Bitcoin

Sławomir Górawski

Agenda

- Introduction by breaking up Satoshi’s paper
  - Motivation
  - Transactions
  - Proof-of-Work
  - Incentive
  - Storage
  - Privacy
  - Safety
- Practical demonstration using a local testnet
  - Creating wallets
  - Transferring money
  - Inspecting data

Motivation

Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes.

Merchants must be wary of their customers, hassling them for more information than they would otherwise need.

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.

Before

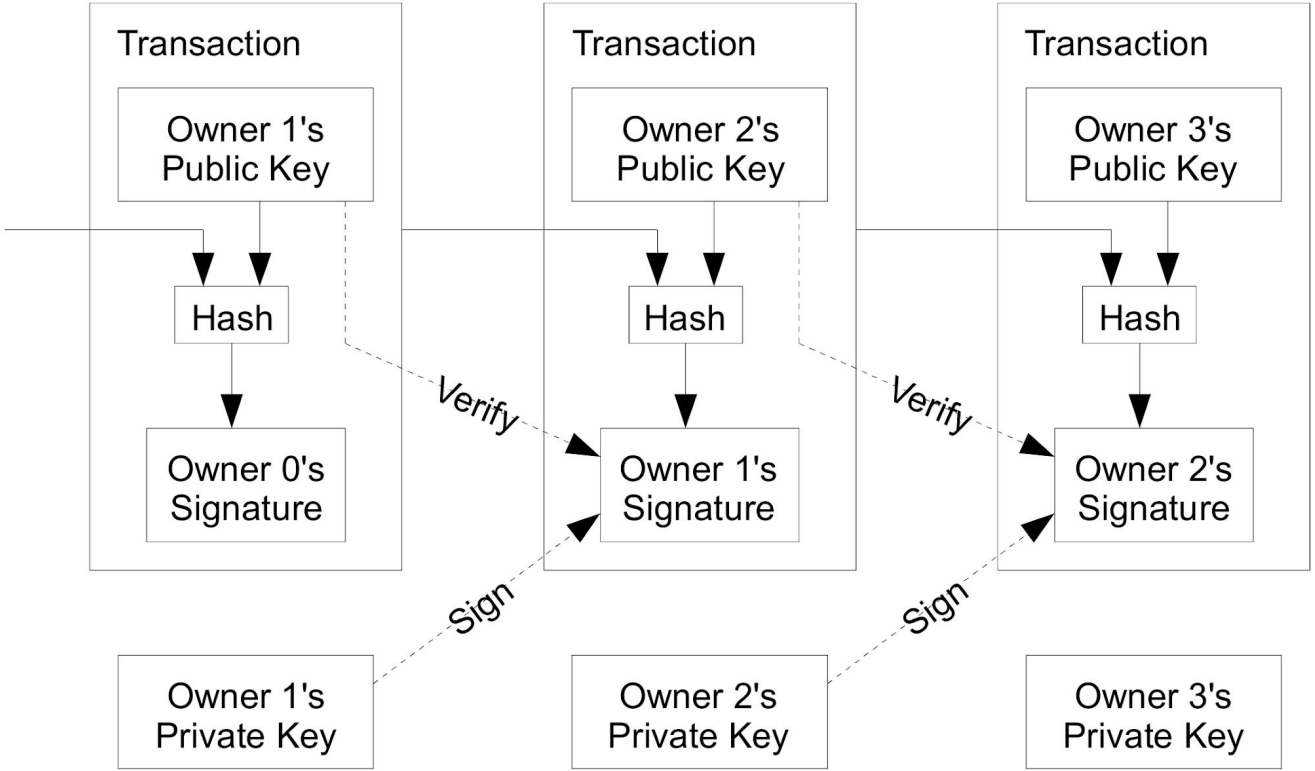
Intermediaries/third-parties
+fees, restrictions, slow, closed

After, with Bitcoin

The Bitcoin network
Open, peer-to-peer, secure, instant, free

REALITY

Transactions



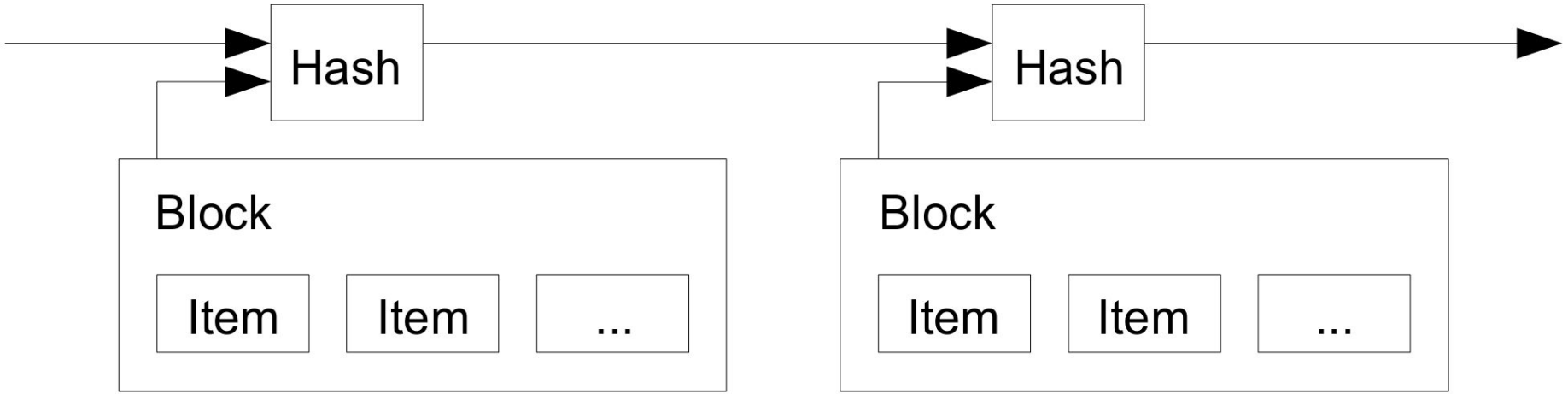
Double-spending problem

The problem of course is the payee can't verify that one of the owners did not double-spend the coin.

We need a way for the payee to know that the previous owners did not sign any earlier transactions.

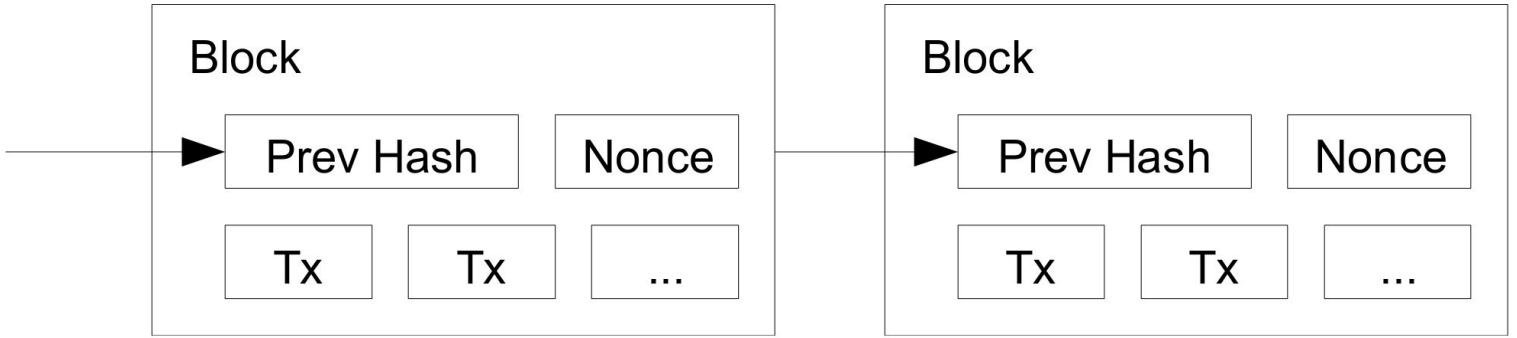
The only way to confirm the absence of a transaction is to be aware of all transactions.

Timestamp server



Proof-of-Work

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.



Proof-of-Work

If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs.

The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it.

...the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour.

Network

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.

Multiple blockchain versions

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.

...the nodes that were working on the other branch will then switch to the longer one.

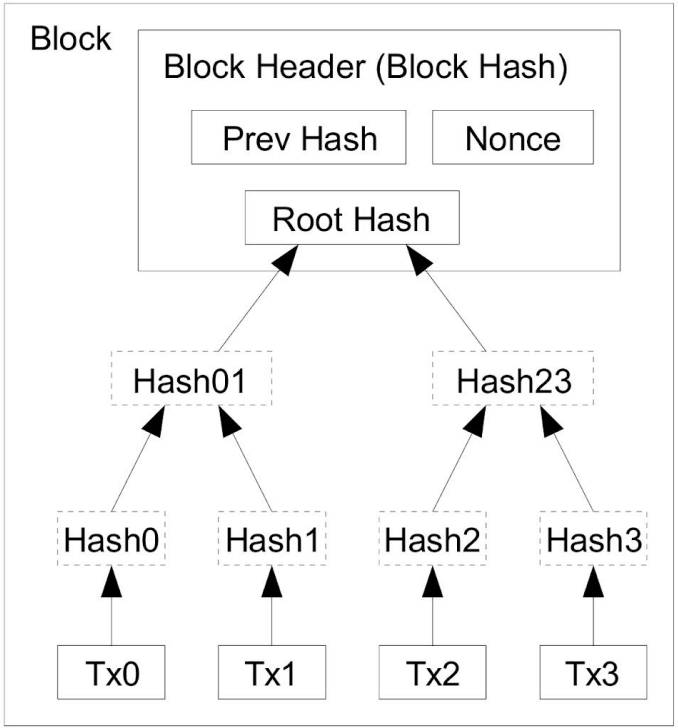
Incentive

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block.

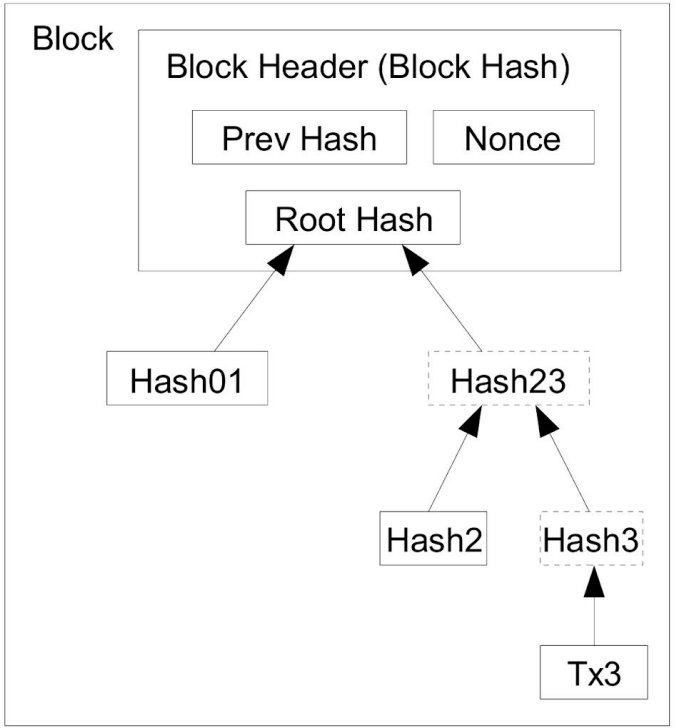
The incentive can also be funded with transaction fees.

The incentive may help encourage nodes to stay honest.

Storage



Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

Storage

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.

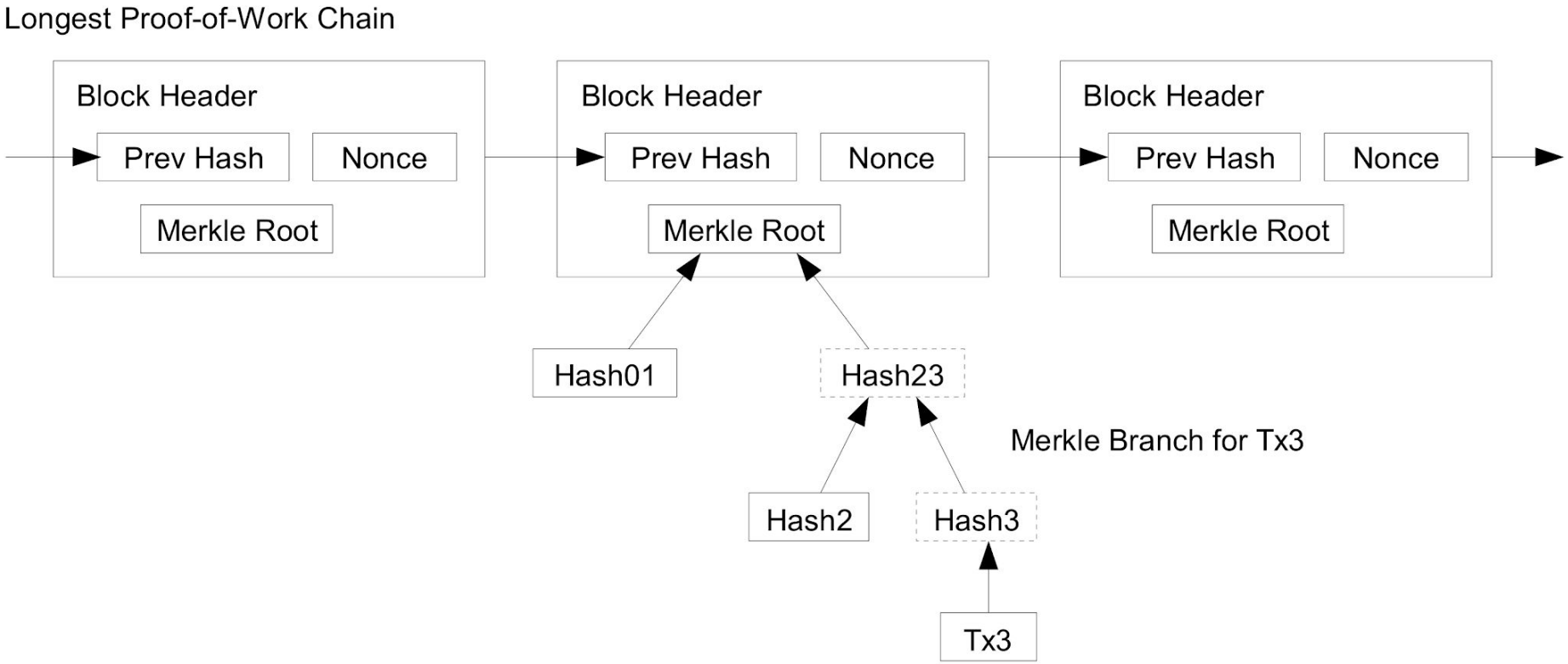
Simplified payment verification

It is possible to verify payments without running a full network node.

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.

Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

Simplified payment verification

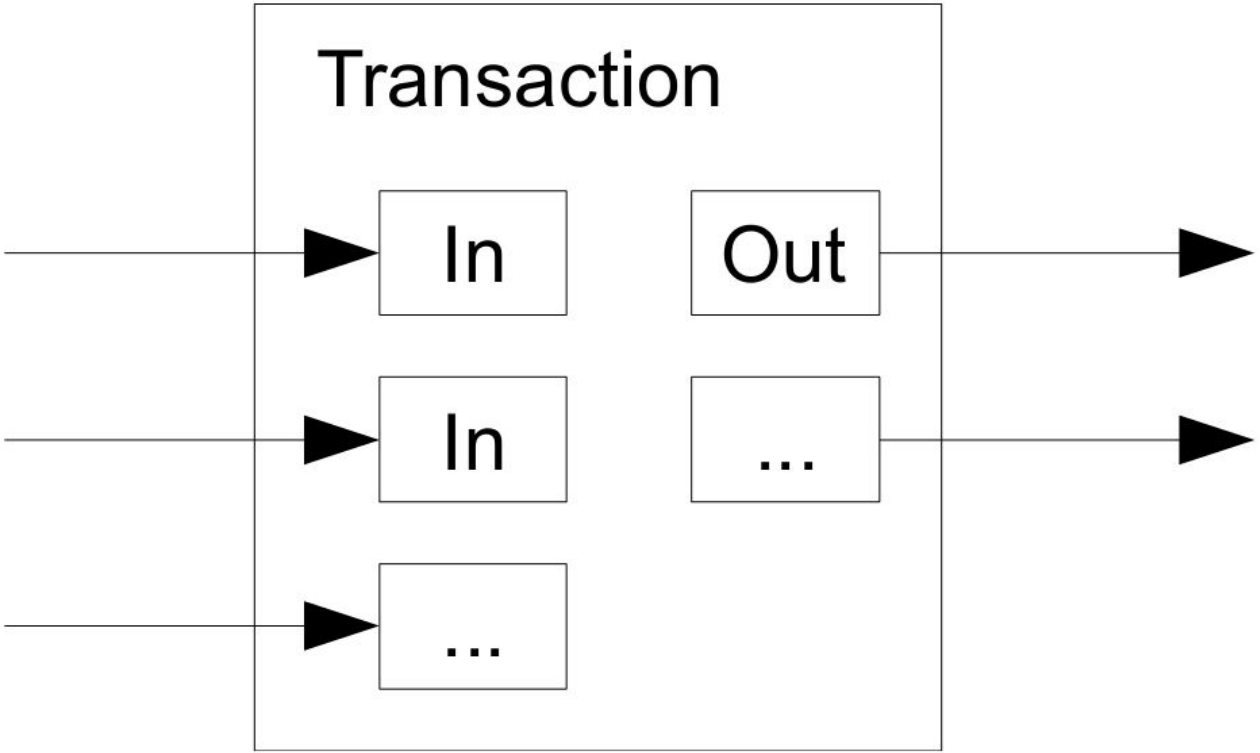


Combining and splitting value

...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.

Combining and splitting value



Privacy

Traditional Privacy Model

New Privacy Model

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.

Safety

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…

An attacker can only try to change one of his own transactions to take back money he recently spent.

Safety

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.

Safety

p = probability an honest node finds the next block

p = probability the attacker finds the next block

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

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

Safety

Given our assumption that p > q, the probability drops exponentially as the number of blocks the attacker has to catch up with increases.

Safety

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.

The recipient waits until the transaction has been added to a block and z blocks have been linked after it.

...the attacker's potential progress will be a Poisson distribution with expected value:

$$\lambda = z \frac{q}{p}$$

Safety

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:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \leq z \\ 1 & \text{if } k > z \end{cases}$$

Safety

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

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \left(1 - (q / p^{(z-k)}\right)$$

Simulation

```
#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;
}
```

Simulation

```
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
```

Simulation

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
```

Conclusion