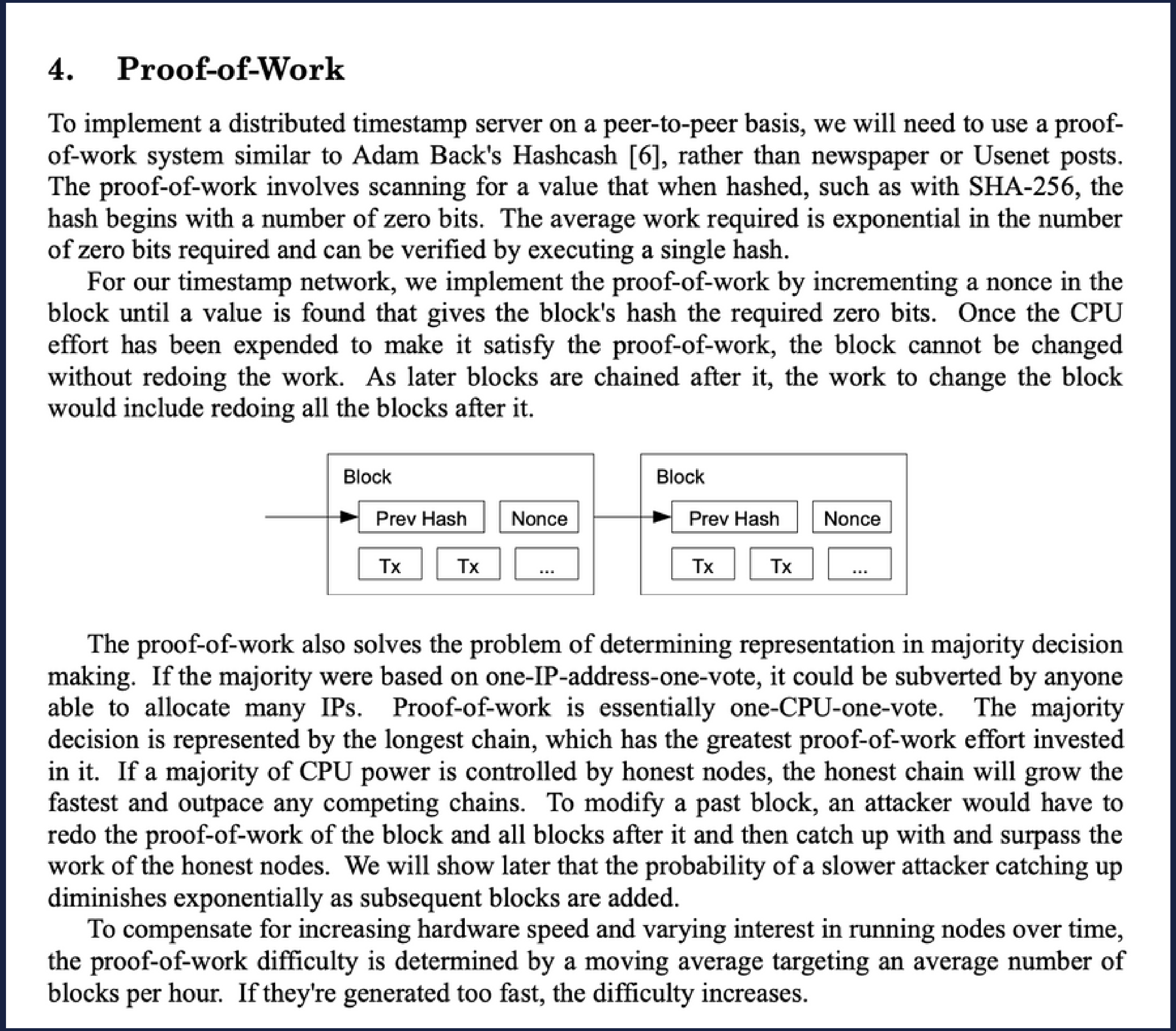
The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by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

*"P2P* を基盤とした分散型タイムスタンプサーバを実装するためには、新聞や*Usenet*の様なものではなく、*Adam Back*の*Hashcash*と同様の*Proof-of-Work*システムを使用する必要がある*"*

*Usenet*は分散型議論システムで、*1980*年に稼働を開始した。
*Bitcoin*の*Proof-of-Work*は*HashCash*を参考にしている。

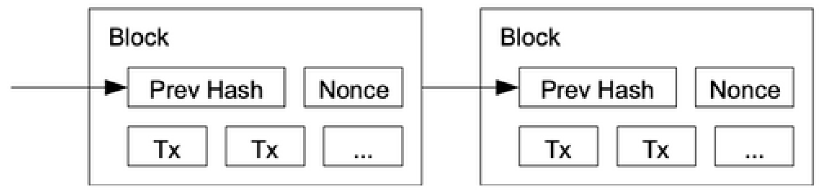
# *4章：Proof-of-Work*



**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], 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. 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.

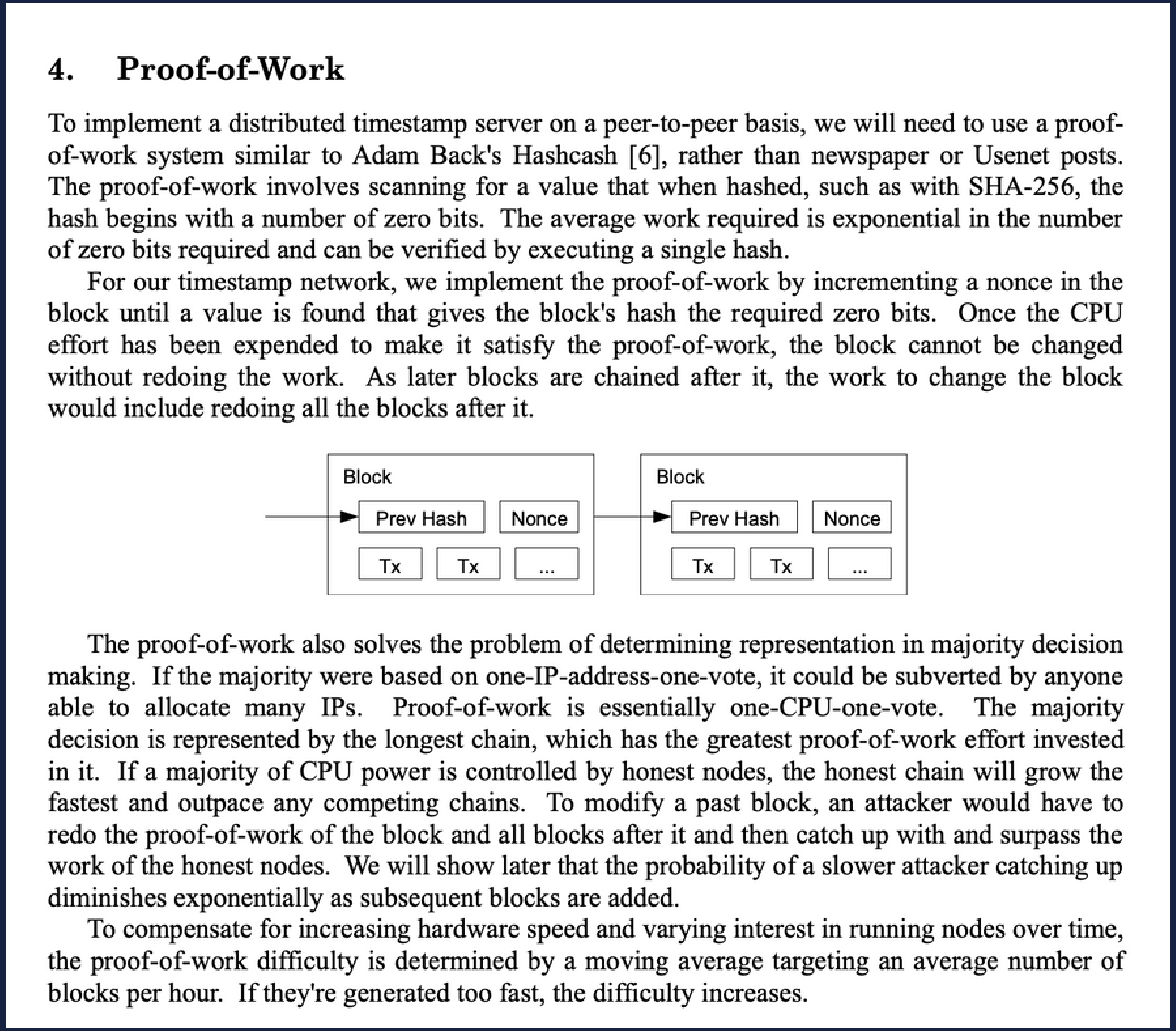
”*Proof-of-Work*の役割としては、数値が*SHA-256*等によりハッシュ化された際、最初の*n*ビットが全て*0*で始まる値を発見する事が挙げられる”

”要求される平均計算量は、必要な*0*のビット数に応じて指数関数的に増加する一方、検証については一つのハッシュ関数の計算により可能である”

原文をハッシュ関数に入れて最初のnビットが0となるものを見つけることはnが大きくなれば指数関数的に計算量が増加する一方で、その十分性を検証するには単に見つけた値をハッシュ関数に入れるだけで良い。

”本論文のタイムスタンプネットワークでは、ブロ ック内のハッシュに対して必要なゼロビットの値が見つかるまで、ブロック内のナンスの値を変化させ、*Proof-of-Work*を実行する”

# *4章：Proof-of-Work*



**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], 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. 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.

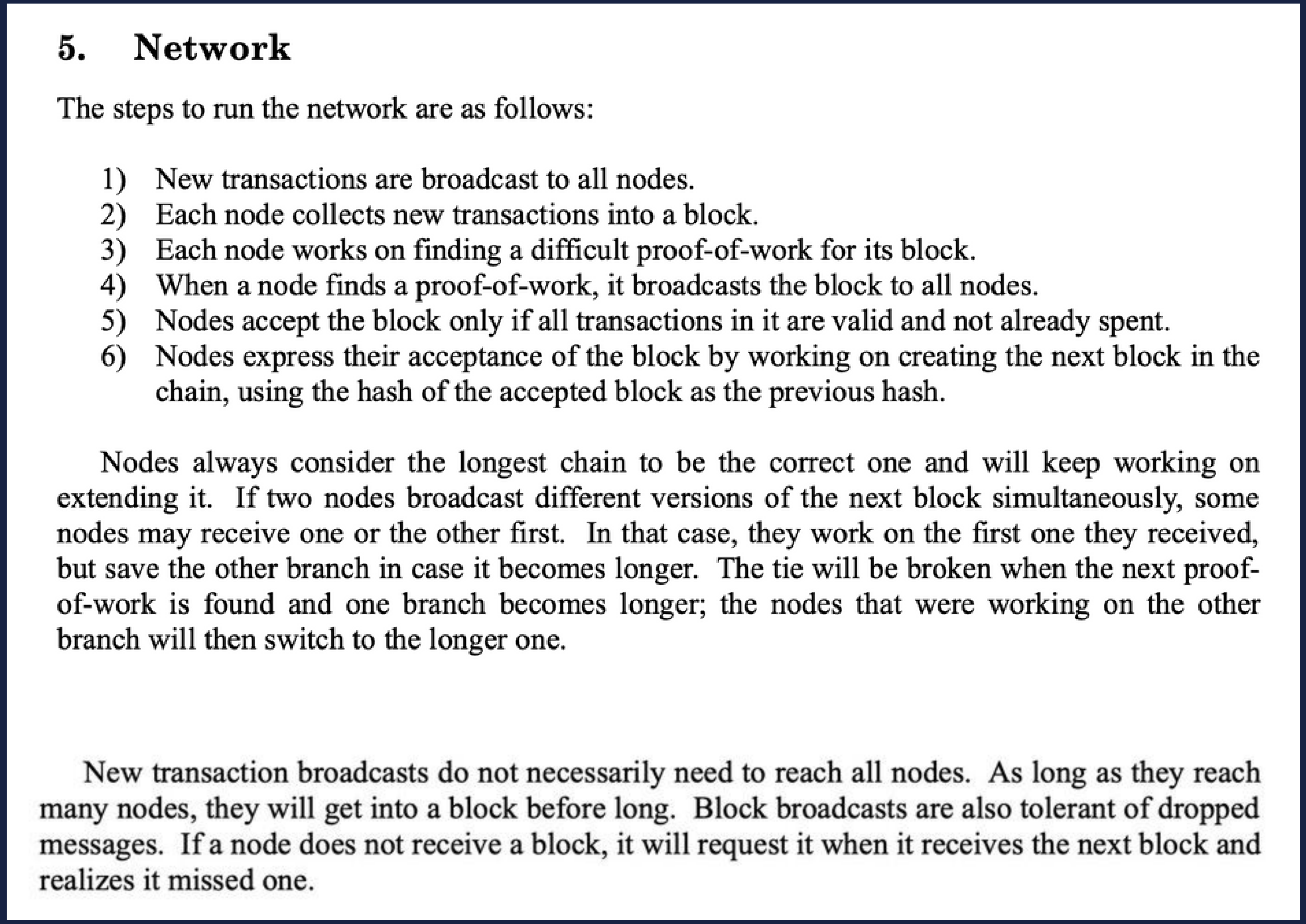
To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

*”Proof-of-Work* における本人確認は原則として*1CPU1*票である*”*

*”*また*Proof-of-Work*の計算量が最も使用されたチェ ーンが最長チェーンとなり、これが集団の意思決定における代表となる*”*

*”*年月の経過に伴うハードウェア能力の増加と実行中のノードの関心の変化に伴う影響を考慮し、*Proof-of-Work* の難易度は、*1*時間あたりの平均ブロック数を基にした移動平均により定められる*”*

# 5章：ネットワーク



**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.

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.

”ネットワークの作動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ノードに送信される

*2)* 各ノードが新しいトランザクションをあるブロックに取り入れる

*3)* 各ノードがそのブロックに対する *Proof-of-Work* を算出する

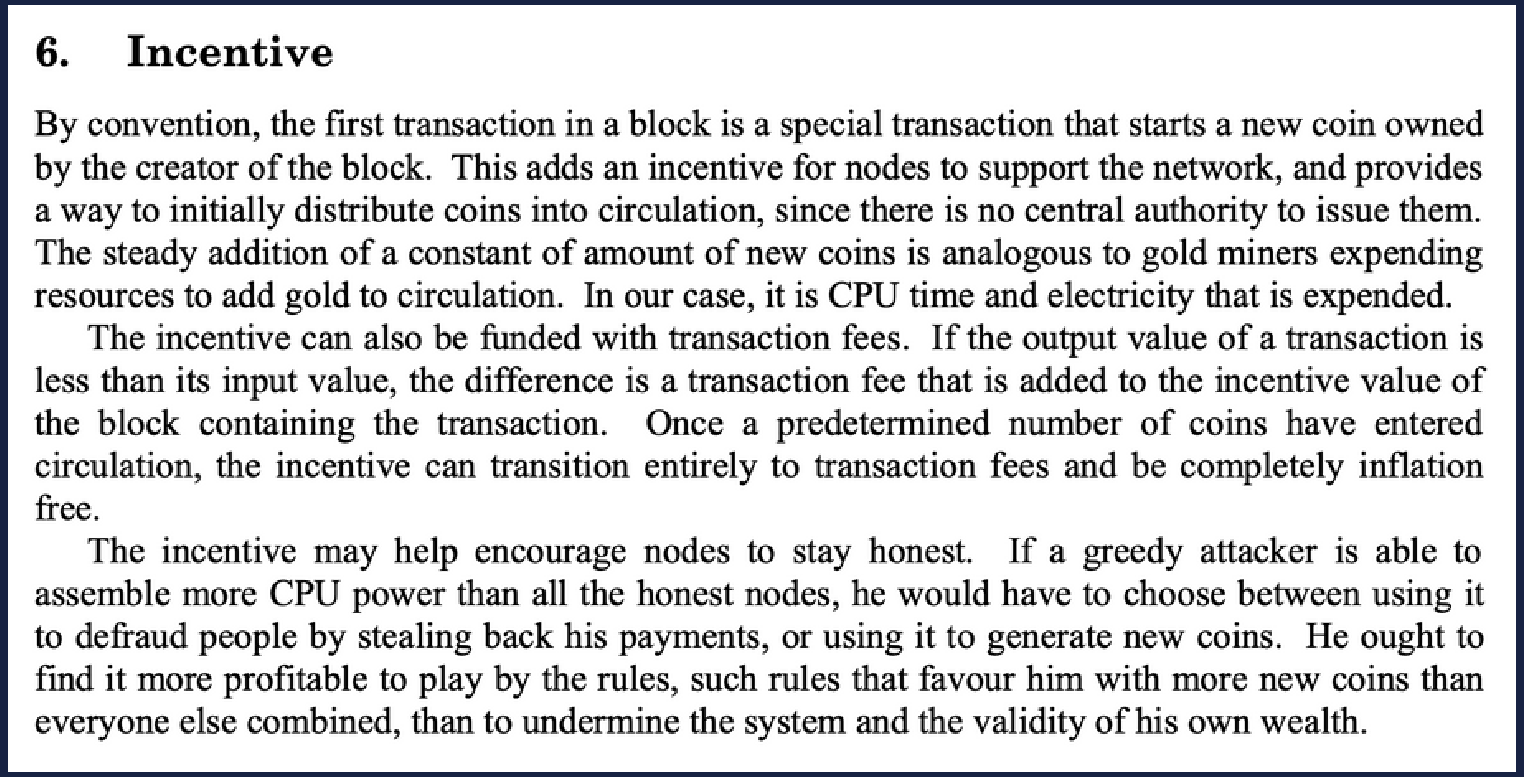
*4)* *Proof-of-Work* を見つけ次第、ノードはそのブロックを
全ノードにブロードキャスト する

*5)* 各ノードは、そのブロック内の全トランザクションが有効かつ
未使用の場合のみ承認を行う

*6)* 各ノードは、承認したブロックのハッシュを直前のハッシュ
として使用し、次のブロ ックの作成を開始することで、
ブロックの承認を表明する

ノードは常に最長チェーンを正しいものと判断し、延長を続ける”

# 6章：インセンティブ



**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.

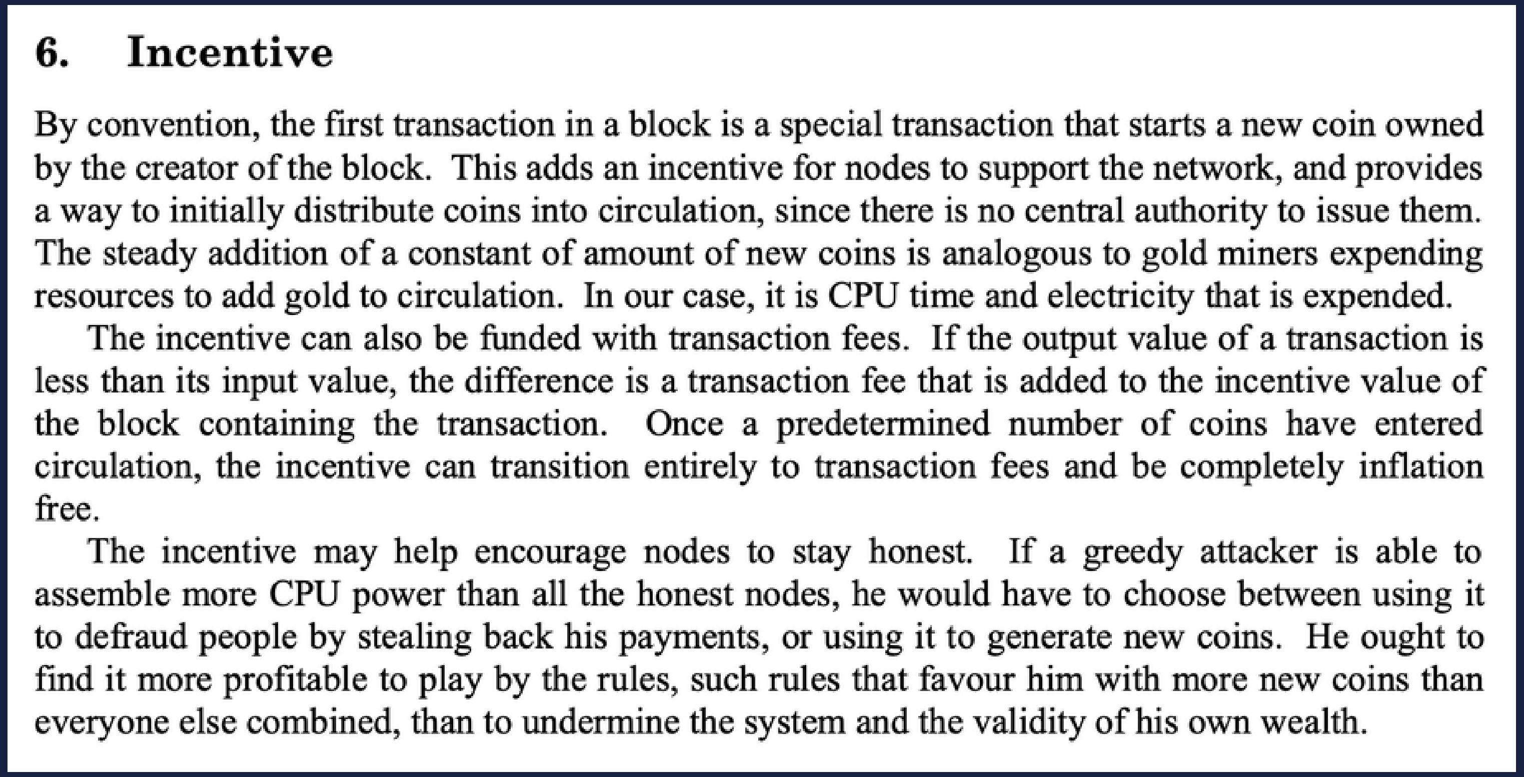
The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the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

”ブロック内の最初のトランザクションは、新しいコインを生成し始める上で特別なものであり、そのコインはブロック作成者のものとなる。
これは各ノードがネットワークを支持するインセンティブになると同時に、コインを発行する中央機関が不在の中、最初にコインを発行する方法としても機能する。
一定量の新しいコインを安定的に追加していく事は、金鉱労働者が採掘して金の流通量を増加させる事に類似している。本システムにおける採掘は、費やすCPU時間と電力である。”

金の新規採掘モデルに類似した仕組みを持ち、ノードがPoWを見つけることが新規採掘のタイミングとすることでDistributedなネットワーク構造を構築している。
ネットワークが存在し続けるためにはノードがPoWを見つけ続ける必要があるが、対価としてBitcoinの新規採掘権利を取得する。
このインセンティブ設計によって、BitcoinはTTPを必要としない持続可能性のあるDistributedネットワーク構造を実現している。

# 6章：インセンティブ



**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.

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.

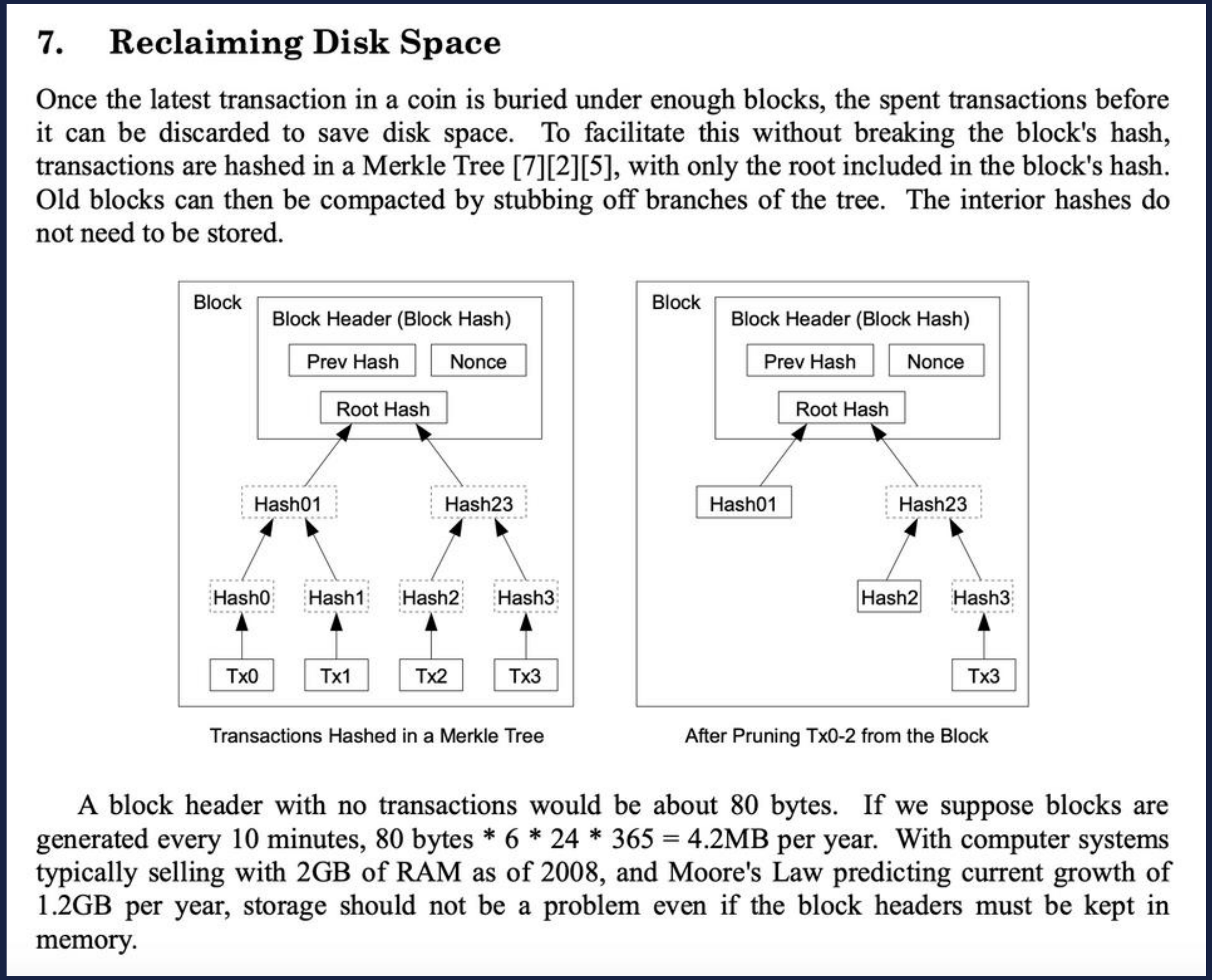
The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to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

”インセンティブは、トランザクション手数料によっても獲得できる。”

”インセンティブはノードが善意であり続ける動機となり得る”

→強欲で悪意のあるノードが半数以上の善意なノードのCPUパワーを
上回る状況下であるなら、半数以上のノードからコインを盗み取る
よりもルールに則ってPoWを見つけた方がインセンティブが
大きくなる設計。

# 7章：ディスクスペースの節約



### 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.

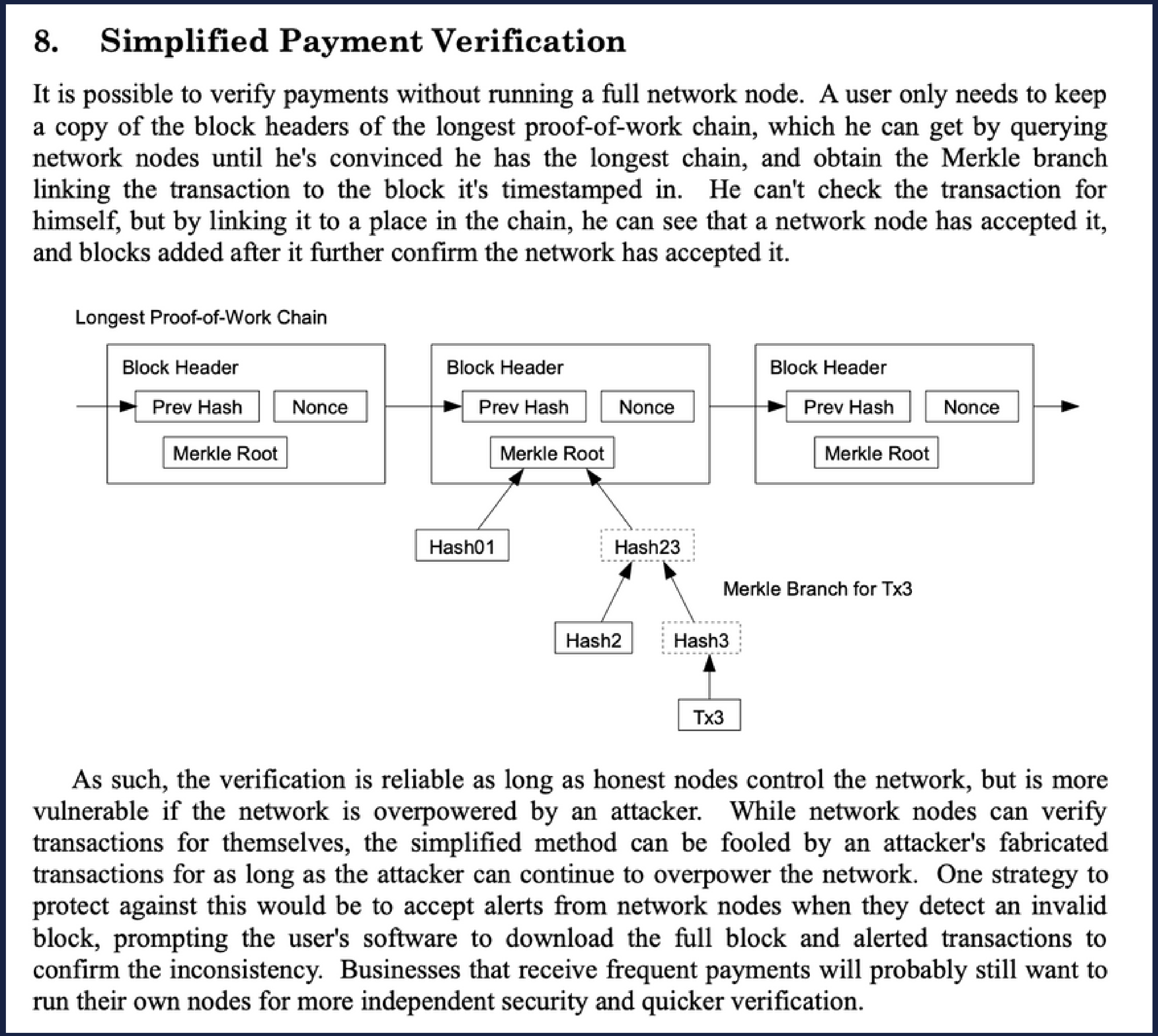
Transactions Hashed in a Merkle Tree

After Pruning Tx0-2 from the Block

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.

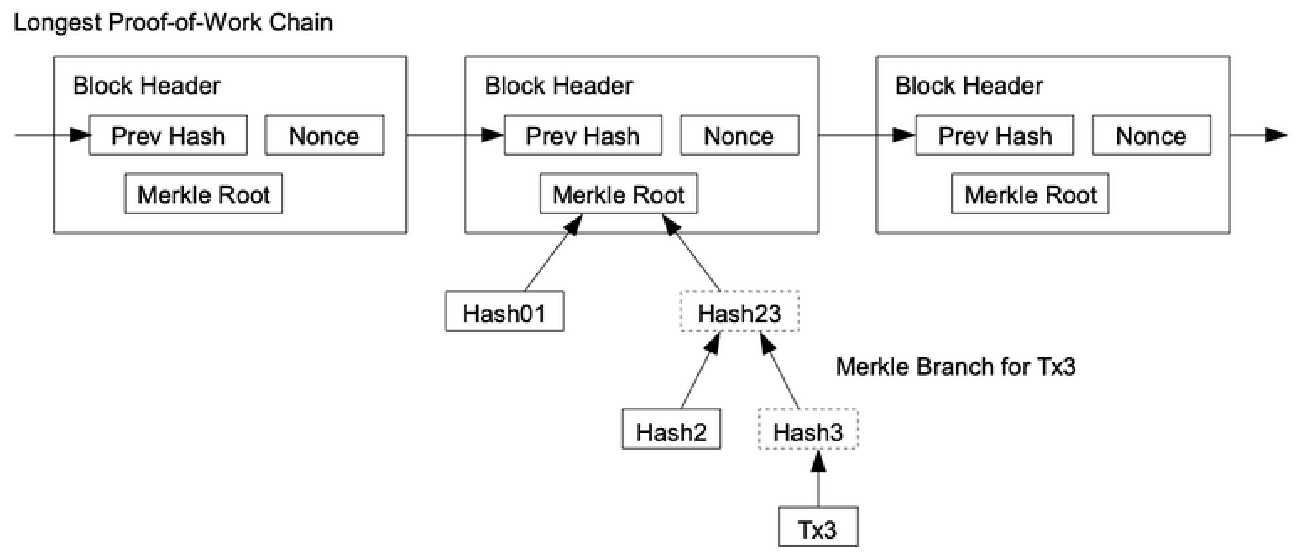
”コインの最新のトランザクションが十分な数のブロックに書き込まれると、それ以前のトランザクション記録はディスク・スペース節約のため破棄できる。
ブロックのハッシュを壊さずにこの作業を行うため、トランザクションはそのブロックにルートハッシュのみ含み、マークル・ツリーを用いてハッシュ化される。”
→古いブロックはブランチを削除してマークルハッシュのみを
保存すればよく、スペースの削減が実現される。

# 8章：トランザクションの簡易検証



**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, and blocks added after it further confirm the network has accepted it.



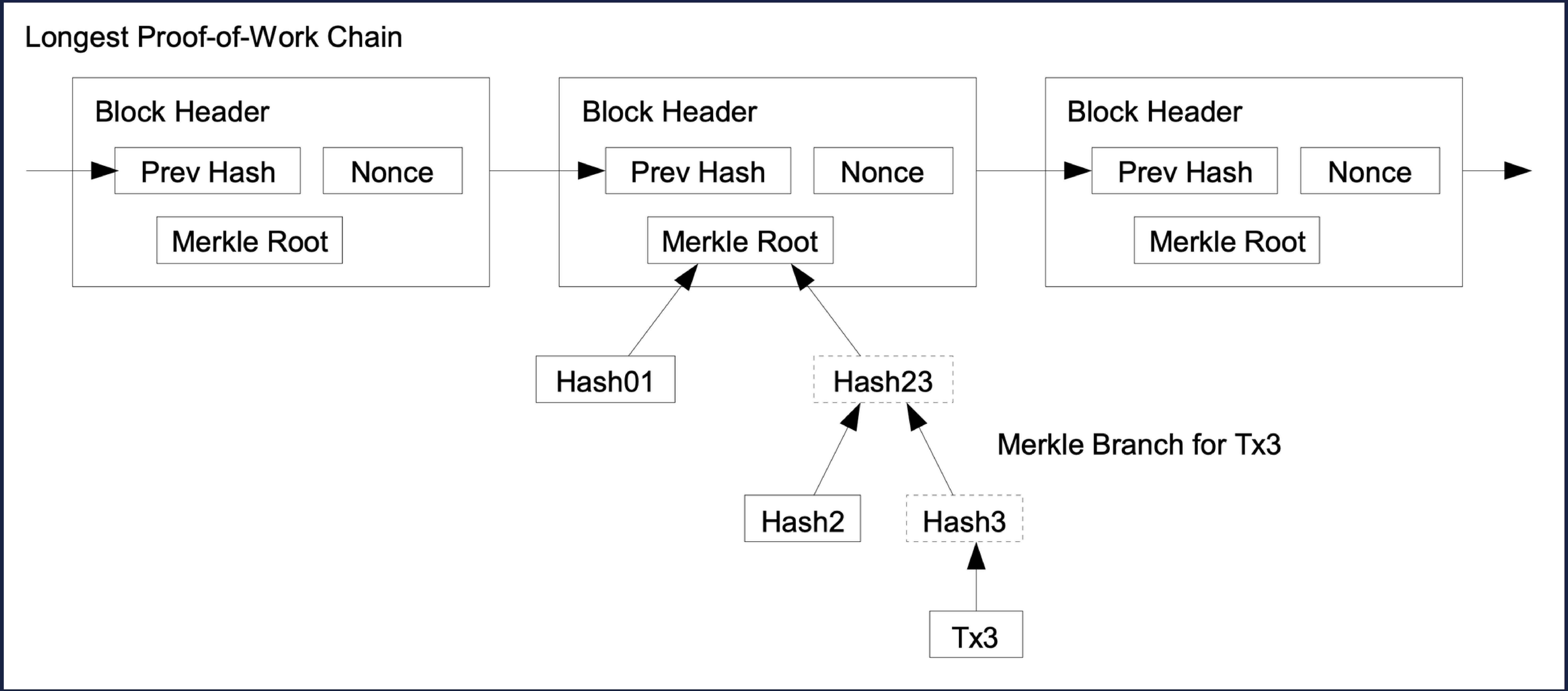
As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long 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.

”完全にネットワークノードを実行していなくとも、支払いの検証は可能である”

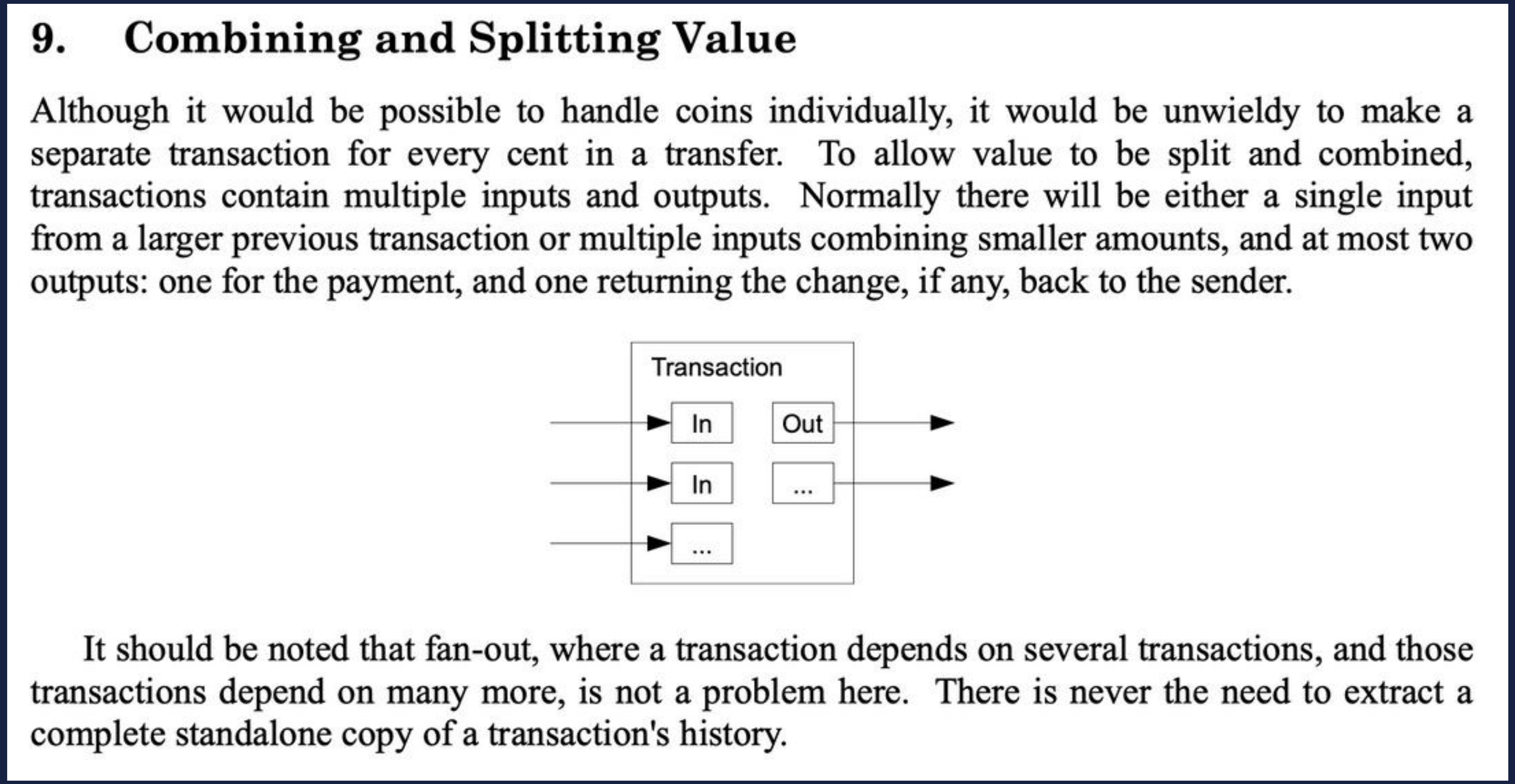
→これまではフルノードの話であったが、*8*章は軽量クライアントであってもトランザクションを簡易的に検証できるという話である。
　構造上ブロックヘッダーのみを保存しておけば、
　マークルブランチを参照することが可能である。

*Satoshi*は軽量クライアントについて述べただけであり、実装はしていない。
いわゆるWalletが軽量クライアントとなる。

## 8章：トランザクションの簡易検証



# 9章：価値の結合および分割



9. **Combining and Splitting Value**

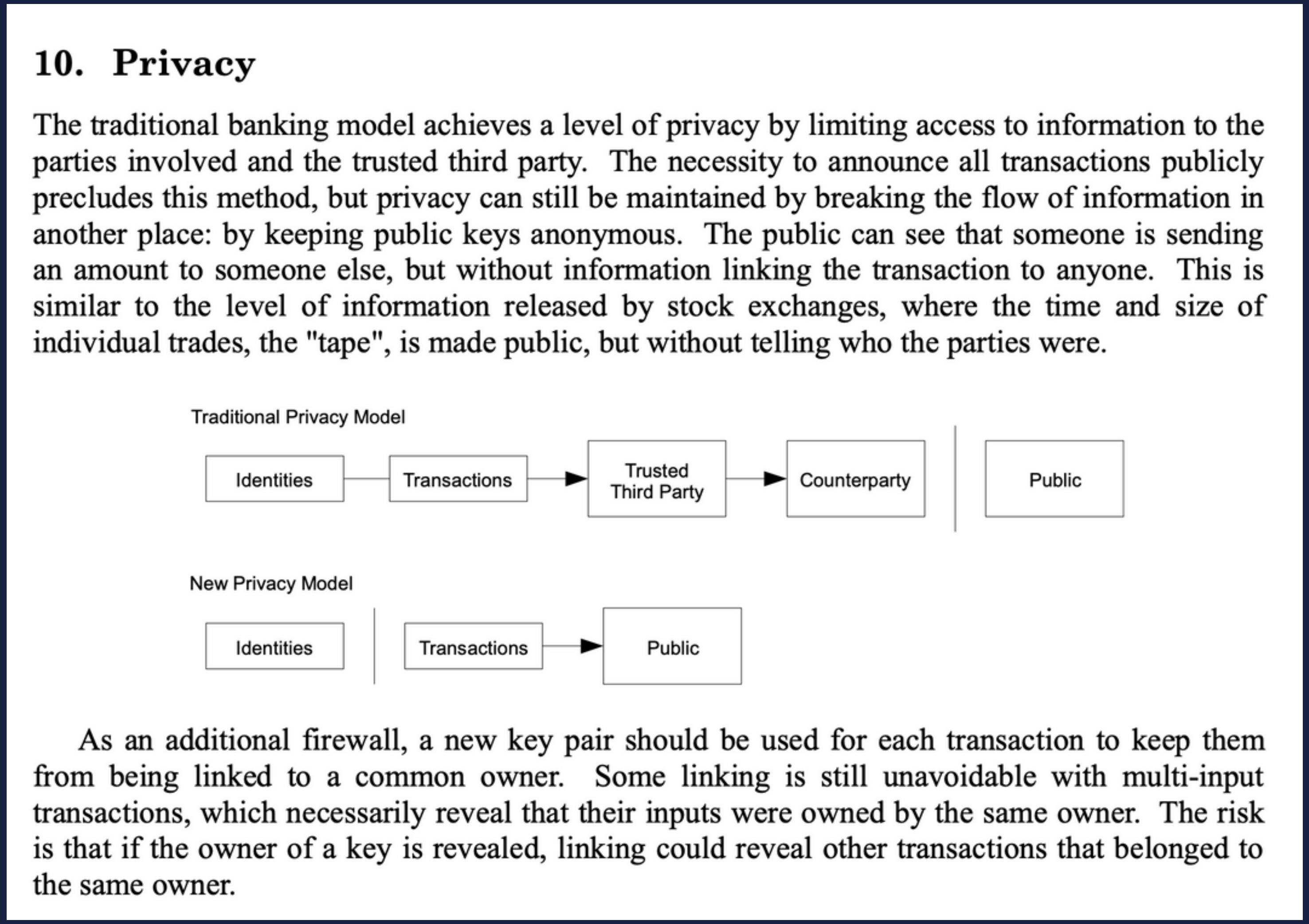
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous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

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.

”コインを個別に扱う事も可能であるが、トランザクション金額を*1*セントずつ個別に扱うのは不便だろう。
価値の分割や結合を可能にするため、トランザクションには複数のインプットとアウトプットが含まれる。
通常インプットは価値のより大きな前トランザクションからの1つのものか、小額のものを組み合わせた複数のものに分別される。
一方、アウトプットは支払い目的のものと、もし釣銭があればそれを支払い元に返還するものに分別される。”

Bitcoinは小数点以下8桁までで表すことが可能。
これはトランザクションの構造によるものであり、図にある通りInput（譲渡前のコイン情報）とOutput（譲渡後のコイン情報）が存在するためである。

# *10*章：プライバシー



## 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. 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.

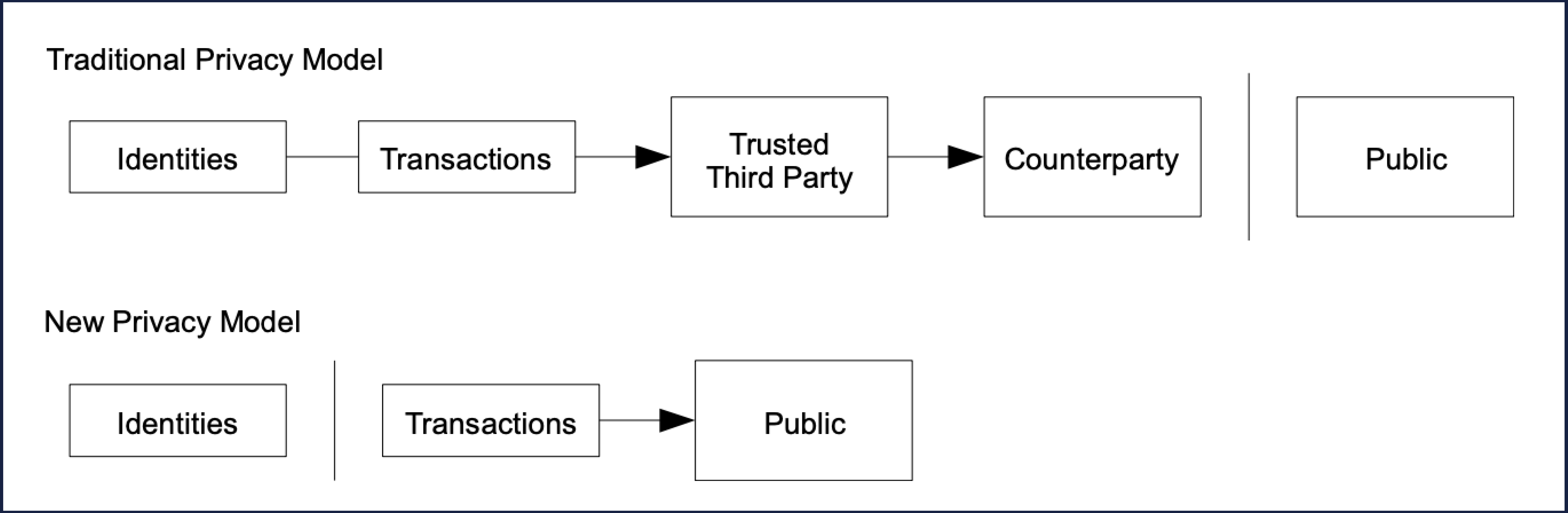
As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a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

”伝統的な銀行モデルでは、情報へのアクセスを関連団体と信頼の置ける第三者機関に限定する事で、一定レベルのプライバシーを実現している。”

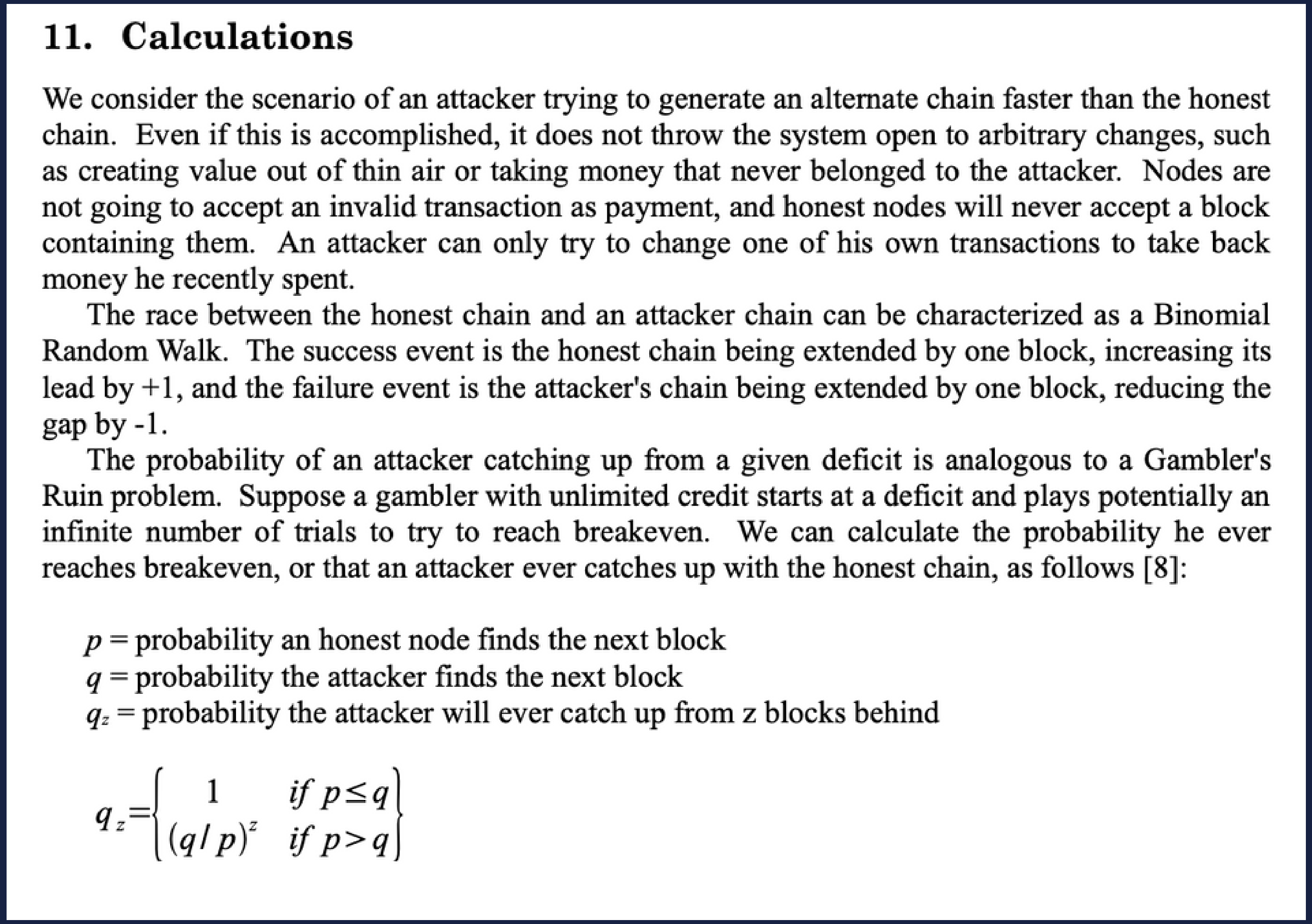
”誰かが誰かにどれだけのコインを送付したかは公開されるが、そのトランザクション情報 は誰にもリンクされていない。これは証券取引で公表されるものと同等の情報レベルであり、個別のトランザクションの時間、数量およびティッカーシンボルは公開されたとしても、 そのトランザクションの当事者自体は公開されないのである。”

アイデンティティとブロックチェーン上のアドレスが紐づくことはなく、それによってプライバシーを高めている。
Public Blockchainである以上、どのアドレスからどのアドレスへと資金がどれくらい流入しているかは公開されているが、この構造によってどの個人もBitcoinによる個人情報漏洩を受けることがない。
（TTPが一つであっても介在すると、この定説は覆る。TTPの具体例は取引所やクレジットカード等でのOn-rampなど。）

## 10章：プライバシー


Traditional Privacy Model
Identities
Transactions
Trusted Third Party
Counterparty
Public
New Privacy Model
Identities
Transactions
Public

# *11*章：数学的根拠



**11. 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 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]:

$p$ = probability an honest node finds the next block
$q$ = probability the attacker finds the next block
$q_z$ = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{Bmatrix} 1 & \text{if } p \le q \\ (q/p)^z & \text{if } p > q \end{Bmatrix}$$

”攻撃者が善意のチェーンより速いスピードで偽のチェーンを生成しようと試みるシナリオを考察する。
仮にそれが成功したとしても、コインを無から生成したり、攻撃者自身が所有した事のないコインを取得したり、といったようにシステムを自由に変更できるわけではない。
各ノードは無効なトランザクションやそのトランザクションを含んだブロックを拒絶するのである。
攻撃者は自身のトランザクション記録を書き換える事で、最近の支払金 額を取り返そうとする事のみが可能である。”

*Whitepaper*では以下、攻撃者が二重支払いを試みている状況下でシミュレーションを行っている。