Polycash: An Increased-Decentralization P2P Blockchain System

Asher Wrobel

https://github.com/Ashy5000

Abstract

Most widely-used cryptocurrency blockchains utilize the PoW (Proof of Work) or PoS (Proof of Stake) consensus algorithms. Unfortunately, these algorithms face a variety of issues. PoW decreases decentralization with mining pools and uses large amounts of energy, while PoS systems can experience centralization and require more trust than PoW blockchains. In response to these issues, APoW(Adjusted Proof of Work) is introduced, using difficulty adjustment algorithms that increase decentralization and decrease power consumption in the blockchain. Combining this consensus algorithm with a new system of verification, time verifiers, creates Polycash, a blockchain designed to allow secure transactions and the operation of peer-to-peer applications to be conducted with increased decentralization and security as opposed to traditional solutions.

1 Introduction

In the current day, financial institutions have largely become adopted as a mediary for digital transactions. Using financial institutions relies on trust, an issue attempted to be resolved by blockchain-based cryptocurrencies. However, blockchains are becoming increasingly centralized when using the existing PoW of PoS consensus algorithms. In PoW, large mining pools have large influence over the state of the blockchain, and cooperation between multiple pools could lead to a 51% attack on the network. In PoS, individuals with large amounts of stake are able to have influence over the blockchain and the network in a similar manner. There are also other issues. PoW uses large amounts of energy, and PoS has low resistance to a 51% attack, as only a third of nodes must be malicious to take over the network and blockchain.

To resolve these issues, a new consensus algorithm is needed: the APoW (Adjusted Proof of Work) algorithm. Instead of giving blockchain power directly proportional to mining power, APoW calculates block speed on a per-miner basis. It decreases target time (base 1 minute) by up to 50% for a miner with infinite mining rate, and increases target time by up to 50% for a miner with zero mining rate. This leads to an increase of decentralization in the blockchain and the network, and allows for a lower energy requirement than with a standard PoW blockchain.

To adapt to the updated algorithm, multiple protocol changes must be put into place to ensure the security and reliability of the APoW blockchain.

2 Adjusted Proof of Work

To increase decentralization in the blockchain, a new consensus algorithm, Adjusted Proof of Work (APoW), is employed. The algorithm's aim is to increase the probability of a miner with a lower hashrate having the ability to create a block, and decrease the probability of a miner with a higher hashrate from doing the same. While the system is designed to maintain the higher chance of a higher hashrate miner winning a block, the gap between higher hashrate miners and lower hashrate miners narrows when using APoW.

The mechanism utilized by APoW to increase decentralization in this manner is block difficulty. As opposed to the traditional PoW algorithm, APoW calculates difficulty on a per-miner basis, attempting to adjust the difficulty for that miner every block so the time it takes them to mine a block matches a target time. In order to encourage miners to contribute their computing power to the network while still giving lower hashrate miners a chance of successfully mining a block, a modified version of the sigmoid function is utilized to alter the target time for a given miner based on the difficulty of the previous block they mined:

$$\frac{1}{1+e^{\frac{-(x-mdpm)}{mdpm}}}$$

1 DPM (Difficulty Point per Minute) represents the speed of a miner that can mine a block with difficulty 1 in 1 minute.

x represents the difficulty of the previous block the miner has created divided by the time it took to mine it, measured in DPM.

mdpm is a constant representing 1 million DPM

Dividing the difficulty by the result of the above formula allows faster miners an advantage, but not to the degree of standard PoW systems.

To determine if a cryptographic proof of work is valid, the SHA-3-512 hash of the block, represented in this case as an unsigned 64-bit integer, must be less than the maximum value of an unsigned 64-bit integer (18446744073709551615) divided by the block's difficulty. This means if block A has difficulty Da and block B has difficulty Db, and if Db / $Da = n$, then block B is expected to take n times longer to mine than block A. In other words, difficulty is proportional to mining time.

3 Time Verifiers

When difficulty is adjusted based on the time it takes for a miner to create blocks, it becomes necessary to ensure miners are honest about the time they start and finish mining. If these two timestamps can be verified, it would also prevent all manipulation of past blocks in the blockchain, even if 51% of the mining power in the network is controlled. To implement this security feature, time verifiers are used in the network. Any node that has mined a block may become a time verifier. When a node starts or finishes mining a block, they must request and collect the signatures of time verifiers. Time verifiers will provide their signature if the current time is within a 10-second range of the time they need to verify. There may be no less than 75% of the time verifiers in the previous block in the current block, and each additional signature beyond the number of signatures in the previous block earns the miner a reward. Because of the reward gained from adding additional time verifiers, miners will attempt to gather signatures from as many nodes as possible. Therefore, the number of signatures will roughly match the number of verifiers in the network. As a given miner will never have control of over 50% of the verifiers in the network, they will fall short of the required number of signatures if they attempt to lie about the times they begin and end mining.

In addition to the number of signatures a miner needs to create a valid block, the majority of the verifiers the miner gathers signatures from must also be verifiers in the previous block. This almost completely prevents a miner from gaining malicious verification. Even if a miner controls 51% percent of time verifiers, these verifiers must not sign blocks- if they did, they would contribute to the number of verifiers in a block, and the number of signatures the miner could receive in their malicious block would still drop by about 50%, below the required 25%. This means a miner must control over 75% of miners in the network *and* 50% of the computing power. Without the second rule, miners controlling malicious verifiers could have those verifiers not sign blocks, so they would not contribute to the verifier count. Then, they could be used to verify the malicious block. However, none or very few of those verifiers will have verified the previous blocks, so with the second rule the malicious block will be invalid. This security measure, along with the next, creates an extremely secure blockchain with significant advantages over one using standard PoW.

4 Smart Contracts

To enable cryptographically verified functionality on the blockchain, smart contracts are implemented. with a blockchain-specific assembly language. When a miner begins mining a block, they evaluate all smart contracts that would be triggered in that block, and, if they initiate any transactions, those transactions are placed in the block to be mined. The block is considered invalid if the smart contract transactions are missing or do not match the correct ones.

A smart contract may automatically make transactions on the behalf of another party, provided that party signs the smart contract when it is created, granting their approval of the instructions contained within it. This functionality is required to allow a contract to be deterministically executed without the need for trust of a party to fulfill their part of the agreement.

Deploying a smart contract requires a deployment fee, paid to the miner by the node deploying the smart contract.

5.1 Virtual Machine

Smart contracts running on the blockchain use a virtual machine with a custom instruction set to deterministically initiate transactions. It uses a Polycash-specific instruction set and assembly language to evaluate these contracts.

5.1.1 Memory Management

The memory of the Polycash virtual machine is contained within a set of virtual "buffers". These are not strictly buffers in the traditional sense. In Rust, the language the VM is written in, they are represented as a value of type Vec<u8>. This means they are expandable lists of bytes with no fixed length. Different data types may be represented by different lengths and organizations of buffers. There are an unlimited number of buffers, and each is an unlimited size, so the virtual machine may hold as much data as it needs in memory (up to its memory limit).

The buffer at hex address 0x00000000, used for global errors is pre-initialized during the VM boot process.

5.1.2 Errors

Errors are split into two types: local errors and global errors. Local errors write an error value (hex code 0x1) into a pre-initialized error buffer if an error occurs in an instruction. Global errors write an error value (also hex code 0x1) into the pre-initialized error buffer at 0x00000000. If the error buffer needed to throw a local error is not found, a global error is thrown.

See Polycash VM Spec for extended subsection 5.1.3 on the VM instruction set.

6 Transaction Body

To allow data to be sent through the blockchain, a body may be attached to each transaction containing data of any form. A data fee is paid to the miner of that block of a size proportional to the amount of data transferred, in bytes.

This transaction body feature allows for the system to be used to create new tokens, implement two-factor authentication, or to permanently and reliably store data, while utilizing the security and reliability of the blockchain.

7 Hashrate Forking Prevention

It is crucial to prevent miners from forking their hashrate through multiple wallet addresses, circumventing the difficulty adjustment algorithm. Three major strategies are applied to this issue.

1. Maximum Miner Count Maximum miner count, as addressed previously, can limit the number of miners in the network and thus the number of wallet addresses miners can fork their mining power into.
2. Block Reward Locking Block reward locking prevents miners gaining block rewards until they have mined at least 3 blocks, lowering the efficiency of mining rate forking.
3. Block Reward Adjustment Block Reward Adjustment decreases the block reward for each new miner that mines a block by 1% in order to impose a net loss of profits on miners that fork their hashrate. This is the most effective approach.

7.1 Block Reward Adjustment

In order to create a net loss of tokens for miners forking their hashrate, a block reward adjustment system is used. For each new miner that joins the network, the block reward decreases by 1%. This change makes miners lose profit when forking their hashrate, instead of gaining tokens. If block reward adjustment was removed, it would be far easier for a miner to fork their hashrate, increasing their rewards.

Conclusion

We have seen that using an alternative consensus algorithm, APoW, with the help of the time verification protocol increases decentralization and decreases energy usage when compared to traditional PoW blockchains. It also avoids the issues in PoS blockchains, most notably the weak subjectivity problem. We have outlined a new blockchain, Polycash, that implements these features alongside a smart contract system in order to enable secure digital transactions and decentralized applications.