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 re!uired to prevent double-spending. " e propose a solution to the double-spending problem using a peer-to-peer network. #he 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. #he longest chain not only serves as proof of the se!uence of events witnessed but proof that it came from the largest pool of $%& power. As long as a ma'ority of $%& power is controlled by nodes that are not cooperating to attack the network they(ll generate the longest chain and outpace attackers. #he network itself re!uires minimal structure.) essages are broadcast on a best effort basis and nodes can leave and re'oin the network at will accepting the longest proof-of-work chain as proof of what happened while they were gone.

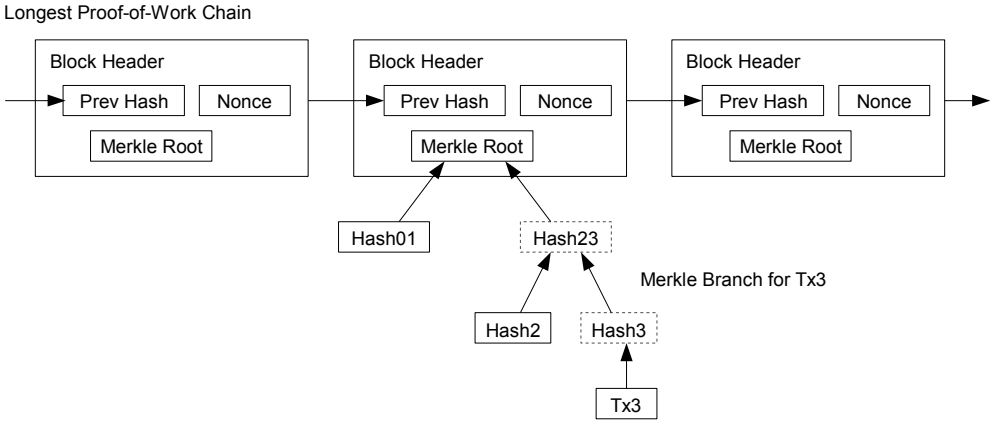
1. Introduction

$ommerce on the *nternet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. " hile the system works well enough for most transactions it still suffers from the inherent weaknesses of the trust based model. $ompletely non-reversible transactions are not really possible since financial institutions cannot avoid mediating disputes. #he cost of mediation increases transaction costs limiting the minimum practical transaction si+e 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. " ith the possibility of reversal the need for trust spreads.) erchants 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. #hese 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.

" hat 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. #ransactions that are computationally impractical to reverse would protect sellers from fraud and routine escrow mechanisms could easily be implemented to protect buyers. *n 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. #he system is secure as long as honest nodes collectively control more $%& power than any cooperating group of attacker nodes.

8. Simplified Payment Verification

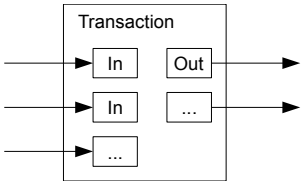
*t is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain which he can get by !uerying network nodes until he(s convinced he has the longest chain and obtain the) erkle branch linking the transaction to the block it(s timestamped in. 4e 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. " hile 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. 6ne 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. 3usinesses that receive fre!uent payments will probably still want to run their own nodes for more independent security and !uicker verification.

9. Combining and Splitting Value

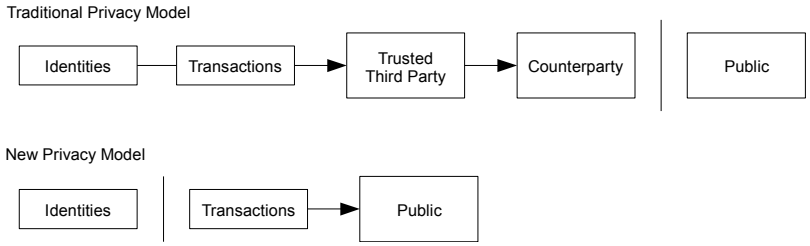
Although it would be possible to handle coins individually it would be unwieldy to make a separate transaction for every cent in a transfer. #o allow value to be split and combined transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts and at most two outputs7 one for the payment and one returning the change if any back to the sender.



*t 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. #here is never the need to extract a complete standalone copy of a transaction(s history.

10. Privacy

#he traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. #he necessity to announce all transactions publicly precludes this method but privacy can still be maintained by breaking the flow of information in another place7 by keeping public keys anonymous. #he public can see that someone is sending an amount to someone else but without information linking the transaction to anyone. #his is similar to the level of information released by stock exchanges where the time and si+e of individual trades the DtapeD 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. #he risk is that if the owner of a key is revealed linking could reveal other transactions that belonged to the same owner.

11. Calculations

"e consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. -ven 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.

#he race between the honest chain and an attacker chain can be characteri+ed as a 3inomial Bandom "alk. #he success event is the honest chain being extended by one block increasing its lead by E, and the failure event is the attacker(s chain being extended by one block reducing the gap by -,.

#he probability of an attacker catching up from a given deficit is analogous to a Aambler(s Buin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. "e can calculate the probability he ever reaches breakeven or that an attacker ever catches up with the honest chain as follows /=07

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 + blocks behind

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

Aiven our assumption that $p > q$ the probability drops exponentially as the number of blocks the attacker has to catch up with increases. "ith 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.

"e now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can(t change the transaction. "e 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. #he receiver will be alerted when that happens but the sender hopes it will be too late.

#he receiver generates a new key pair and gives the public key to the sender shortly before signing. #his 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. 6nce the transaction is sent the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

#he recipient waits until the transaction has been added to a block and z blocks have been linked after it. 4e 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 %oisson distribution with expected value7

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

#o get the probability the attacker could still catch up now we multiply the %oisson density for each amount of progress he could have made by the probability he could catch up from that point7

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

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

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

$onverting to $ 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;
}
```