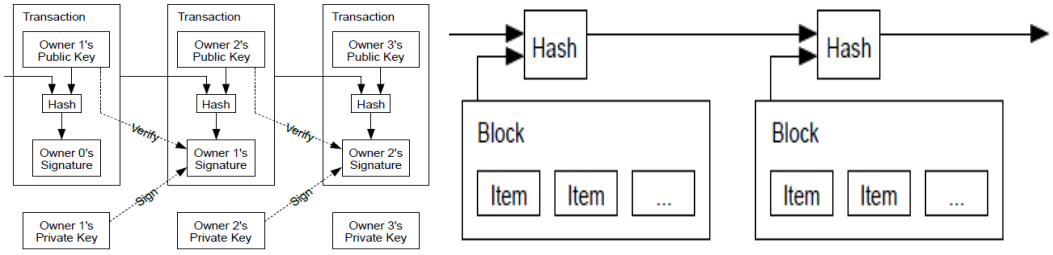
A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one to another party without going through a financial institution. Here the researcher proposes a solution to double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work form a record that can't be changed without the proof-of-work. As a majority of CPU power is controlled by nodes that aren't cooperating to attack the network, they'll generate the longest chain & outpace attackers. Messages are broadcast on a best effort basis & nodes can leave & rejoin the network. While the system works well enough for transactions suffers from the inherent weaknesses of the trust-based model. Completely non-reversible transactions aren't really possible, financial institutions can't avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size & cutting off the possibility for small casual transactions & there's a broader cost in the loss of ability to make nonreversible payments for nonreversible services.

An electronic payment system based on cryptographic proof instead of trust, allowing any 2 willing parties to transact directly with each other without the need for a trusted third party. Transactions that're computationally impractical to reverse would protect sellers from fraud & routine mechanisms could easily be implemented to protect buyers. In this paper, they 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.

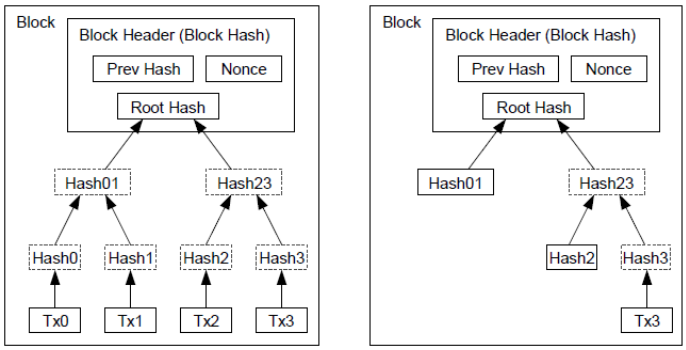


The researcher defines an electronic coin as a chain of digital signatures which is transactions. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction & the public key of the next owner & adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership. The solution they propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped & widely publishing the hash post. 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 one in its hash, forming a chain with each additional timestamp reinforcing the ones before it. To implement a distributed timestamp server on a peer-to-peer basis, they'll need to use a proof-of-work system like Hashcash. Proof-of-work involves scanning for a value that when hashed, such as with SHA-256 & solves the problem of determining representation. 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.

Steps to run this network are given:
Firstly, New transaction broadcast to all nodes. Then, node collects new transactions into a block & works on finding a difficult proof-of-work for its block. When a node finds a proof-of-work, it broadcasts the block to all nodes which accept the block only if all transactions in it are valid & not already spent. Finally, nodes express their acceptance of the block.

Then 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 & be completely inflation free.

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 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 don't need to be stored.

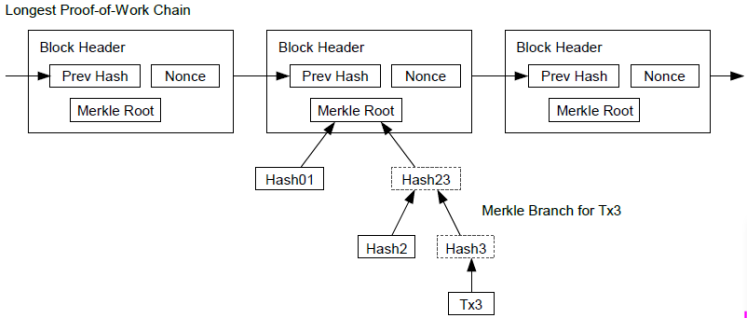


Transactions Hashed in a Merkle Tree — After Pruning Tx0-2 from the Block

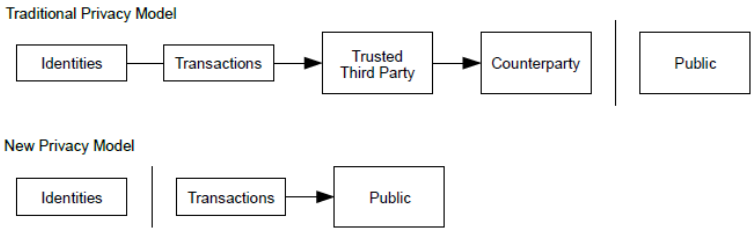
A block header with no transactions would be about 80 bytes. If they suppose blocks are generated every 10 minutes, 80 bytes*6*24*365 = 4.2MB/year. With computer systems typically selling with 2GB of RAM as of 2008, & Moore's Law predicting current growth of 1.2GB per year, storage shouldn't be a problem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

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 & obtain the Merkle branch linking the transaction to the block it's timestamped in. Here, 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.

To allow value to be split & combined, transactions contain multiple inputs & outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts & at most 2 outputs: one for the payment & one returning the change, then return to the sender.



The traditional banking model achieves a level of privacy by limiting access to information to the parties involved & 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. This is similar to the level of information released by stock exchanges, where the time & size of individual trades, the tape is made public, but without telling who the parties were.



We consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. The race between the honest chain & 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, & 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. Here, p = probability an honest node finds the next block
q = probability of attacker finds the next block
qz = probability of attacker catch up from z blocks behind
qz= {1 if p≤q, (q/p)^z if p>=q}

Following that, the receiver generates a new key pair & gives the public key to the sender shortly before signing. 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 & z blocks have been linked after it. the attacker's potential progress will be a Poisson distribution with expected value: $\lambda = z\frac{q}{p}$
To get the probability the attacker could still catch up now, they multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point: $\sum_{k=0}^{\infty}\frac{\lambda^k e^{-\lambda}}{k!}\cdot\begin{cases}(q/p)^{(z-k)} & if\ k\le z\\ 1 & if\ k>z\end{cases}$
Rearranging to avoid summing the infinite tail of the distribution: $1-\sum_{k=0}^{z}\frac{\lambda^k e^{-\lambda}}{k!}\left(1-(q/p)^{(z-k)}\right)$
The converting C code & running some results of the researchers the probability drop off exponentially with z are given.

```
#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;}
```

q=0.1	q=0.3
z=0 P=1.0000000	z=0 P=1.0000000
z=1 P=0.2045873	z=5 P=0.1773523
z=2 P=0.0509779	z=10 P=0.0416605
z=3 P=0.0131722	z=15 P=0.0101008
z=4 P=0.0034552	z=20 P=0.0024804
z=5 P=0.0009137	z=25 P=0.0006132
z=6 P=0.0002428	z=30 P=0.0001522
z=7 P=0.0000647	z=35 P=0.0000379
z=8 P=0.0000173	z=40 P=0.0000095
z=9 P=0.0000046	z=45 P=0.0000024
z=10 P=0.0000012	z=50 P=0.0000006

Solving for P less than 0.1% gives the final result are given in the paper.

Finally, we can say, the researcher has proposed a system for electronic transactions without relying on trust. they 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, they 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. Thus, the network is robust in its unstructured simplicity.