Proof-of-Proof:
A Trustless, Permissionless, Decentralized, and Scalable Means of Inheriting Proof-of-Work Security

VeriBlock, Inc.

Justin Fisher
Maxwell Sanchez

Version 1.0i

Abstract

The Proof-of-Proof consensus protocol enables blockchains to inherit proof-of-work security from other blockchains, creating an ecosystem wherein security originates on established blockchains like Bitcoin and extends to other blockchains. Such an ecosystem creates indirect scalability of Bitcoin by utilizing it as a security mechanism for purpose-built chains. Current progress in other areas of scalability, including off-chain transactional networks and sidechains, benefit from a hierarchical security model which enables auxiliary blockchains to operate under the security context of Bitcoin. We propose a means for this inheritance without the involvement or approval of Bitcoin miners, without any centralized entities or federated nodes, and without imposing significant technological limitations on blockchains which adopt this protocol.

1 Introduction

One of the largest issues facing blockchains today—their ability to reach and maintain consensus over blockchain data—has sparked a variety of debates over the complexity and security of a broad selection of upcoming technologies.

The proof-of-work protocol used by Bitcoin has met two primary criticisms: (1) wasteful electricity consumption, and (2) weak on chains with less hash power. The criticism regarding inefficiency in consuming electricity stands on unsolid footing—so too could the filament of a lightbulb be similarly called an inefficient conductor. It is true, however, that smaller cryptocurrencies implementing proof-of-work are vulnerable to relatively low-cost attacks, especially when a larger chain utilizing the same hashing algorithm exists.

In answer to these criticisms, a variety of alternative consensus mechanisms have been proposed and developed, including *proof-of-stake* where users hold balances in native currency to mine, P*ractical Byzantine Fault Tolerance* and the *Ripple Protocol Consensus Algorithm* which adapt the ideas behind classical consensus algorithms like RAFT and Paxos to function on large-scale, trustless systems, and *federated nodes* or *trusted nodes* which act as network authorities and resolve consensus conflicts.

Each of these consensus algorithms trade off some of the advantages of a proof-of-work consensus mechanism: thermodynamically-sound security expectations, trustless and permissionless involvement of miners, mathematically-verifiable replaying of network history for new nodes, significant opportunity costs to attack, etc.

Our proof-of-proof consensus protocol addresses both of these concerns by recycling the hashing power of a powerful proof-of-work blockchain to secure an unlimited quantity of additional blockchains.

2 Previous Technologies

There have been previous attempts to reuse the security of existing, high-security blockchains. In 2011 several blockchains, including Namecoin, adopted merge mining and the AuxPoW protocol, which allowed Bitcoin miners to simultaneously mine both Bitcoin and one or more auxiliary blockchains. In 2013, Mastercoin (now Omni/Omnilayer) launched a protocol which ran on top of Bitcoin by embedding data in the Bitcoin blockchain.

2.1 Merged Mining (AuxPoW)

Merged mining enables the miner of one *parent blockchain* to simultaneously mine on one or more *auxiliary blockchains*. The parent blockchain itself requires no modification to allow other blockchains to merge-mine using AuxPoW. To merge mine, a miner must first build valid block(s) for the auxiliary blockchain(s), and then include some proof of these blocks in the parent blockchain which they attempt to mine (often by embedding the auxiliary blockchain hash in the parent block coinbase transaction). If a miner successfully solves the proof-of-work below a target that satisfies one or more of the merge-mined or parent blockchains, the corresponding block(s) and the proof-of-work solution are combined and relayed to their respective blockchain(s).

Merge-mining requires active participation of parent blockchain miners, and the percentage of the hash-rate which the auxiliary blockchain inherits is the percentage of the parent blockchain's hashing power which is performing merge-mining for the particular auxiliary blockchain.

Merge-mining doesn't scale effectively to secure a large number of auxiliary blockchains, because it would require that parent blockchain miners track and assemble blocks for a large quantity of auxiliary blockchains. It also forces the auxiliary blockchains to use the same hashing algorithm as the parent blockchain. Finally, in some implementations, the opportunity cost for parent blockchain miners to attack the auxiliary blockchain is only the cost of not merge-mining the auxiliary blockchain legitimately, as the miner can continue to mine the parent blockchain (and merge-mine other blockchains) honestly while attempting an attack on another auxiliary network.

2.2 Layered Technologies

Blockchains or pseudo-blockchains inherit the security of highly-secure blockchains by writing the entirety or near-entirety of their blockchain within another blockchain. “Enhanced” or “aware” clients for these technologies act as nodes on the parent blockchain network and look for embedded data which has special meaning to their blockchain. These data are then interpreted under their own rules to perform manipulations of the secondary or embedded blockchain. Some implementations of layered technologies: *Omni/Omnilayer (formerly Mastercoin), Colored Coins,* and *Counterparty.*

Reorganizations in the parent blockchain result in reorganizations on the secondary/embedded blockchain. Generally, a transaction on the parent blockchain is created whenever a transaction on the secondary/embedded blockchain is created. This transactional data or a representation (hash) thereof is embedded into the parent blockchain transaction using a variety of means including OP_RETURN and “impossible” addresses which embed data and don’t have a known corresponding public/private keypair.

Embedding a secondary blockchain within a parent blockchain imposes significant limitations on the secondary blockchain including block-time limitation and minimal storage capacity. For the sake of efficiency, this often requires the secondary blockchain to utilize the address format (and corresponding signature algorithm) of the parent blockchain. Users of the secondary blockchain must also own and spend token on the parent blockchain to transact on the secondary blockchain. Finally, these technologies have significant difficulty scaling beyond the transactional volume (measured in number of transactions, not necessarily size of transactions) supported by the parent blockchain. While technologies like Omni(layer) store and transmit transaction “attachments” on a torrent network, each unique transaction on the Omni blockchain requires a transaction to be broadcast on Bitcoin as well.

2.3 ChainDB

The ChainDB proposal for securing a chain to Bitcoin requires that ChainDB block-building nodes collaborate to build a Bitcoin transaction which denotes the next ChainDB block, limits the secured chain’s minimum block time to the block time of Bitcoin, requires that fully-validating ChainDB nodes also be fully-validating Bitcoin nodes (although a model using SPV-like knowledge of Bitcoin embedded in the ChainDB chain would appear to function as well), and poses an attack vector wherein Bitcoin miners take the fee paid by legitimate ChainDB bidders, but still controls the ChainDB blockchain by including an alternate ChainDB block-defining transaction with an incredibly-large fee. Additionally, an attacker who wishes to modify a ChainDB blockchain would only need to pay a fees that are each a few times greater than the block reward of the ChainDB block to have a significant chance of a multi-block rewrite on the ChainDB blockchain; a ChainDB blockchain would have potential security issues when significantly more value than its per-block coinbase is transferred per block.

2.4 Summary of Existing Technologies

Existing technologies for recycling the proof-of-work security of one parent blockchain onto auxiliary/secondary blockchains comes with significant drawbacks regarding level of security, limitations imposed on the auxiliary/secondary blockchains, and scalability concerns.

3 The Goal

Proof-of-proof aims to enable a *subordinate blockchain* (analogous to a merge mining auxiliary blockchain or a layered technology secondary blockchain) to inherit the complete proof-of-work security of a *master blockchain* (analogous to a parent blockchain).

This inheritance should not impose any non-trivial limitations on the subordinate blockchain, should not require the permission of the master blockchain or the knowledge/involvement of master blockchain miners, should not require any centralized network authority (including federated nodes), and should not leave the subordinate blockchain non-functional in the event that the master blockchain fails. Additionally, non-mining users of the subordinate network should not have to interface with the master blockchain network, nor should they be required to hold any of its native token.

4 PoP Protocol

The PoP protocol introduces a new type of miner who performs periodic publications of one blockchain's current state to another blockchain. These publications are referenced in the event of a potential blockchain reorganization. PoP requires a blockchain has some means of creating blocks, such as low-hashrate local PoW, PoS, etc.

4.1 Definitions

Subordinate Blockchain: A blockchain secured by PoP, which inherits PoW from another blockchain.

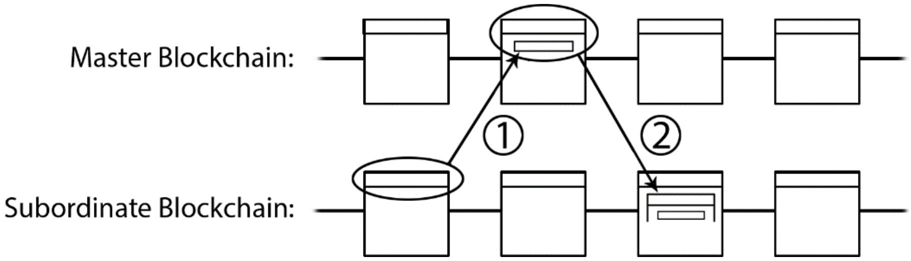
Master Blockchain: An established, high-security blockchain which a subordinate blockchain inherits PoW from.

Blockchain State Data: Data regarding the current state of a blockchain, such as a the most recent block header, block hash, merkle root of transactions, etc.

PoP Miner**:** A new type of miner who performs publications of blockchain state data from a subordinate blockchain to a master blockchain.

4.2 PoP Mining Process Overview

PoP miners serve as the communication/transactional bridges between a subordinate blockchain and a master blockchain. As often as they wish, PoP miners will take the most recent blockchain state data from the subordinate blockchain and publish it to the master blockchain, along with some identifier, which allows them to later receive compensation by creating a master blockchain transaction with the subordinate blockchain state data and identifier embedded in it, and submits it to the master blockchain network. Several different methods can be used for embedding the blockchain state data in a master blockchain transaction: OP_RETURN, fake addresses, fake addresses in multisig, etc. The PoP miner then waits for the transaction to be included in a master blockchain block, constructs some form of proof of publication, adds any identifying information necessary for them to take credit for the publication, and submits this proof back to the subordinate blockchain in the form of a special PoP mining transaction.



4.3 PoP Publication Data

In order to take advantage of OP_RETURN, the subordinate blockchain state data along with some means of identifying the miner for payment needs to fit within 80 bytes. It is recommended that the entire block header of the subordinate blockchain be published to close a security vulnerability discussed later. By using 192-bit hashes for the previous block hash and merkle tree, the standard block header format consisting of a version, previous block hash, merkle tree hash, timestamp, nbits-style target, and nonce only occupies 64 bytes of space, leaving 16 bytes of the OP_RETURN data for PoP miner identification (such as the first 16 bytes of the miner's address). When the PoP miner submits their PoP mining transaction to the subordinate blockchain, they will include the full subordinate blockchain address whose first 16 bytes match these 16 bytes of miner identification.

4.4 PoP Mining Transactions

The specialized PoP mining transaction demonstrates that subordinate blockchain state data was included in a master blockchain transaction, which was included in a master blockchain block. As such, it needs to contain the subordinate blockchain state data which was originally published (along with the miner identification), the master blockchain transaction containing the subordinate blockchain state data, the merkle path (or another form of proof such as a witness for a cryptographic accumulator, if the master blockchain uses a structure other than a merkle tree for transactions) which demonstrates inclusion of the transaction in a master blockchain block, and the master blockchain block header corresponding to the block in which the subordinate blockchain state data was published. Additionally, the mining transaction needs to provide the full miner identification if it wasn't included in its entirety in the published data (example: the full address whose first 16 bytes match the 16 bytes of miner identification published in an OP_RETURN). Finally, additional contextual information may be required, such as sufficient previous block hashes from the master blockchain to enable the subordinate blockchain to construct and validate the entire master blockchain up to the block holding the PoP miner's publication. The simplest algorithm for this requires that the PoP miner submit adequate master blockchain block headers to build from the subordinate network's previously-known and highest-height master blockchain header to the header of the block in which the PoP publication occurs.

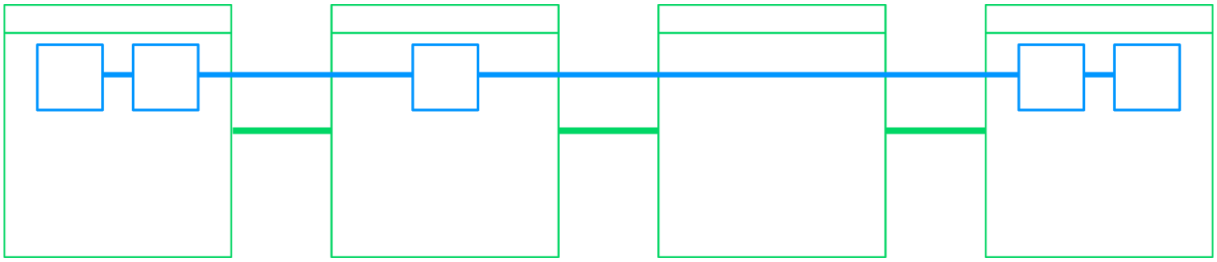
4.5 PoP Mining Transaction Validation

Peers on the subordinate blockchain validate a PoP mining transaction by checking the validity of the published subordinate blockchain state data, checking for its inclusion in the provided master blockchain transaction, ensuring the master blockchain transaction is included in the provided master blockchain block header's merkle tree (or evaluating some other form of proof, such as a cryptographic accumulator witness), and ensuring that the provided block header(s) of the master blockchain build the "longest" PoW chain on the master blockchain.

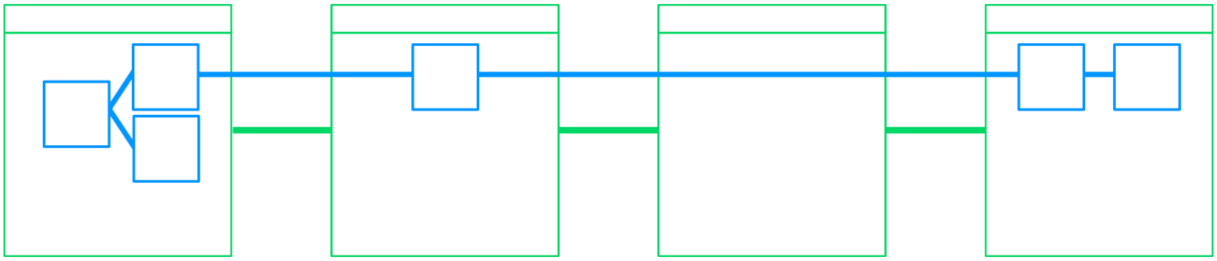
Checking the validity of the subordinate blockchain state data only requires looking back in the subordinate blockchain for the block corresponding to the published state data. Checking for its inclusion in the provided master blockchain transaction involves parsing the transaction and checking the data after OP_RETURN, or for the blockchain state data to appear in an encoded form, such as inside multisig addresses. Then, the master blockchain transaction is hashed and the merkle path is followed, which should result in the merkle root embedded in the provided master blockchain header. Since only the headers of a pure PoW blockchain are sufficient for determining consensus on blocks, subordinate blockchain peers have sufficient information to ensure that the PoP publication occurred in a valid master blockchain block.

4.6 PoP Block Format

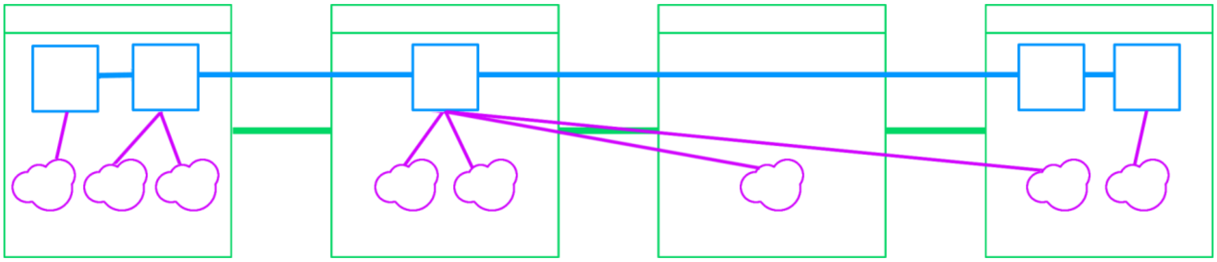
In order for PoP mining transactions to later be used for consensus, they must be stored in the subordinate blockchain. Additionally, the block headers of the master blockchain need to be stored in such a way that consensus of the master blockchain can be tracked without requiring peers to interface with the master blockchain network. As such, the blocks on a blockchain implementing PoP contain a special segment to hold the new master blockchain block headers since the last subordinate blockchain block's included master headers.



In the diagram above, the green blockchain is a subordinate blockchain implementing PoP. The blue blocks are headers from the master blockchain. By linking together the master blockchain headers stored in the subordinate blockchain, the entire master blockchain's PoW consensus can be confirmed. In the event that a fork on the master blockchain occurs, a subordinate block can include multiple competing blocks and allow master blockchain headers embedded in future subordinate blockchain blocks to resolve the conflict:



Proof-of-proof mining transactions can reference, as the master blockchain block in which they published subordinate blockchain state data, any master blockchain block headers stored in their enclosing block or in previous blocks (PoP mining transactions in purple):



To facilitate this, block miners (PoW/PoS/etc.) take the block header data provided by the PoP mining transactions, and embed the zero or more master blockchain headers necessary to provide context for the PoP mining transactions they wish to include in their block.

5 Fork Resolution with PoP

The "best" fork amongst all proposed forks is selected based on a cumulative score. In PoP, however, the score of a fork is calculated relative to another fork; the timeliness of PoP publications in the master blockchain determine their weight and the timeliness of a PoP publication of a subordinate block at a particular height is relative to the first publication of any subordinate block from any of the considered forks.

5.1 Master Blockchain Tracking

In order for a peer on the subordinate blockchain network to perform fork resolution, the peer must construct and evaluate a version of the master blockchain using all of the master blockchain headers provided by all of the subordinate blockchain blocks (including those on every potential fork for which the client has knowledge). In order to do so, the peer collects every single master blockchain header from the subordinate blockchain blocks, and determines consensus according to the rules of the master blockchain (finding the “heaviest” or chain requiring the most computational power to build).

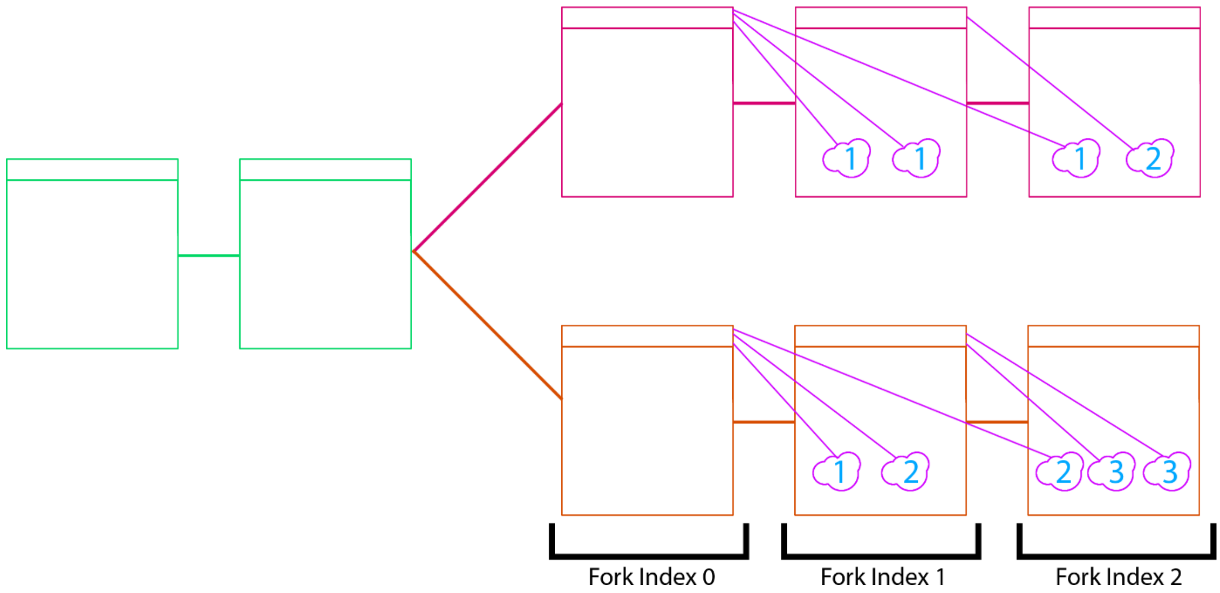
By utilizing information available from all potential forks when reconstructing the master blockchain, the peer can ensure that whatever final picture they get of the master blockchain represents the state of the master blockchain at the latest time any of the blocks in any of the forks were created, allowing for evaluation nearly as if the peer had direct access to the entirety of the master blockchain.

Through this mechanism, the relative weight of any two particular chains can be calculated on-demand by peers who join the network at a later point in time.

5.2 Fork Weight Calculation

The weight of two competing forks is calculated by summing up all of the scores of all of the blocks for which two chains diverge. The scores of competing blocks between multiple forks are calculated relative to each other, following the algorithm:

- For all competing blocks at height n, find all PoP mining transactions in each chain that match said chain’s block at height n
    - For all PoP mining transactions endorsing any block at height n from all competing chains, find the one with the earliest publication (by block height) in the master blockchain. Store this height as m
    - For each competing block n:
        - For each PoP mining transaction endorsing block n:
            - If the PoP mining transaction publishes to a block in the longest known master blockchain fork:
                - Determine the difference in master blockchain publication height from m, add the value floor(1/((difference + 1) * (difference + 1))) to the score for the current block n



In the diagram above, the subordinate blockchain encounters a fork, with two competing chains (red and orange). The blue numbers inside PoP mining transactions represent the index of the master blockchain which they published data to. Not shown for the sake of complexity are the master blockchain blocks embedded in the two forks, which allow the reconstruction of the master blockchain and subsequent ordering of PoP transactions.

To determine which blockchain to accept, the score for each block at each index is calculated relative to the other block at the same index, and the scores for all blocks on each chain are added together: The first PoP publication for either competing block at fork index 0 was at index 1 of the master blockchain. The red block at fork index 0, R_0, is endorsed by three PoP publications which occur at index 1 of the master blockchain, so its score is 3*(1/((1-1+1)*(1-1+1))) = 300; **R_0 = 300**. The orange block at fork index 0, O_0, is endorsed by two PoP publications which occur at index 1 of the master blockchain, and one PoP publication at index 2 in the master blockchain, so its score is 2*(1/((1-1+1)*(1-1+1))) + 1*(1/((2-1+1)*(2-1+1))) = 225; **O_0 = 225**. Similarly, **R_1** = 1*(1/((2-2+1)*(2-2+1))) **= 100** and **O_1** = 2*(1/((3-2+1)*(3-2+1))) **= 50**. The last block in any chain never has any proof weight, because no block has come after it to contain PoP endorsements for it; **R_2 = 0** and **O_2 = 0**. Summing these up, the weight of the red chain is 300 + 100 = 400, and the weight of the orange chain is 225 + 50 = 275. Since 400 > 275, the red chain is the more-endorsed blockchain.

Despite the orange chain's inclusion of 5 PoP mining transactions compared to the red chain's inclusion of 4 PoP mining transactions, the relative timeliness of the red chain's transactions caused it to have a higher proof weight, making it the better chain.

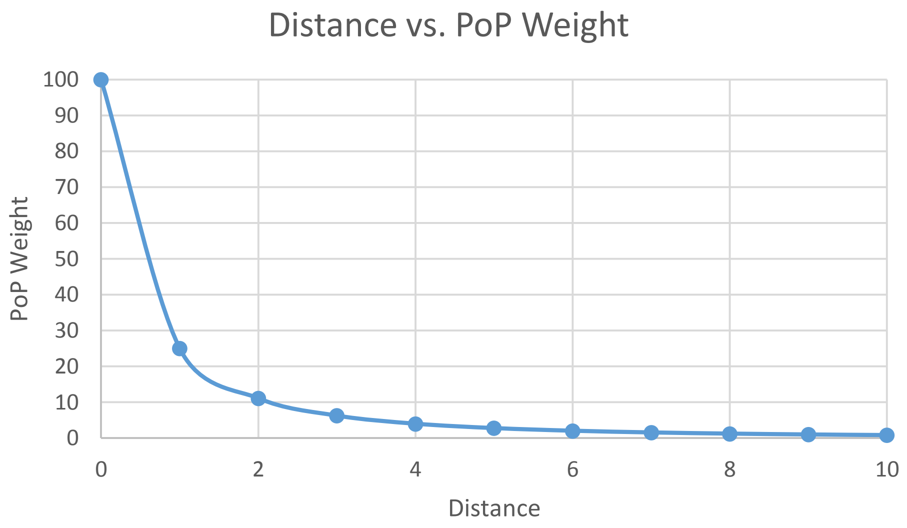
5.3 Fork Resolution Design Rationale

The use of relative weighting of PoP mining transactions based on their timeliness in the master blockchain makes it incredibly difficult for an attacker to produce a fork of any significant period of time into the future without forking the master blockchain itself.

As such, an attacker needs to generate their potential fork of the blockchain alongside the legitimate network, and must do so in full public view by publishing the block hashes of their forking chain in the master blockchain. Anyone can monitor the master blockchain for forks being built and delay accepting transactions until the fork is resolved, and subordinate blockchain networks can implement additional features to invalidate these chains using mechanisms such as balance-based voting to invalidate the attacker's chain before it is released onto the network.

5.4 Weighting Function

The weighting function suggested for a PoP mining transaction is floor(1/((distance+1)^2)), where distance is the afore-described distance between the PoP mining transaction's corresponding master blockchain block and the first master blockchain block in which any block of any considered competing chain at the same index was first published to the master blockchain. This formula can be tweaked to better fit the desired security profile of a particular subordinate blockchain network. Using a function which trends towards 0 faster will result in an existing blockchain being easier to attack, but also increases the possibility of short-term in-step attacks, where an attacker attempts to get a single block in their attacking chain into a master blockchain block before the corresponding legitimate chain block occurs on the network. This suggested function will continue to be tweaked as we run more simulations of different attack scenarios. The following graph illustrates the suggested distance-vs-PoP-weight function.



Instead of calculating the function each time, a simple lookup table can be used:

Distance	Weight
0	100

1	25
2	11
3	6
4	4
5	2
6	2
7	1
8	1
9	1
>=10	0

Any PoP mining transaction corresponding to a master blockchain block 10 or more master blockchain blocks after the first publication of the relevant subordinate blockchain block index has no weight.

As mentioned below as a solution to a potential vulnerability, a similar weighting scheme can also be applied to prioritize the relative scores of blocks closer to the forking point, to ensure that subordinate blockchain forks need to be announced to the master blockchain early.

6 Potential Attack Vectors and Mitigations

As with all consensus mechanisms, an adversarial party can attempt to force the network to reestablish consensus. On a properly-implemented PoP network, these attack vectors include forking the master blockchain and building and proving an alternate subordinate blockchain. Some of the design decisions of PoP eliminate other potential attack vectors of simpler theoretical PoP-like implementations.

6.1 Master Blockchain Forking

In the event that an adversarial party successfully forks the master blockchain, they can re-write the forked master blockchain blocks with new PoP data, enabling them to produce a subordinate blockchain with a higher PoP weight. The amount/length (measured in real-world time, not blocks) of the subordinate blockchain they are able to rewrite is approximately equal to the distance they successfully fork the master blockchain for.

Note that a fork of the master blockchain without specific intention to fork the subordinate blockchain won't result in a subordinate blockchain fork. However, such a reorganization of the master blockchain will cause subordinate blockchain's PoP mining transactions which occurred in the forked master blockchain blocks to no longer exist in the master blockchain, and thus hold no weight. *An area of further research is whether, if the attacker still includes the PoP publications in their new blocks to earn their transaction fees, some sort of process* *could be used by PoP miners to re-demonstrate their old proofs' presence on the new master blockchain.*

In the event that the master blockchain forks but doesn't do so to attack the subordinate blockchain, and the subordinate blockchain's PoP mining transactions are impacted and no longer hold weight, the current security of the subordinate blockchain will drop down to its own intermediate (PoW/PoS/etc.) consensus mechanism until PoP miners publish new blockchain state data to the master blockchain and provide PoP mining transactions back to the subordinate blockchain.

6.2 Building an Alternative High-Proof-Weight Subordinate Blockchain

Performing this attack requires that the adversarial party build an alternate subordinate chain which has a higher proof weight than the current subordinate chain. In order to execute this attack successfully (due to proofs being evaluated on their timeliness), an attacker would need to build their attacking blockchain simultaneously with (or faster than) the current subordinate chain. This requires that the attacker publish their attacking chain's blockchain state data to the master blockchain promptly, allowing users of the network to see the pending attack and its properties. As such, anyone watching the master blockchain would see what block(s) are at risk for the fork, how much stronger (or weaker) the current chain is compared to the adversarial party's chain, and could potentially use some means (like balance-based voting) to invalidate the adversarial chain before it is released to the network.

In another possible (although more difficult) implementation, the attacker would build an alternate subordinate chain whose earlier blocks have little-to-no proof weight when compared to the current chain, but whose later blocks are published extensively to the master blockchain. This sort of attack would still be publicly visible due to the publications on the master blockchain, but it would not necessarily reveal how far back the fork could occur, and would also not appear to users of the network at the time when some of the earlier blocks of the attacking chain were being built without proof weight. To mitigate this attack, blockchain networks simply weight blocks closer to the forking point with significantly more weight (so the sum might look something like 100 * weight_0 + 70 * weight_1 + 55 * weight_2 …), making this attack difficult or impossible to successfully perform.

6.3 Publishing Bogus Subordinate Blockchain State Data

In a version of a PoP implementation where the subordinate blockchain state data published to the master blockchain isn't enough to verify the potential validity of the data, an adversarial party could cause parties on the network to delay in accepting transactions by faking a potential fork which doesn't actually exist. This attack does not require overpowering the intermediate consensus, but only allows the attacker to be a nuisance because the network won't fork if the attacker can't provide the complete blocks for which data tagged to the master blockchain exists. This attack involves the attacker publishing apparently valid blockchain state data for which they don't actually have blocks for.

In a subordinate blockchain which relies solely on PoW for immediate consensus, this can be mitigated by requiring, as PoP currently does, the publication of the entire block header. This way, the attacker cannot publish bogus subordinate blockchain data because the data would not be a valid PoW solution.

In a subordinate blockchain employing PoS, it is also possible to publish additional information proving the ownership of coin age or similar network-asset-based mining resources, or which could be verified by informed network participants such as full nodes (txid containing the coinage claiming to be spent, etc.).

7 Combination with PoS

PoP requires an intermediate method of creating blocks (or other discrete units of consensus) and maintaining short-term consensus pending PoP publications. Implementing PoP on a network using PoW as the intermediate consensus mechanism is straight-forward, given PoP's natural extension of PoW-like consensus. Implementing PoP on a PoS network requires additional considerations, and offers solutions for the long-standing issues with pure PoS.

7.1 Existing PoS Issues

Note: Several implementations of PoS exist. PoS is not a single consensus algorithm, but rather a collection of several closely-related consensus algorithms which share the common trait of "balance-based" (or unspent-output-based) mining, in which miners use their native token balance to produce blocks, and the "resource expenditure" of mining is the time-value of the tokens. Some of the issues present in the original Peercoin implementation of PoS (such as long-range attacks due to fixed stake modifiers) have been solved by newer PoS implementations, and those solved issues will not be resurrected for discussion here.

Two primary issues face the latest iteration of PoS:

1. There is no way to mathematically demonstrate the validity of a blockchain to a new node during bootstrapping (the chain has "weak subjectivity").
2. There is only a short-term solution (last 'n' blocks) to the "nothing at stake" problem.

Both of these issues fall back to the subjectivity of PoS—a number of private keys representing a 'critical mass' of network token ownership at an arbitrary point in a blockchain's history can be used to produce a more valid fork of the network weeks, months, or years into the past, and doesn't require present ownership of any tokens. Additionally, it's impossible to prove that no party has access to a critical mass of network tokens. The Slasher protocol presents a punitive system to solve "nothing at stake" problems in the short-term.

7.2 Mathematical Demonstration of Validity During Bootstrapping

Traditional PoW systems have an objective definition of the "best blockchain," (the blockchain which requires the most cumulative work to build) and assuming that a bootstrapping node has unrestricted access to the blockchain network, the node will always be able to independently determine the best blockchain.

In pure-PoS systems, the solution to the aforementioned critical mass ownership problem is simply to, as part of the protocol, prevent any node from forking back more than a certain number of blocks. Such a system results in a rolling checkpointing system wherein each node simply refuses to remove more than a certain number of blocks ‘n’ from their current ‘best blockchain’ view. As such, if a client uses old coin ownership to create a fork which begins more than n blocks ago, nodes on the network will simply reject the fork. However, a boostrapping client might be fed the illegitimate blockchain first, and subsequently refuse to fork back more than n blocks back on the illegitimate chain, permanently (without human intervention) preventing them from tracking the correct blockchain.

PoP provides a simple solution to this problem, as a blockchain using PoP will have a mathematically-verifiable “best blockchain” defined by the blockchain’s inclusion in a PoW master blockchain.

7.3 More than ‘n’ Blocks “Nothing at Stake” Solution

The current solution to the “nothing at stake” solution in the long-run is for clients to ignore forks that remove more than x blocks from the current blockchain, as explained above.

Refusing to perform a blockchain reorganization more than n blocks deep presents one interesting attack vector: deploying a fork on the blockchain which forks exactly y blocks of history during the propagation of a new block, leaving portions of the network (who haven’t yet seen the latest block and as such as willing to fork back x blocks) permanently desynchronized (without human intervention) from the portions of the network who had already received the new block, and refused to fork back x+1 blocks. This attack’s plausibility and potential damage increases with increasing block propagation times, which can result from larger and more-complex-to-validate blocks. Since we were unable to find any mention of this type of attack elsewhere, we describe it in ***Appendix A***.

Implementing PoP would ensure that the acquisition of a critical mass of ownership of coins at a given time in a blockchain’s history couldn’t be used to attack the network, because the accompanying PoP publications would be either non-existent or irrelevant due to untimeliness, allowing clients to remove the rule regarding maximum fork distance, since forking any significant portion of the blockchain would require successfully and simultaneously attacking the master blockchain.

In fact, successful non-contested PoP endorsements of a block act as an effective “soft maximum forking distance,” growing stronger as the difficulty of creating a fork from before a certain point in history becomes exceedingly implausible due to the PoP weighting algorithms.

8 Appendix A: Limited Reorg Distance Fracturing Attack

8.1 Review of Short-Term Consensus Protection in PoS

In the short term, pure-PoS blockchains can avoid “nothing at stake” issues by requiring deposits (or freezing of assets) for a given period of time in order to enable PoS mining on those coins (a method considered for Ethereum, and termed ‘Slasher’ by Ethereum developers). In normal PoS systems, there is a negligible cost in attempting to produce PoS blocks on top of all possible forks. When a PoS miner receives two or more competing blocks, it is in their best interest to attempt to build upon both chains, or potentially withhold any of the competing blocks which they are unable to mine atop. However, by freezing assets, the network can punish miners who practice this behavior by allowing “bounty hunters” to watch for this type of behavior, provide cryptographic proof (such as two signatures from the same miner which vote for competing blocks at the same height), and receive a portion of the frozen coins.

Naïve short-term forking attacks require ownership of a significant portion of the total network’s staking coins immediately before the attack, making them largely implausible and uneconomical.

The “stake grinding” attack is also ineffective in the short-term due to minimum coin age before staking and dynamic stake modifiers implemented by projects like Blackcoin and Neucoin.

These issues do not exist in PoW networks, as computational power spent attempting to build on one chain can’t be spent attempting to build on another chain.

8.2 Long-Term Attack

In the long term, most of the means of protecting consensus are ineffective. Frozen deposits are eventually returned, long periods of time allow attackers to sell their entire position in the coin or acquire private keys which held a large portion of the staking coins long ago, and grinding attacks become possible since an attacker able to reliably rewrite a critical mass of the blockchain far in the past can explore a number of possible blockchains (each with different transactions/transaction order, which alter things like the stake modifier) bound only by their available computational power. This allows a critical mass of token ownership/control far in the blockchain’s past to, given sufficient computational power, create a more-valid blockchain than the current blockchain.

An attacker attempting to conduct this form of attack doesn’t necessarily own any of the token currently, and is more likely interested in causing a massive disruption in services than in performing double-spends.

8.3 Current Long-Term Attack Solution

In order to eliminate the possibility of long-term attacks from rewriting potentially years of blockchain history, clients on PoS networks are simply programmed to not accept reorganizations which fork the network more than ‘n’ blocks ago. This makes it possible for new bootstrapping peers to be permanently stranded on an incorrect fork, but under normal operation doesn’t cause any potential disruptions to nodes which are always connected, or connect more often than it takes the network to add n blocks.

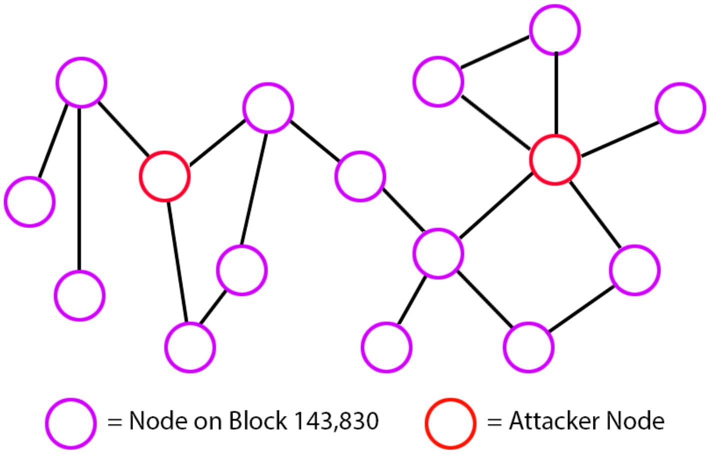
The value n chosen as the maximum reorganization depth relies on several factors, including the period for which coins are locked up (if on a deposit-based PoS system, as Slasher proposes), the time expected for an attacker to acquire old private keys/sell their position in the currency, the speed of blocks on the network, etc.

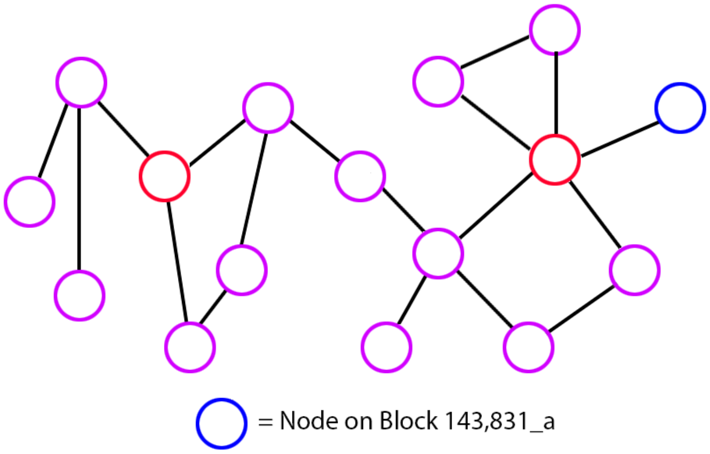
Too small of an n makes it possible for the network to easily desynchronize (since even extremely difficult attacks are plausible in very short timespans due to probability), and too large of an n makes long-term attacks more efficacious.

8.4 Issues with Maximum Reorganization Depth

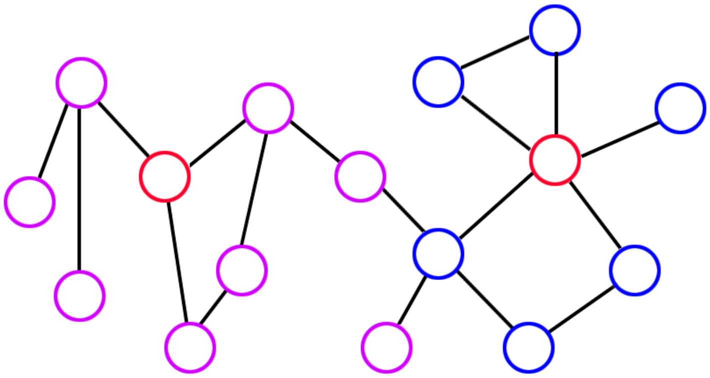
In the event that an attacker was capable of creating a fork which diverges from the correct blockchain n blocks ago, the attacker could release this fork while the current block is still propagating across the network (meaning some peers on the network are at a different block than others).

On a blockchain like Neucoin where the maximum reorganization depth is 43,830 blocks, if the current block height on the network was 143,830, then releasing a fork which forks the network back to block 100,000 would be accepted by all peers. However, releasing the fork back to block 100,000 once the network is at block 143,831 wouldn’t produce any results.

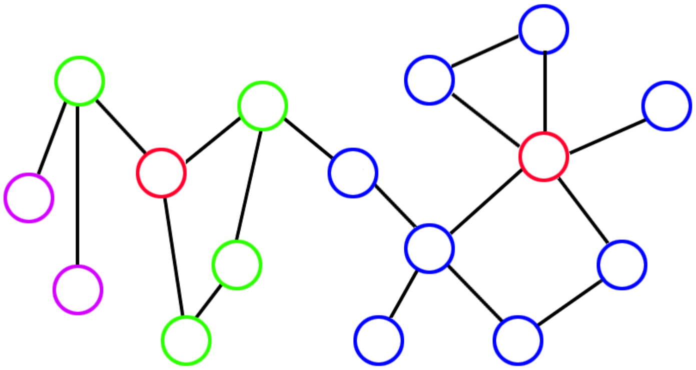




Attacker learns of block 143,831 from one owned node

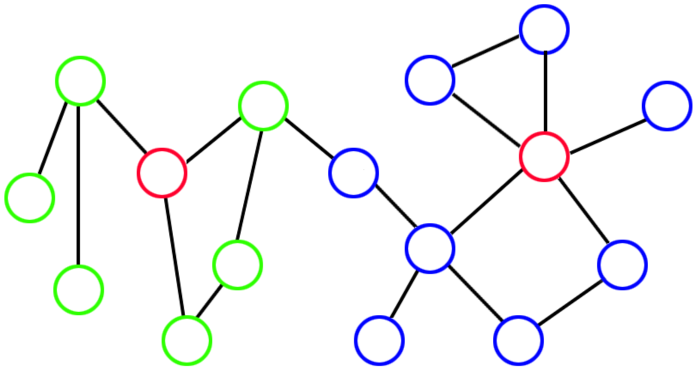


Propagation of Block 143,831_a continues



○ = Node on Block 143,831_b

Attacker releases fork back to 100,000 which ends with block 143,831_b



Network settles, nodes on block 143,831_a and block 143,831_b are unable to reach consensus because they diverge at block 100,000, and switching from _a to _b or from _b to _a would require a reorganization deeper than 43,830 blocks.

Due to network/processing latency of new blocks, there is a period between the entire network being on block 143,830 and the entire network being on block 143,831. During this period, carefully-distributed and well-connected nodes controlled by the attacker could release the fork to their peers as soon as block 143,831 is first observed on the network. As such, peers who are only aware of block 143,830 would accept the fork and overwrite 43,830 blocks of history (and would end on the last block of the fork, which could be 143,831). However, peers which receive the legitimate block 143,831 first don't accept the fork propagating across the network. At this point, the network is fractured into two segments, which both have an apparently-valid block 143,831. The two sides of the split aren't able to reconcile and determine which blockchain is correct, since either side accepting the other side's blockchain would require a reorganization 43,831 blocks deep, which no clients will perform without human intervention, as per protocol.

In a non-punitive PoS system where the optimal self-serving mining strategy involves mining on all available blockchain forks, it is possible that several competing forks stemming from a long-ago common point continue to be built in parallel. When the length since a common ancestor of these competing forks reaches n, clients are likely to permanently desynchronize from each other as they individually choose one of these forks and refuse to fork to any other fork since the other forks require organizations that are deeper than they are willing to perform.

8.5 Mitigation

This attack requires that n be sufficiently large that an attacker can successfully build a "self-sustaining" blockchain based on private keys owned in the past, *or* that n is sufficiently small that shorter-term attacks of a distance n are practical to produce. Setting n to a value as far as possible from both extremes significantly reduces an attacker's ability to perform this attack.

Alternately, implementing PoP removes the need for a maximum reorganization depth (and makes it nearly impossible for a long reorganization to be produced), eliminating the potential for this permanent desynchronization attack.

References

[1] Peercoin Whitepaper [https://peercoin.net/assets/paper/peercoin-paper.pdf]

[2] BlackCoin PoS Whitepaper v2 [https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf]

[3] Neucoin Whitepaper [http://www.neucoin.org/en/whitepaper]

[4] Ethereum Blog Post on Weak Subjectivity [https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity/]

[5] OmniLayer Specifications [https://github.com/OmniLayer/spec]

[6] Counterparty Specifications [http://counterparty.io/docs/protocol_specification/]

[7] Colored Coins Specifications [https://github.com/Colored-Coins/Colored-Coins-Protocol-Specification]

[8] Merged Mining Specifications [https://en.bitcoin.it/wiki/Merged_mining_specification]

[9] ChainDB Whitepaper [https://bitpay.com/chaindb.pdf]