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

**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.

**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 nonreversible 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.

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.

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.

4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proofof-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.

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 proofof-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

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.

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.

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**

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.

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.

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.

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

qz = probability the attacker will ever catch up from z blocks behind

qz={ 1 if p≤q

q/ pz if pq}

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.

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:

=z q

p

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:

∑k=0

∞ k e−

k! ⋅{q/ p z−k if k≤ z

1 if k z}

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

1−∑

k=0

z

k e−

k! 1−q/ p z−k

Converting to C code…

#include <math.h>

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

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. 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

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal

trust requirements," In 20th Symposium on Information Theory in the Benelux, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, vol 3, no

2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping,"

In Sequences II: Methods in Communication, Security and Computer Science, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4th ACM Conference

on Computer and Communications Security, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure,"

http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and

Privacy, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.

**Bitcoin: Hệ thống tiền mặt điện tử phi tập trung**

**Tóm tắt.** Một phiên bản tiền mặt điện tử hoàn toàn ngang hàng sẽ cho phép gửi thanh toán trực tuyến trực tiếp từ bên này sang bên kia mà không cần thông qua một tổ chức tài chính. Chữ ký số cung cấp một phần giải pháp, nhưng các lợi ích chính sẽ bị mất nếu vẫn cần một bên thứ ba đáng tin cậy để ngăn chặn việc chi tiêu kép. Chúng tôi đề xuất một giải pháp cho vấn đề chi tiêu kép bằng cách sử dụng mạng ngang hàng. Mạng này ghi dấu thời gian cho các giao dịch bằng cách băm chúng vào một chuỗi liên tục của chứng minh công việc dựa trên băm, tạo thành một bản ghi không thể thay đổi mà không cần làm lại chứng minh công việc. Chuỗi dài nhất không chỉ đóng vai trò là bằng chứng về trình tự các sự kiện được chứng kiến, mà còn là bằng chứng cho thấy nó đến từ nhóm sức mạnh CPU lớn nhất. Miễn là phần lớn sức mạnh CPU được kiểm soát bởi các nút không hợp tác để tấn công mạng, chúng sẽ tạo ra chuỗi dài nhất và vượt qua kẻ tấn công. Bản thân mạng lưới yêu cầu cấu trúc tối thiểu. Thông điệp được phát sóng trên cơ sở nỗ lực tốt nhất, và các nút có thể rời khỏi và tham gia lại mạng lưới bất cứ lúc nào, chấp nhận chuỗi chứng minh công việc dài nhất là bằng chứng cho những gì đã xảy ra trong thời gian chúng vắng mặt.

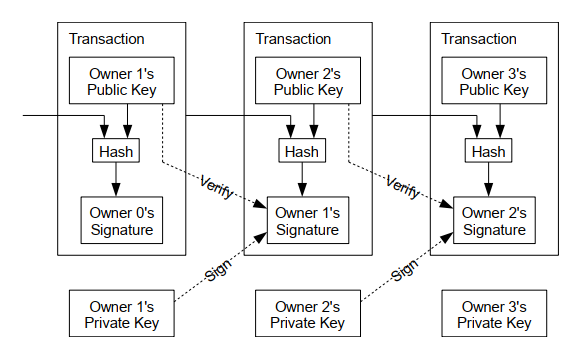
**1. Giới thiệu**

Thương mại trên Internet hầu như hoàn toàn phụ thuộc vào các tổ chức tài chính đóng vai trò là bên thứ ba đáng tin cậy để xử lý các khoản thanh toán điện tử. Mặc dù hệ thống hoạt động khá tốt đối với hầu hết các giao dịch, nhưng nó vẫn gặp phải những điểm yếu cố hữu của mô hình dựa trên sự tin tưởng. Các giao dịch hoàn toàn không thể đảo ngược thực sự không khả thi, vì các tổ chức tài chính không thể tránh được việc hòa giải tranh chấp. Chi phí hòa giải làm tăng chi phí giao dịch, hạn chế quy mô giao dịch tối thiểu thực tế và cắt đứt khả năng thực hiện các giao dịch nhỏ lẻ, tình cờ, và có một chi phí rộng hơn trong việc mất khả năng thực hiện các khoản thanh toán không thể đảo ngược cho các dịch vụ không thể đảo ngược. Với khả năng đảo ngược, nhu cầu về sự tin tưởng lan rộng. Người bán phải thận trọng với khách hàng của họ, làm phiền họ để lấy thêm thông tin hơn mức cần thiết. Một tỷ lệ phần trăm nhất định của gian lận được chấp nhận là không thể tránh khỏi. Những chi phí và sự không chắc chắn về thanh toán này có thể được tránh bằng cách sử dụng tiền mặt vật lý khi giao dịch trực tiếp, nhưng không có cơ chế nào tồn tại để thực hiện thanh toán qua kênh truyền thông mà không cần một bên đáng tin cậy.

Điều cần thiết là một hệ thống thanh toán điện tử dựa trên chứng minh mật mã thay vì sự tin tưởng, cho phép bất kỳ hai bên nào sẵn sàng giao dịch trực tiếp với nhau mà không cần bên thứ ba đáng tin cậy. Các giao dịch về mặt tính toán là không thể đảo ngược sẽ bảo vệ người bán khỏi gian lận, và các cơ chế ký quỹ thông thường có thể dễ dàng được triển khai để bảo vệ người mua. Trong bài báo này, chúng tôi đề xuất một giải pháp cho vấn đề chi tiêu kép bằng cách sử dụng máy chủ thời gian phân tán ngang hàng để tạo ra chứng minh tính toán về thứ tự thời gian của các giao dịch. Hệ thống an toàn miễn là các nút trung thực cùng kiểm soát nhiều sức mạnh CPU hơn bất kỳ nhóm kẻ tấn công hợp tác nào.

**2. Giao dịch**

Chúng tôi định nghĩa một đồng tiền điện tử là một chuỗi chữ ký số. Mỗi chủ sở hữu chuyển đồng tiền cho người tiếp theo bằng cách ký số một băm của giao dịch trước đó và khóa công khai của chủ sở hữu tiếp theo, và thêm chúng vào cuối đồng tiền. Người nhận có thể xác minh các chữ ký để xác minh chuỗi sở hữu.



Vấn đề tất nhiên là người nhận không thể xác minh rằng một trong những chủ sở hữu đã không chi tiêu kép đồng tiền. Một giải pháp phổ biến là giới thiệu một cơ quan trung ương đáng tin cậy, hoặc cơ sở đúc tiền, kiểm tra mọi giao dịch để phát hiện chi tiêu kép. Sau mỗi giao dịch, đồng tiền phải được trả lại cho cơ sở đúc để phát hành một đồng tiền mới, và chỉ những đồng tiền được phát hành trực tiếp từ cơ sở đúc mới được tin tưởng là không bị chi tiêu kép. Vấn đề với giải pháp này là số phận của toàn bộ hệ thống tiền tệ phụ thuộc vào công ty điều hành cơ sở đúc, với mọi giao dịch phải thông qua họ, giống như một ngân hàng.

Chúng ta cần một cách để người nhận biết rằng các chủ sở hữu trước đó đã không ký bất kỳ giao dịch nào trước đó. Đối với mục đích của chúng ta, giao dịch sớm nhất là giao dịch được tính, vì vậy chúng ta không quan tâm đến các nỗ lực chi tiêu kép sau này. Cách duy nhất để xác nhận sự vắng mặt của một giao dịch là phải biết tất cả các giao dịch. Trong mô hình dựa trên cơ sở đúc tiền, cơ sở đúc biết tất cả các giao dịch và quyết định giao dịch nào đến trước. Để thực hiện điều này mà không cần một bên đáng tin cậy, các giao dịch phải được công bố công khai [1], và chúng ta cần một hệ thống để những người tham gia đồng ý về một lịch sử duy nhất về thứ tự mà chúng được nhận. Người nhận cần bằng chứng rằng tại thời điểm mỗi giao dịch, đa số các nút đã đồng ý rằng đó là giao dịch được nhận đầu tiên.

**3. Máy chủ ghi dấu thời gian**

Giải pháp chúng tôi đề xuất bắt đầu với một máy chủ ghi dấu thời gian. Một máy chủ ghi dấu thời gian hoạt động bằng cách lấy một băm của một khối các mục cần được ghi dấu thời gian và công bố rộng rãi băm đó, chẳng hạn như trên báo hoặc bài đăng Usenet [2-5]. Dấu thời gian chứng minh rằng dữ liệu chắc chắn đã tồn tại tại thời điểm đó, rõ ràng là để có thể đưa vào băm. Mỗi dấu thời gian bao gồm dấu thời gian trước đó trong băm của nó, tạo thành một chuỗi, với mỗi dấu thời gian bổ sung củng cố những dấu thời gian trước đó.

**4. Chứng minh công việc (Proof-of-Work)**

Để triển khai một máy chủ ghi dấu thời gian phân tán trên cơ sở ngang hàng, chúng ta sẽ cần sử dụng một hệ thống chứng minh công việc tương tự như Hashcash của Adam Back [6], thay vì bài đăng trên báo hoặc Usenet.

Chứng minh công việc liên quan đến việc tìm kiếm một giá trị mà khi được băm, chẳng hạn như với SHA-256, băm bắt đầu bằng một số bit bằng 0. Công việc trung bình cần thiết tăng theo cấp số mũ với số lượng bit bằng 0 cần thiết và có thể được xác minh bằng cách thực hiện một băm duy nhất.

Đối với mạng lưới ghi dấu thời gian của chúng ta, chúng ta triển khai chứng minh công việc bằng cách tăng một nonce trong khối cho đến khi tìm thấy một giá trị tạo ra băm của khối có số lượng bit bằng 0 cần thiết. Sau khi nỗ lực tính toán đã được bỏ ra để làm cho nó thỏa mãn chứng minh công việc, khối không thể được thay đổi mà không cần làm lại công việc. Vì các khối sau đó được nối chuỗi sau nó, nên công việc để thay đổi khối sẽ bao gồm làm lại tất cả các khối sau nó.

A diagram of a block diagram

Description automatically generated

Chứng minh công việc cũng giải quyết vấn đề xác định sự đại diện trong việc ra quyết định đa số. Nếu đa số dựa trên một-địa-chỉ-IP-một-phiếu, nó có thể bị phá hoại bởi bất kỳ ai có khả năng phân bổ nhiều IP. Chứng minh công việc về cơ bản là một-CPU-một-phiếu. Quyết định đa số được thể hiện bằng chuỗi dài nhất, chuỗi có lượng công việc chứng minh lớn nhất được đầu tư vào nó. Nếu đa số sức mạnh CPU được kiểm soát bởi các nút trung thực, chuỗi trung thực sẽ phát triển nhanh nhất và vượt qua bất kỳ chuỗi cạnh tranh nào. Để sửa đổi một khối trong quá khứ, kẻ tấn công sẽ phải làm lại chứng minh công việc của khối đó và tất cả các khối sau đó và sau đó bắt kịp và vượt qua công việc của các nút trung thực. Chúng tôi sẽ chứng minh sau rằng xác suất một kẻ tấn công chậm hơn bắt kịp giảm theo cấp số mũ khi các khối tiếp theo được thêm vào.

Để bù đắp cho tốc độ phần cứng ngày càng tăng và sự quan tâm khác nhau đến việc chạy các nút theo thời gian, độ khó của chứng minh công việc được xác định bởi một trung bình di động nhắm mục tiêu đến một số lượng khối trung bình mỗi giờ. Nếu chúng được tạo ra quá nhanh, độ khó sẽ tăng lên.

5. Mạng lưới

Các bước để chạy mạng lưới như sau:

Các giao dịch mới được phát sóng đến tất cả các nút.

Mỗi nút thu thập các giao dịch mới vào một khối.

Mỗi nút làm việc để tìm một chứng minh công việc khó khăn cho khối của nó.

Khi một nút tìm thấy một chứng minh công việc, nó phát sóng khối đó đến tất cả các nút.

Các nút chỉ chấp nhận khối nếu tất cả các giao dịch trong đó đều hợp lệ và chưa được chi tiêu.

Các nút thể hiện sự chấp nhận khối bằng cách làm việc để tạo khối tiếp theo trong chuỗi, sử dụng băm của khối được chấp nhận làm băm trước đó.

Các nút luôn xem chuỗi dài nhất là chuỗi chính xác và sẽ tiếp tục làm việc để mở rộng nó. Nếu hai nút phát sóng các phiên bản khác nhau của khối tiếp theo cùng một lúc, một số nút có thể nhận được cái này hoặc cái kia trước. Trong trường hợp đó, chúng làm việc trên cái đầu tiên chúng nhận được, nhưng lưu nhánh khác trong trường hợp nó trở nên dài hơn. Sự ràng buộc sẽ được phá vỡ khi chứng minh công việc tiếp theo được tìm thấy và một nhánh trở nên dài hơn; các nút đang làm việc trên nhánh khác sau đó sẽ chuyển sang nhánh dài hơn.

Việc phát sóng các giao dịch mới không nhất thiết phải đến tất cả các nút. Miễn là chúng đến được nhiều nút, chúng sẽ sớm được đưa vào một khối. Việc phát sóng khối cũng chịu đựng được các tin nhắn bị mất. Nếu một nút không nhận được một khối, nó sẽ yêu cầu khối đó khi nó nhận được khối tiếp theo và nhận ra rằng nó đã bỏ lỡ một khối.

6. Khích lệ

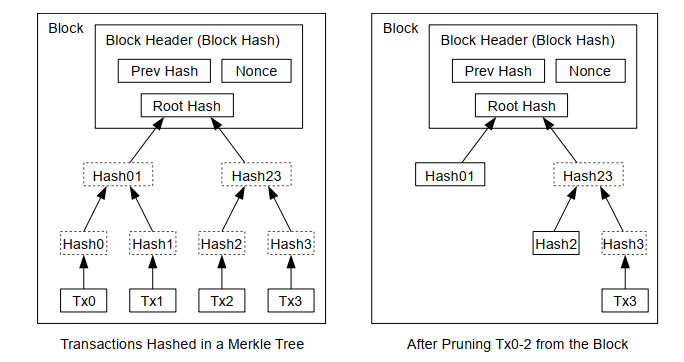
Theo quy ước, giao dịch đầu tiên trong một khối là một giao dịch đặc biệt khởi tạo một đồng tiền mới thuộc sở hữu của người tạo ra khối đó. Điều này thêm một động lực cho các nút hỗ trợ mạng lưới và cung cấp một cách để ban đầu phân phối tiền vào lưu thông, vì không có cơ quan trung ương nào để phát hành chúng. Việc liên tục bổ sung một lượng tiền xu mới không đổi tương tự như những người khai thác vàng bỏ ra tài nguyên để thêm vàng vào lưu thông. Trong trường hợp của chúng ta, đó là thời gian CPU và điện năng đã được tiêu hao.

Động lực cũng có thể được tài trợ bằng phí giao dịch. Nếu giá trị đầu ra của một giao dịch nhỏ hơn giá trị đầu vào của nó, sự khác biệt là phí giao dịch được thêm vào giá trị khuyến khích của khối chứa giao dịch đó. Khi một số lượng tiền xu nhất định đã được đưa vào lưu thông, động lực có thể chuyển hoàn toàn sang phí giao dịch và hoàn toàn không có lạm phát.

Động lực có thể giúp khuyến khích các nút duy trì sự trung thực. Nếu một kẻ tấn công tham lam có thể tập hợp nhiều sức mạnh CPU hơn tất cả các nút trung thực, hắn sẽ phải lựa chọn giữa việc sử dụng nó để lừa đảo mọi người bằng cách đánh cắp lại khoản thanh toán của mình hoặc sử dụng nó để tạo ra tiền xu mới. Hắn nên thấy việc tuân thủ các quy tắc, những quy tắc đó ưu ái hắn với nhiều tiền xu mới hơn tất cả mọi người cộng lại, sẽ có lợi hơn là phá hoại hệ thống và tính hợp lệ của của cải của chính hắn.

7. Thu hồi dung lượng đĩa

Khi giao dịch mới nhất trong một đồng tiền bị chôn vùi dưới đủ khối, các giao dịch đã được chi tiêu trước đó có thể bị loại bỏ để tiết kiệm dung lượng đĩa. Để tạo điều kiện cho điều này mà không làm hỏng băm của khối, các giao dịch được băm trong một Cây Merkle [7][2][5], với chỉ có gốc được bao gồm trong băm của khối. Sau đó, các khối cũ có thể được nén lại bằng cách loại bỏ các nhánh của cây. Các băm bên trong không cần phải được lưu trữ.



Một tiêu đề khối không có giao dịch sẽ khoảng 80 byte. Nếu chúng ta giả sử các khối được tạo ra cứ sau 10 phút, 80 byte \* 6 \* 24 \* 365 = 4,2MB mỗi năm. Với các hệ thống máy tính thường được bán với 2GB RAM vào năm 2008, và Định luật Moore dự đoán tốc độ tăng trưởng hiện tại là 1,2GB mỗi năm, dung lượng lưu trữ sẽ không phải là vấn đề ngay cả khi các tiêu đề khối phải được giữ trong bộ nhớ.

8. Xác minh thanh toán đơn giản hóa

Có thể xác minh các khoản thanh toán mà không cần chạy một nút mạng đầy đủ. Người dùng chỉ cần giữ một bản sao các tiêu đề khối của chuỗi chứng minh công việc dài nhất, mà anh ta có thể nhận được bằng cách truy vấn các nút mạng cho đến khi anh ta chắc chắn rằng mình có chuỗi dài nhất, và lấy nhánh Merkle liên kết giao dịch với khối mà nó được ghi dấu thời gian. Anh ta không thể tự mình kiểm tra giao dịch, nhưng bằng cách liên kết nó với một vị trí trong chuỗi, anh ta có thể thấy rằng một nút mạng đã chấp nhận nó, và các khối được thêm vào sau đó tiếp tục xác nhận rằng mạng đã chấp nhận nó.

A diagram of a blockchain

Description automatically generated

(Hình ảnh minh họa cây Merkle ở đây, mô tả sự thu gọn dữ liệu sau khi loại bỏ các giao dịch cũ)

Vì vậy, việc xác minh đáng tin cậy miễn là các nút trung thực kiểm soát mạng, nhưng dễ bị tổn thương hơn nếu mạng bị kẻ tấn công chế áp. Trong khi các nút mạng có thể tự mình xác minh các giao dịch, phương pháp đơn giản hóa có thể bị kẻ tấn công lừa bằng các giao dịch giả mạo miễn là kẻ tấn công có thể tiếp tục chế áp mạng. Một chiến lược để chống lại điều này là chấp nhận cảnh báo từ các nút mạng khi chúng phát hiện ra một khối không hợp lệ, nhắc nhở phần mềm của người dùng tải xuống toàn bộ khối và các giao dịch được cảnh báo để xác nhận sự không nhất quán. Các doanh nghiệp nhận được thanh toán thường xuyên có thể vẫn muốn tự chạy các nút của họ để có bảo mật độc lập hơn và xác minh nhanh hơn.

9. Kết hợp và tách giá trị

Mặc dù có thể xử lý các đồng tiền riêng lẻ, nhưng sẽ rất khó khăn để thực hiện một giao dịch riêng biệt cho từng xu trong một khoản chuyển tiền. Để cho phép giá trị được tách và kết hợp, các giao dịch chứa nhiều đầu vào và đầu ra. Thông thường sẽ có hoặc một đầu vào duy nhất từ một giao dịch trước đó lớn hơn hoặc nhiều đầu vào kết hợp các lượng nhỏ hơn, và tối đa hai đầu ra: một cho khoản thanh toán và một trả lại tiền thừa, nếu có, cho người gửi.

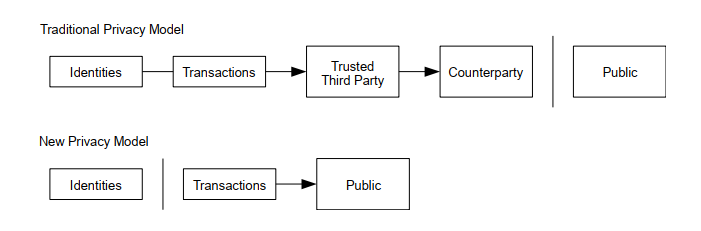
A diagram of a transaction

Description automatically generated

Cần lưu ý rằng việc phân nhánh (fan-out), trong đó một giao dịch phụ thuộc vào một số giao dịch và các giao dịch đó phụ thuộc vào nhiều giao dịch hơn nữa, không phải là vấn đề ở đây. Không bao giờ cần phải trích xuất một bản sao độc lập hoàn chỉnh về lịch sử của một giao dịch.

10. Quyền riêng tư

Mô hình ngân hàng truyền thống đạt được một mức độ riêng tư bằng cách hạn chế quyền truy cập thông tin cho các bên liên quan và bên thứ ba đáng tin cậy. Sự cần thiết phải công bố công khai tất cả các giao dịch loại trừ phương pháp này, nhưng quyền riêng tư vẫn có thể được duy trì bằng cách phá vỡ dòng thông tin ở một nơi khác: bằng cách giữ bí mật các khóa công khai. Công chúng có thể thấy rằng ai đó đang gửi một số tiền cho người khác, nhưng không có thông tin liên kết giao dịch với bất kỳ ai. Điều này tương tự như mức độ thông tin được công bố bởi các sàn giao dịch chứng khoán, nơi thời gian và quy mô của các giao dịch cá nhân, "băng ghi", được công khai, nhưng không cho biết các bên là ai.



Là một tường lửa bổ sung, một cặp khóa mới nên được sử dụng cho mỗi giao dịch để ngăn chúng bị liên kết với một chủ sở hữu chung. Một số liên kết vẫn không thể tránh khỏi với các giao dịch đa đầu vào, điều này chắc chắn cho thấy rằng các đầu vào của chúng thuộc sở hữu của cùng một chủ sở hữu. Nguy cơ là nếu chủ sở hữu của một khóa bị tiết lộ, việc liên kết có thể tiết lộ các giao dịch khác thuộc về cùng một chủ sở hữu.

11. Tính toán

Chúng ta xem xét trường hợp một kẻ tấn công cố gắng tạo ra một chuỗi thay thế nhanh hơn chuỗi trung thực. Ngay cả khi điều này được thực hiện, nó cũng không làm cho hệ thống mở ra những thay đổi tùy tiện, chẳng hạn như tạo ra giá trị từ hư không hoặc lấy tiền mà không bao giờ thuộc về kẻ tấn công. Các nút sẽ không chấp nhận một giao dịch không hợp lệ làm phương thức thanh toán, và các nút trung thực sẽ không bao giờ chấp nhận một khối chứa chúng. Kẻ tấn công chỉ có thể cố gắng thay đổi một trong các giao dịch của riêng mình để lấy lại tiền mà hắn ta gần đây đã chi tiêu.

Cuộc đua giữa chuỗi trung thực và chuỗi của kẻ tấn công có thể được đặc trưng như một Mô hình Ngẫu nhiên Nhị thức (Binomial Random Walk). Sự kiện thành công là chuỗi trung thực được mở rộng thêm một khối, tăng khoảng cách lên +1, và sự kiện thất bại là chuỗi của kẻ tấn công được mở rộng thêm một khối, giảm khoảng cách xuống -1.

Xác suất một kẻ tấn công bắt kịp từ một khoảng cách nhất định tương tự như vấn đề Quá trình phá sản của con bạc (Gambler's Ruin problem). Giả sử một con bạc có tín dụng không giới hạn bắt đầu ở một điểm thua lỗ và chơi tiềm năng một số lượng không giới hạn các lượt chơi để cố gắng đạt được điểm hòa vốn. Chúng ta có thể tính toán xác suất hắn ta đạt được điểm hòa vốn, hoặc một kẻ tấn công bắt kịp chuỗi trung thực, như sau [8]:

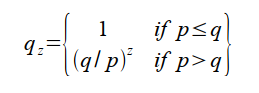
p = xác suất một nút trung thực tìm thấy khối tiếp theo

q = xác suất kẻ tấn công tìm thấy khối tiếp theo

qz = xác suất kẻ tấn công sẽ bắt kịp từ khoảng cách z khối

qz={ 1 nếu p≤q

(q/p)^z nếu p>q}



Cho giả định p > q của chúng ta, xác suất giảm theo cấp số mũ khi số lượng khối mà kẻ tấn công phải bắt kịp tăng lên. Với tỷ lệ cược chống lại hắn, nếu hắn không có một bước tiến may mắn ngay từ đầu, cơ hội của hắn sẽ trở nên nhỏ dần khi hắn càng tụt hậu.

Bây giờ chúng ta xem xét thời gian người nhận một giao dịch mới cần chờ đợi trước khi đủ chắc chắn rằng người gửi không thể thay đổi giao dịch. Chúng ta giả định người gửi là một kẻ tấn công muốn làm cho người nhận tin rằng hắn đã trả tiền cho anh ta một thời gian, sau đó chuyển nó để trả lại cho chính mình sau một thời gian đã trôi qua. Người nhận sẽ được cảnh báo khi điều đó xảy ra, nhưng người gửi hy vọng rằng sẽ quá muộn.

Người nhận tạo ra một cặp khóa mới và cung cấp khóa công khai cho người gửi ngay trước khi ký. Điều này ngăn người gửi chuẩn bị một chuỗi khối trước thời hạn bằng cách liên tục làm việc trên đó cho đến khi hắn may mắn đủ để vượt lên trước, sau đó thực hiện giao dịch tại thời điểm đó. Sau khi giao dịch được gửi, người gửi không trung thực bắt đầu làm việc bí mật trên một chuỗi song song chứa một phiên bản thay thế của giao dịch của hắn.

Người nhận chờ cho đến khi giao dịch được thêm vào một khối và z khối được liên kết sau nó. Anh ta không biết chính xác tiến độ của kẻ tấn công đã đạt được bao nhiêu, nhưng giả sử các khối trung thực mất thời gian trung bình dự kiến cho mỗi khối, tiến độ tiềm năng của kẻ tấn công sẽ là một phân phối Poisson với giá trị kỳ vọng:

λ = zq/p

Để có được xác suất kẻ tấn công vẫn có thể bắt kịp bây giờ, chúng ta nhân mật độ Poisson cho mỗi lượng tiến độ mà hắn ta có thể đạt được với xác suất hắn ta có thể bắt kịp từ điểm đó:

∑(k=0 đến ∞) [λ^k \* e^-λ / k!] \* { (q/p)^(z-k) nếu k≤z ; 1 nếu k>z }

Sắp xếp lại để tránh tổng hợp phần đuôi vô hạn của phân phối...

1 - ∑(k=0 đến z) [λ^k \* e^-λ / k!] \* (1 - (q/p)^(z-k))

Chuyển đổi sang mã C...

#include <math.h>

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;

}

Thực hiện một số kết quả, chúng ta có thể thấy xác suất giảm theo cấp số mũ với 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

Lời giải cho P nhỏ hơn 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

12. Kết luận

Chúng tôi đã đề xuất một hệ thống cho các giao dịch điện tử mà không cần dựa vào sự tin tưởng. Chúng tôi bắt đầu với khuôn khổ thông thường của các đồng tiền được tạo ra từ chữ ký số, điều này cung cấp khả năng kiểm soát mạnh mẽ về quyền sở hữu, nhưng không đầy đủ nếu không có cách nào để ngăn chặn việc chi tiêu kép. Để giải quyết vấn đề này, chúng tôi đã đề xuất một mạng ngang hàng sử dụng chứng minh công việc để ghi lại lịch sử giao dịch công khai mà nhanh chóng trở nên không khả thi về mặt tính toán đối với một kẻ tấn công để thay đổi nếu các nút trung thực kiểm soát phần lớn sức mạnh CPU. Mạng lưới mạnh mẽ trong sự đơn giản không cấu trúc của nó. Các nút hoạt động cùng một lúc với sự phối hợp nhỏ. Chúng không cần phải được xác định, vì các tin nhắn không được định tuyến đến bất kỳ nơi nào cụ thể và chỉ cần được phân phối trên cơ sở nỗ lực tốt nhất. Các nút có thể rời khỏi và tham gia lại mạng lưới bất cứ lúc nào, chấp nhận chuỗi chứng minh công việc là bằng chứng cho những gì đã xảy ra trong thời gian chúng vắng mặt. Chúng bỏ phiếu bằng sức mạnh CPU của chúng, thể hiện sự chấp nhận các khối hợp lệ bằng cách làm việc để mở rộng chúng và từ chối các khối không hợp lệ bằng cách từ chối làm việc trên chúng. Bất kỳ quy tắc và khuyến khích nào cần thiết đều có thể được thực thi với cơ chế đồng thuận này.

**Tài liệu tham khảo:**

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal

trust requirements," In 20th Symposium on Information Theory in the Benelux, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In Journal of Cryptology, vol 3, no

2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping,"

In Sequences II: Methods in Communication, Security and Computer Science, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In Proceedings of the 4th ACM Conference

on Computer and Communications Security, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure,"

http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. 1980 Symposium on Security and

Privacy, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.