

# Proof-of-Proof and VeriBlock Blockchain Protocol Consensus Algorithm and Economic Incentivization Specifications

VeriBlock, Inc.

[VeriBlock.org](http://VeriBlock.org)

v1.0

## Abstract

The Proof-of-Proof (“PoP”) consensus protocol enables blockchains to inherit the Proof-of-Work security from other blockchains in an entirely Decentralized, Trustless, Transparent, and Permissionless (“DTTP”) manner. PoP functions without the involvement or approval of Bitcoin miners, without any centralized entities or federated nodes, and without imposing any technical limitations on blockchains which adopt the protocol.

The VeriBlock blockchain implements the Proof-of-Proof consensus protocol and is designed to allow the efficient, secure, and easy-to-implement inheritance of Bitcoin’s Proof-of-Work security by a theoretically unbounded quantity of additional blockchains.

To best serve this purpose, many design decisions were made in the VeriBlock blockchain protocol to properly incentivize rational economic actor behavior which benefits its security profile. The resulting incentive system design considers the existing structure and design of the Bitcoin blockchain protocol, the desired characteristics of Proof-of-Proof and Proof-of-Work miner behavior on the VeriBlock blockchain, and the behaviors of PoW and PoP mining markets.

To provide the highest possible security profile in a variety of adverse conditions, the consensus algorithm for the VeriBlock blockchain is designed with consideration for many flavors of consensus attacks.

Current progress in other areas of scalability, including off-chain transactional networks and sidechains, benefit from a hierarchical security model which enables all blockchains to operate under the security context of Bitcoin.

# Contents

|     |  |    |
|-----|--|----|
| 1   | Introduction .....   | 4  |
| 2   | VeriBlock and Proof-of-Proof Introduction.....                           | 5  |
| 2.1 | VeriBlock and Proof-of-Proof At-A-Glance .....                           | 5  |
| 2.2 | Proof-of-Proof: End-User Perspective .....                               | 6  |
| 3   | Desirable Miner Behaviors.....   | 8  |
| 3.1 | SI PoP Miners .....  | 8  |
| 3.2 | SI Block Miners.....   | 9  |
| 3.3 | SP PoW Miners .....  | 9  |
| 4   | Modeling Block Time Probabilities .....                                  | 10 |
| 4.1 | Expected Bitcoin Block Times .....                                       | 10 |
| 4.2 | Expected VeriBlock Block Times .....                                     | 11 |
| 4.3 | Expected Relative Bitcoin Block per VeriBlock Block Occurrences .....    | 12 |
| 4.4 | Expected Relative VeriBlock Block per Bitcoin Block Occurrences .....    | 13 |
| 4.5 | Expected Relative Altchain Block per VeriBlock Block Occurrences .....   | 14 |
| 5   | PoP vs PoW Mining Market Comparison .....                                | 15 |
| 5.1 | PoW Reward Market .....  | 15 |
| 5.2 | PoP Reward Market.....   | 18 |
| 6   | Proof-of-Proof Technical Overview .....                                  | 21 |
| 6.1 | Desirable Proof-of-Proof Protocol Traits .....                           | 21 |
| 6.2 | Keystoning .....   | 22 |
| 6.3 | Reduced Publication View .....   | 24 |
| 6.4 | Publication Continuity Filtering .....                                   | 25 |
| 6.5 | Fork Resolution.....   | 26 |
| 6.6 | Proof-of-Proof Reward Scheme.....  | 29 |
| 6.7 | Alternatives to Coinbase PoP Payouts.....                                | 34 |
| 7   | VeriBlock Blockchain PoP Parameter Selection.....                        | 34 |
| 7.1 | Incentivizing Rapid Publication.....                                     | 35 |
| 7.2 | Incentivizing Return of PoP Data .....                                   | 35 |
| 7.3 | Disincentivizing Bitcoin PoW Miners from Censoring PoP Transactions..... | 35 |
| 7.4 | PoP Payout Round Multipliers / Reward Levels .....                       | 38 |
| 7.5 | VeriBlock Bitcoin Finality .....   | 40 |
| 7.6 | Block Time and Finality Delay Parameter Selection .....                  | 41 |

- 8 VeriBlock Blockchain Specifications..... 45
  - 8.1 Merkle Patricia Tree Ledger..... 45
  - 8.2 Blocksize Calculation and Allowances ..... 45
  - 8.3 Standard Addresses and Standard Transactions ..... 45
  - 8.4 Multisig Addresses and Transactions ..... 47
  - 8.5 PoP Transactions..... 48
  - 8.6 Block Format..... 49
  - 8.7 Difficulty Algorithm..... 52
  - 8.8 Segwit-Like Transactions ..... 52
  - 8.9 Altchain Data Publication Format and ID Tracking ..... 52
  - 8.10 Alternate PoP Data Encoding Format in Bitcoin..... 53
- 9 Integrating VeriBlock Security Into Altchains ..... 56
  - 9.1 Altchain PoP Parameters ..... 57
  - 9.2 Difficulty Algorithm..... 57
  - 9.3 New Transaction Types / Script Templates, Modified Block “Size” Calculation..... 58
  - 9.4 Altchain PoP Payouts ..... 59
  - 9.5 Additional Block Validation Rules ..... 59
  - 9.6 Altchain Fork Resolution..... 60
- 10 False EAD Triggering Prevention With PoW Proofs..... 61
  - 10.1 Background ..... 61
  - 10.2 PoW Proofs Introduction ..... 63
  - 10.3 Parameter Selection ..... 63
  - 10.4 Alternate Proofs When mindm Isn’t Met by topb Blocks..... 65
  - 10.5 [Experimental] Dynamic Keystoning..... 65
  - 10.6 [Experimental] Zero-Knowledge Proofs..... 66
- Appendix A: Summary of VeriBlock Blockchain Parameters..... 67
- Appendix B: VeriBlock Integration Point ..... 68

# 1 Introduction

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

While many altchains adopted the same Proof-of-Work consensus protocol as Bitcoin, these blockchains are vulnerable to 51% attacks (wherein a miner controlling more than half of the Proof-of-Work power of a particular blockchain is able to re-write the history of transactions on the chain). In an attempt to solve this vulnerability, a variety of alternative consensus protocols have been proposed and developed. Each of these consensus algorithms trade off some of the advantages of Proof-of-Work: losing thermodynamically-sound security expectations and permissionless block creation, no-longer having mathematically-verifiable replaying of network history, having the potential for a majority validator set controller to exert complete censorship of the network in perpetuity by preventing new validator set entries, etc.

Our Proof-of-Proof (“PoP”) consensus protocol allows any blockchain, regardless of consensus protocol (PoW, PoS, DPoS, PoC, etc.) to inherit the Bitcoin blockchain’s Proof-of-Work security while simultaneously addressing weak subjectivity and perpetual validator set control for non-PoW consensus.

In order to extend Bitcoin’s PoW security to other blockchain in the most efficient, secure, and scalable manner possible, we designed the VeriBlock blockchain, which acts as an intermediate security aggregation layer. The incentive structure of the VeriBlock blockchain protocol is designed to incentivize the continued operation of the VeriBlock blockchain while maintaining decentralization at both the PoW and PoP miner levels, maintaining a high security profile in a large variety of external fee market and information availability scenarios, de-incentivize attacks against the system by increasing the cost of and reducing or eliminating the possible rewards for a successful attack, and promoting timely and easily-verifiable inclusion of PoP publication data from other blockchains inheriting security from VeriBlock in order to extend these same benefits to them. The consensus algorithm of the VeriBlock blockchain is designed to simultaneously resist a variety of attacks including censorship by Bitcoin PoW miners, standard 51% attacks, and ambiguous disconnected chain publications.

## 2 VeriBlock and Proof-of-Proof Introduction

The Proof-of-Proof consensus protocol allows one blockchain to inherit the full Proof-of-Work security of another blockchain in an entirely decentralized, trustless, transparent, and permissionless manner. In the VeriBlock ecosystem, VeriBlock secures to Bitcoin using PoP, and altchains use PoP to secure themselves to VeriBlock (and by extension, Bitcoin).

### 2.1 VeriBlock and Proof-of-Proof At-A-Glance

PoP works by incentivizing the publication of a Security-Inheriting (SI) blockchain's current state to a Security-Providing (SP) blockchain, and then referencing those publications in the future when an alternate fork of the SI blockchain is proposed to the SI blockchain network.

Rules regarding the weighting and validity requirements (see chapter 6) of the publications force an attacker to announce the potential attack of the SI blockchain to the SP blockchain within a particular timeframe from the first challenged block in the attack. The absence of challenge publications after this timeframe (or the abandonment of a potential alternate chain due to a lack of future publications) marks all transactions in the block in question (as well as all previous blocks) with SP-finality, meaning an attacker would have to 51% attack the SP blockchain to insert fingerprints of the attacking SI chain sufficiently high in the SP blockchain to trigger a SI reorganization.

In order to be able to use PoP to secure itself to Bitcoin, the VeriBlock blockchain (the SI blockchain to Bitcoin, and the SP blockchain to altchains) itself has made a large number of design decisions that differ from those of other blockchains. Additionally, the architecture of the VeriBlock blockchain makes it the most cost-effective, easy-to-use, and secure way for an altchain to inherit Bitcoin's PoW security while imposing no limitations on the altchain's structure, consensus algorithm, etc. and requiring minimal changes to the altchain's codebase.

The particular implementation details and blockchain architecture decisions of the VeriBlock blockchain (see chapters 7 and 8) enable it to continue to function optimally in a variety of adversarial conditions including censorship by minority Bitcoin miners, highly volatile Bitcoin fee markets, and rapid (malicious or non-malicious) increases and decreases in VeriBlock and Bitcoin mining power. Additionally, VeriBlock can continue to inherit and extend Bitcoin's full PoW security even in times of infrequent publication to Bitcoin, and its incentivization structure is implemented in such a way as to economically disincentivize VeriBlock and Bitcoin PoW miners from censoring PoP transactions, even if they themselves are competing in the PoP mining process.

Similarly, the recommendations for how altchains should leverage VeriBlock for security carry the same resistance to adversarial conditions to the altchain network.

## 2.2 Proof-of-Proof: End-User Perspective

Proof-of-Proof prevents double-spending by ensuring full-visibility of the security level of particular SI block (and thus, all transactions within) at any particular point in time. “Early Attack Detection” (EAD) uses publications of SI state data to the SP chain to determine these security levels. For example, the lifecycle of a transaction on the VeriBlock blockchain (secured with PoP to Bitcoin) looks like this:

| VeriBlock Transaction Lifecycle |   |
|---------------------------------|---|
| Stage                           | Description   |
| UNCONFIRMED                     | The transaction is in the VeriBlock blockchain mempool  |
| N-CONFIRMED                     | The transaction is confirmed by N VeriBlock blocks  |
| N-BTC-REFERENCES                | The transaction is in a VeriBlock block which was referenced N Bitcoin blocks ago, and early attack detection metrics can begin to function |
| N-BTC-FINALITY                  | The transaction has achieved Bitcoin finality; and N BTC blocks would need to be forked to challenge it                                     |

For an altchain using PoP through VeriBlock, the transaction lifecycle would look like this:

| Altchain Transaction Lifecycle |   |
|--------------------------------|---|
| Stage                          | Description   |
| UNCONFIRMED                    | The transaction is in the altchain blockchain mempool   |
| N-CONFIRMED                    | The transaction is confirmed by N altchain blocks   |
| N-VBK-REFERENCES               | The transaction is in an altchain block which was referenced N VeriBlock blocks ago, and early attack detection metrics can begin to function |
| N-VBK-FINALITY                 | The transaction has achieved VeriBlock finality; and N VBK blocks would need to be forked to challenge it                                     |
| N-BTC-REFERENCES               | The VeriBlock block which provided VeriBlock finality to the transaction was referenced N Bitcoin blocks ago                                  |
| N-BTC-FINALITY                 | The transaction has achieved Bitcoin finality; and N BTC blocks would need to be forked to challenge it                                       |

Depending on the properties of the particular altchain (its block time, expected block time creation time variance, etc.) the “N-VBK-FINALITY” may occur after “N-BTC-REFERENCES” occurs, but will always occur before N-BTC-FINALITY if altchain security parameters are chosen correctly.

The integration of VeriBlock Proof-of-Proof into the altchain allows each of the altchain's users to decide on what level of transaction security they are comfortable with for a particular transaction, and to get real-time information about whether an incoming transaction has reached that level of security yet.

Most merchants, exchanges, and other users working with large or numerous high-risk altchain transactions will wait for some N-BTC-FINALITY security level (such as 3-BTC-FINALITY, meaning 3 BTC blocks would have to be forked to challenge the transaction). Users who are comfortable with the security of the altchain's short-term consensus protocol (PoW/PoC/PoS/DPoS/etc.) itself can operate in the same fashion as they would without PoP, and users with moderate risk profiles may opt to accept transactions at the VeriBlock-finality level if VeriBlock's native PoW is deemed sufficient protection.

It should be noted that even if N BTC blocks were to be forked, an attacker would also have to simultaneously fork a large number of VeriBlock blocks and a large number of altchain blocks to successfully perform a double-spend, *meaning the altchain transaction is actually more secure against double-spends than a transaction on Bitcoin with N confirmations.*

### 3 Desirable Miner Behaviors

At a high level, an SI blockchain uses an intermediate consensus protocol (PoW/PoC/PoS/DPoS, etc.) to construct blocks and maintain intermediate consensus, and incentivizes Proof-of-Proof miners to periodically publish fingerprints of the current state of the SI chain to the SP chain, which are referenced in the event of an SI fork to determine the legitimate chain.

The reward protocol of the SI chain is designed to incentivize three behaviors which benefit the system's intended functionality:

- SI PoP miners should publish PoP data to the first possible SP block and return the proofs of publication to the SI chain as soon as they are available
- SI block (PoW/PoC/PoS/DPoS, etc.) miners should include all available PoP transactions in their blocks
- SP PoW miners should include all fee-market-competitive SI PoP transactions in their blocks

In the special case of VeriBlock, VeriBlock PoP miners publish data to Bitcoin, altchain PoP miners publish data to VeriBlock, and VeriBlock PoW miners create blocks which include VeriBlock-to-Bitcoin PoP transactions and normal VeriBlock transactions (including those made by altchain PoP miners).

#### 3.1 SI PoP Miners

A SI PoP miner takes data from the SI chain (such as the block header of the most recent block, along with data to identify the PoP miner for payment), embeds that data in a SP transaction, submits that SP transaction to the SP network, waits for it to be included in a SP block, constructs an SPV-like proof that the transaction exists in the SP block (including any additional SP headers required to demonstrate that the block itself is a valid block in the SP chain), packages this proof in a PoP transaction, and submits this transaction to the SI network in order to inform the network about the completed publication.

PoP miners compete by trying to pay the lowest possible SP transaction fees while earning the highest possible PoP miner payout. Lower fees have a potential for a higher profit, but also risk being paid less (or not being paid at all) because of their delayed incorporation into the SP blockchain.

The ideal PoP miner:

- Publishes data regarding a SI block from the most recent keystone period (see section 6.2)
- Publishes this data in a SP transaction which makes it into the next SP block
- Returns a proof of publication to the SI chain as soon as it is available

### **3.2 SI Block Miners**

A SI block miner bundles transactions on the SI blockchain network (including returning PoP receipt transactions from SI PoP miners) and does some form of mining (PoW/PoC/PoS/DPoS, etc.) to create SI blocks and provide intermediate consensus to the SI chain.

The ideal block miner:

- Mines on top of the latest SI block
- Includes as many available SI PoP transactions as possible in their block
- Includes as many regular transactions as possible/allowed in their block, giving priority to transactions with higher per-byte fees

### **3.3 SP PoW Miners**

A SP PoW miner bundles transactions on the SP blockchain network (including “publication” transactions from SI PoP miners) and does Proof-of-Work mining to create SP blocks, providing SP-level security to the SI blocks which are published, directly or indirectly, in the produced SP block.

The ideal SP PoW Miner:

- Includes SP transactions based on their competitiveness in the normal SP fee market
- Doesn't censor SI PoP transactions (putting in only their own transaction to claim a large reward), even if the SP PoW miner is also a participant in the SI PoP mining process

## 4 Modeling Block Time Probabilities

The expected block times of Bitcoin, VeriBlock, and altchain blockchains can be modeled as exponential distributions. Understanding the distribution of these block times allows for the proper determination of reward and consensus finality intervals which maximize the security profile of blockchains adopting PoP, including VeriBlock itself.

### 4.1 Expected Bitcoin Block Times

The probability of a Bitcoin block taking more than  $x$  seconds, with a target blocktime of 600 seconds (or 10 minutes), can be modeled with an exponential formula:

$$p = e^{\frac{-x}{600}}$$

The following table illustrates these probabilities:

| Probability | Block Takes Longer Than: |           | Probability | Block Takes Longer Than: |           |
|-------------|--------------------------|-----------|-------------|--------------------------|-----------|
|             | (seconds)                | (minutes) |             | (seconds)                | (minutes) |
| 1.000       | 0.0000                   | 0.0000    | 0.500       | 415.8884                 | 6.9315    |
| 0.999       | 0.6003                   | 0.0100    | 0.450       | 479.1047                 | 7.9851    |
| 0.995       | 3.0075                   | 0.0501    | 0.400       | 549.7745                 | 9.1629    |
| 0.990       | 6.0303                   | 0.1001    | 0.350       | 629.8933                 | 10.4982   |
| 0.980       | 12.1217                  | 0.2020    | 0.300       | 722.3837                 | 12.0397   |
| 0.970       | 18.2756                  | 0.3046    | 0.250       | 831.7767                 | 13.8629   |
| 0.960       | 24.4932                  | 0.4082    | 0.200       | 965.6628                 | 16.0944   |
| 0.950       | 30.7760                  | 0.5129    | 0.180       | 1028.8791                | 17.1480   |
| 0.940       | 37.1253                  | 0.6188    | 0.160       | 1099.5489                | 18.3258   |
| 0.920       | 50.0290                  | 0.8388    | 0.140       | 1179.6678                | 19.6611   |
| 0.900       | 64.2164                  | 1.0536    | 0.120       | 1272.1582                | 21.2026   |
| 0.880       | 76.7001                  | 1.2783    | 0.100       | 1381.5511                | 23.0259   |
| 0.860       | 90.4938                  | 1.5082    | 0.080       | 1515.4372                | 25.2573   |
| 0.840       | 104.6121                 | 1.7435    | 0.060       | 1688.0465                | 28.1341   |
| 0.820       | 119.0706                 | 1.9845    | 0.050       | 1797.4394                | 29.9573   |
| 0.800       | 133.8862                 | 2.2314    | 0.040       | 1931.3255                | 32.1888   |
| 0.750       | 172.6093                 | 2.8768    | 0.030       | 2103.9348                | 35.0656   |
| 0.700       | 214.0050                 | 3.5668    | 0.020       | 2347.2139                | 39.1202   |
| 0.650       | 258.4698                 | 4.3078    | 0.010       | 2763.1022                | 46.0517   |
| 0.600       | 306.4954                 | 5.1083    | 0.005       | 3178.9904                | 52.9832   |
| 0.550       | 358.7023                 | 5.9784    | 0.001       | 4144.6532                | 69.0776   |

Additionally, the probability that exactly  $y$  Bitcoin blocks occur in a period of  $x$  seconds can be modeled with a Poisson distribution:

$$p = \frac{\left(\frac{x}{600}\right)^y e^{\left(\frac{-x}{600}\right)}}{y!}$$

## 4.2 Expected VeriBlock Block Times

The probability of a VeriBlock block taking more than  $x$  seconds, with a target blocktime of 30 seconds (which was chosen for reasons detailed in section 7.6) can be modeled with an exponential formula:

$$p = e^{\frac{-x}{30}}$$

The following table illustrates these probabilities:

| Probability | Block Takes Longer Than: |           |
|-------------|--------------------------|-----------|
|             | (seconds)                | (minutes) |
| 1.000       | 0.0000                   | 0.0000    |
| 0.999       | 0.0300                   | 0.0005    |
| 0.995       | 0.1504                   | 0.0025    |
| 0.990       | 0.3015                   | 0.0050    |
| 0.980       | 0.6061                   | 0.0101    |
| 0.970       | 0.9138                   | 0.0152    |
| 0.960       | 1.2247                   | 0.0204    |
| 0.950       | 1.5388                   | 0.0256    |
| 0.940       | 1.8563                   | 0.0309    |
| 0.920       | 2.5015                   | 0.0417    |
| 0.900       | 3.1608                   | 0.0527    |
| 0.880       | 3.8360                   | 0.0639    |
| 0.860       | 4.5247                   | 0.0754    |
| 0.840       | 5.2306                   | 0.0872    |
| 0.820       | 5.9535                   | 0.0992    |
| 0.800       | 6.6943                   | 0.1116    |
| 0.750       | 8.6305                   | 0.1438    |
| 0.700       | 10.7002                  | 0.1783    |
| 0.650       | 12.9235                  | 0.2154    |
| 0.600       | 15.3248                  | 0.2554    |
| 0.550       | 17.9351                  | 0.2989    |

| Probability | Block Takes Longer Than: |           |
|-------------|--------------------------|-----------|
|             | (seconds)                | (minutes) |
| 0.500       | 20.7944                  | 0.3466    |
| 0.450       | 23.9552                  | 0.3993    |
| 0.400       | 27.4887                  | 0.4581    |
| 0.350       | 31.4947                  | 0.5249    |
| 0.300       | 36.1192                  | 0.6020    |
| 0.250       | 41.5888                  | 0.6931    |
| 0.200       | 48.2831                  | 0.8047    |
| 0.180       | 51.4440                  | 0.8574    |
| 0.160       | 54.9774                  | 0.9163    |
| 0.140       | 58.9834                  | 0.9831    |
| 0.120       | 63.6079                  | 1.0601    |
| 0.100       | 69.0776                  | 1.1513    |
| 0.080       | 75.7719                  | 1.2629    |
| 0.060       | 84.4023                  | 1.4067    |
| 0.050       | 89.8720                  | 1.4979    |
| 0.040       | 96.5663                  | 1.6095    |
| 0.030       | 105.1970                 | 1.7533    |
| 0.020       | 117.3610                 | 1.9560    |
| 0.010       | 138.1550                 | 2.3026    |
| 0.005       | 158.9500                 | 2.6492    |
| 0.001       | 207.2330                 | 3.4539    |

Additionally, the probability that exactly  $y$  VeriBlock blocks occur in a period of  $x$  seconds can be modeled with a Poisson distribution:

$$p = \frac{\left(\frac{x}{30}\right)^y e^{\left(\frac{-x}{30}\right)}}{y!}$$

### 4.3 Expected Relative Bitcoin Block per VeriBlock Block Occurrences

As both the creation of Bitcoin and VeriBlock blocks exist as Poisson processes, the probability  $p$  that  $n$  Bitcoin blocks occur before  $m$  VeriBlock blocks (assuming the difficulty of both networks matches their current hashrate) can be modeled as:

$$p = \sum_{k=n}^{n+m-1} \binom{n+m-1}{k} \left( \frac{\frac{1}{600}}{\frac{1}{600} + \frac{1}{30}} \right)^k \left( \frac{\frac{1}{30}}{\frac{1}{600} + \frac{1}{30}} \right)^{n+m-1-k}$$

Or more simply:

$$p = \sum_{k=n}^{n+m-1} \binom{n+m-1}{k} \left( \frac{1}{21} \right)^k \left( \frac{20}{21} \right)^{n+m-1-k}$$

To find only the probability that  $n$  Bitcoin blocks occur before a single VeriBlock block, a simpler equation can be used:

$$p = \left( \frac{\frac{1}{600}}{\frac{1}{600} + \frac{1}{30}} \right)^n = \left( \frac{1}{21} \right)^n$$

To illustrate these probabilities:

| Probability | $n$ Bitcoin Blocks | Before $m$ VeriBlock Blocks |
|-------------|--------------------|-----------------------------|
| 0.0476      | 1                  | 1                           |
| 0.0023      | 2                  | 1                           |
| 0.0001      | 3                  | 1                           |
| 0.0000      | 4                  | 1                           |
| 0.0930      | 1                  | 2                           |
| 0.0066      | 2                  | 2                           |
| 0.0004      | 3                  | 2                           |
| 0.0000      | 4                  | 2                           |
| 0.1362      | 1                  | 3                           |
| 0.0128      | 2                  | 3                           |
| 0.0010      | 3                  | 3                           |
| 0.0001      | 4                  | 3                           |
| 0.1773      | 1                  | 4                           |
| 0.0206      | 2                  | 4                           |
| 0.0019      | 3                  | 4                           |
| 0.0002      | 4                  | 4                           |
| 0.2538      | 1                  | 6                           |
| 0.0406      | 2                  | 6                           |

| Probability | $n$ Bitcoin Blocks | Before $m$ VeriBlock Blocks |
|-------------|--------------------|-----------------------------|
| 0.0050      | 3                  | 6                           |
| 0.0005      | 4                  | 6                           |
| 0.3861      | 1                  | 10                          |
| 0.0937      | 2                  | 10                          |
| 0.0172      | 3                  | 10                          |
| 0.0026      | 4                  | 10                          |
| 0.5190      | 1                  | 15                          |
| 0.1754      | 2                  | 15                          |
| 0.0445      | 3                  | 15                          |
| 0.0092      | 4                  | 15                          |
| 0.6231      | 1                  | 20                          |
| 0.2642      | 2                  | 20                          |
| 0.0847      | 3                  | 20                          |
| 0.0220      | 4                  | 20                          |
| 0.0049      | 5                  | 20                          |
| 0.0009      | 6                  | 20                          |
| 0.0002      | 7                  | 20                          |
| 0.0000      | 8                  | 20                          |

#### 4.4 Expected Relative VeriBlock Block per Bitcoin Block Occurrences

As both the creation of Bitcoin and VeriBlock blocks exist as Poisson processes, the probability  $p$  that  $n$  VeriBlock blocks occur before  $m$  Bitcoin blocks (assuming the difficulty of both networks matches their current hashrate) can be modeled as:

$$p = \sum_{k=n}^{n+m-1} \binom{n+m-1}{k} \left( \frac{\frac{1}{30}}{\frac{1}{30} + \frac{1}{600}} \right)^k \left( \frac{\frac{1}{600}}{\frac{1}{30} + \frac{1}{600}} \right)^{n+m-1-k}$$

Or more simply:

$$p = \sum_{k=n}^{n+m-1} \binom{n+m-1}{k} \left( \frac{20}{21} \right)^k \left( \frac{1}{21} \right)^{n+m-1-k}$$

To find only the probability that  $n$  VeriBlock blocks occur before a single Bitcoin block, a simpler equation can be used:

$$p = \left( \frac{\frac{1}{30}}{\frac{1}{30} + \frac{1}{600}} \right)^n = \left( \frac{20}{21} \right)^n$$

To illustrate these probabilities:

| Probability | $n$ VeriBlock Blocks | Before $m$ Bitcoin Blocks |
|-------------|----------------------|---------------------------|
| 0.9524      | 1                    | 1                         |
| 0.9070      | 2                    | 1                         |
| 0.8227      | 4                    | 1                         |
| 0.6768      | 8                    | 1                         |
| 0.4810      | 15                   | 1                         |
| 0.3769      | 20                   | 1                         |
| 0.2314      | 30                   | 1                         |
| 0.0872      | 50                   | 1                         |
| 0.0076      | 100                  | 1                         |
| 0.9977      | 1                    | 2                         |
| 0.9934      | 2                    | 2                         |
| 0.9794      | 4                    | 2                         |
| 0.9347      | 8                    | 2                         |
| 0.8246      | 15                   | 2                         |
| 0.7358      | 20                   | 2                         |
| 0.5619      | 30                   | 2                         |
| 0.2948      | 50                   | 2                         |
| 0.0438      | 100                  | 2                         |

| Probability | $n$ VeriBlock Blocks | Before $m$ Bitcoin Blocks |
|-------------|----------------------|---------------------------|
| 0.9999      | 1                    | 3                         |
| 0.9996      | 2                    | 3                         |
| 0.9980      | 4                    | 3                         |
| 0.9899      | 8                    | 3                         |
| 0.9555      | 15                   | 3                         |
| 0.9153      | 20                   | 3                         |
| 0.8059      | 30                   | 3                         |
| 0.5470      | 50                   | 3                         |
| 0.1309      | 100                  | 3                         |
| 0.9999      | 1                    | 4                         |
| 0.9999      | 2                    | 4                         |
| 0.9998      | 4                    | 4                         |
| 0.9987      | 8                    | 4                         |
| 0.9908      | 15                   | 4                         |
| 0.9780      | 20                   | 4                         |
| 0.9298      | 30                   | 4                         |
| 0.7551      | 50                   | 4                         |
| 0.2719      | 100                  | 4                         |

#### 4.5 Expected Relative Altchain Block per VeriBlock Block Occurrences

As both the creation of (most types of) Altchain blocks and all VeriBlock blocks exist as Poisson processes, the probability  $p$  that  $n$  Altchain blocks with a target time of  $t$  occur before  $m$  VeriBlock blocks (assuming the difficulty of both networks matches their current hashrate) can be modeled as:

$$p = \sum_{k=n}^{n+m-1} \binom{n+m-1}{k} \left( \frac{\frac{1}{t}}{\frac{1}{t} + \frac{1}{30}} \right)^k \left( \frac{\frac{1}{30}}{\frac{1}{t} + \frac{1}{30}} \right)^{n+m-1-k}$$

To find only the probability that  $n$  Altchain blocks occur before a single VeriBlock block, a simpler equation can be used:

$$p = \left( \frac{\frac{1}{t}}{\frac{1}{t} + \frac{1}{30}} \right)^n$$

## 5 PoP vs PoW Mining Market Comparison

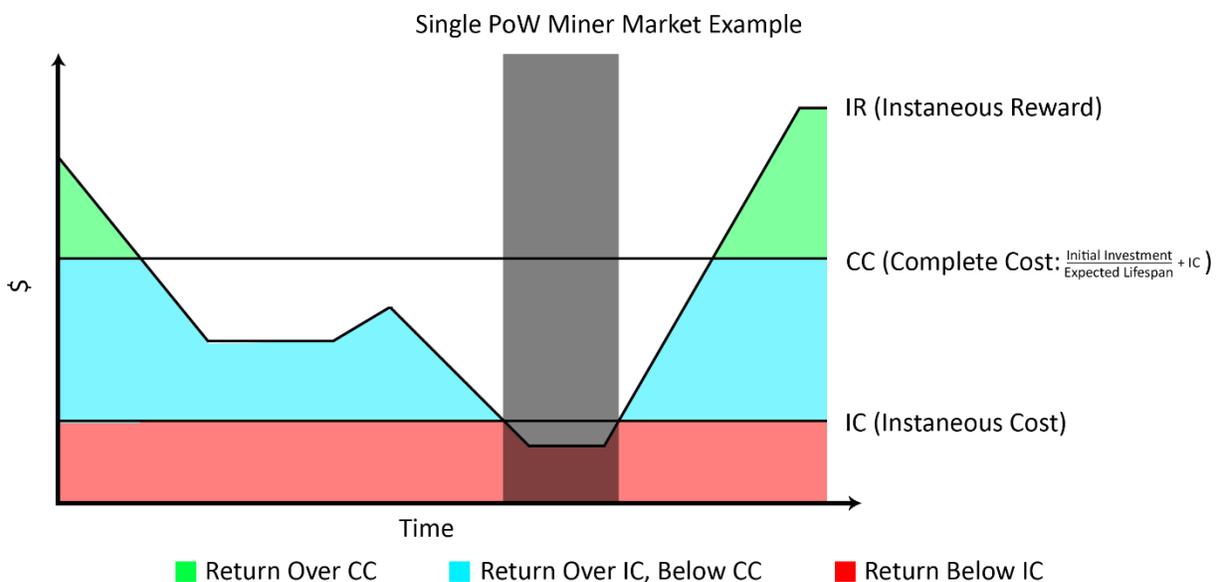
While the incentivization mechanism of PoP is similar to that of PoW, the instantaneous nature of the mining market must be taken into account to optimize the security profile PoP can provide.

Generally, PoW (Proof-of-Work) miners make medium- or long-term investments (directly or otherwise) into mining hardware. As a result, they experience and respond to two fundamentally different costs: the initial costs of hardware purchase, and the instantaneous (or in the case of most large-scale PoW mining operations, semi-instantaneous) costs of hardware operation.

In contrast, PoP miners experience a market which highly resembles that of only the instantaneous costs of PoW miner hardware operation without the initial investment into any sort of mining hardware, and where electrical/cooling/etc. costs are roughly the same for all participants.

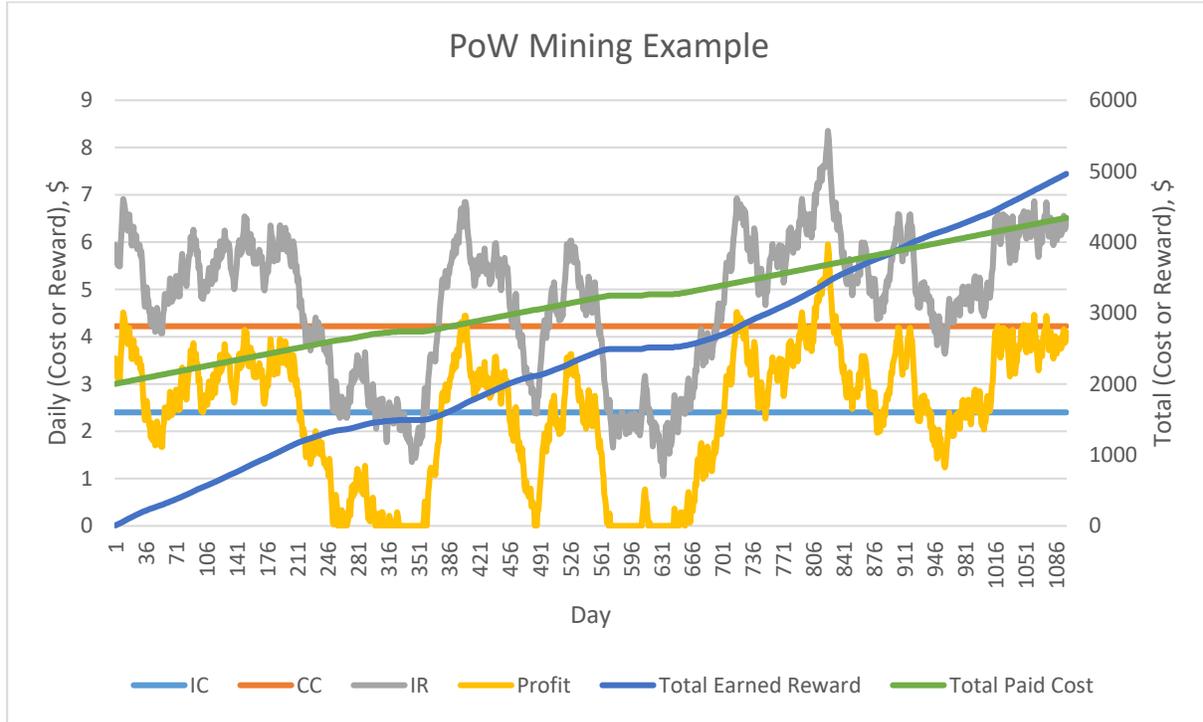
### 5.1 PoW Reward Market

Despite the initial costs of hardware purchase being comparatively large, the instantaneous costs of hardware operation over a period in which a PoW miner can reasonably respond to changes in the PoW mining market (on the order of minutes to days) are highly subject to instantaneous profitability; if a PoW miner spends  $\frac{x}{day}$  and receives  $\frac{y}{day}$  as a result, the PoW miner is expected to only continue operations if  $x > y$ .



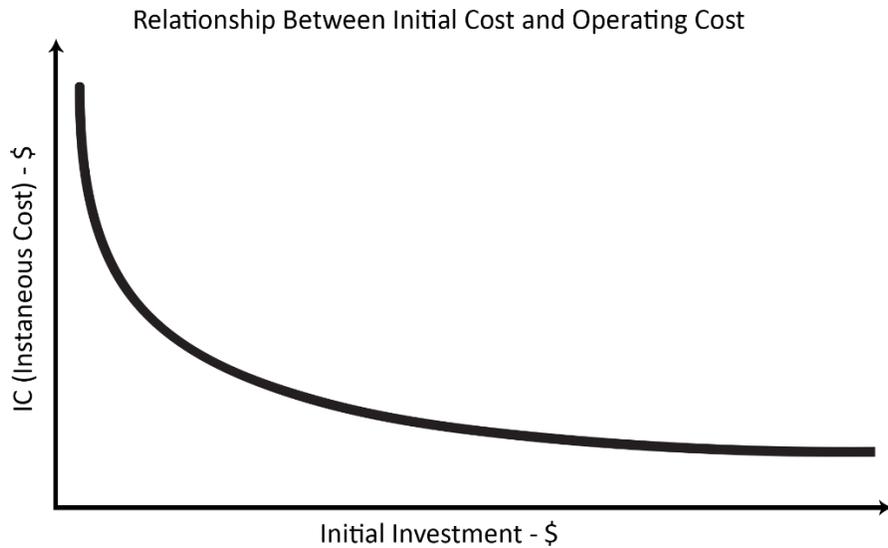
In the above graph, the PoW miner in question is expected to mine at all times except the grey-shaded region, as  $IR \geq IC$  (miner covers the cost to operate their equipment).

It should also be noted that IC is shown as a flat line, although IC can fluctuate over time (not needing to run cooling as much at night, electrical costs fluctuating based on time of day, etc.)

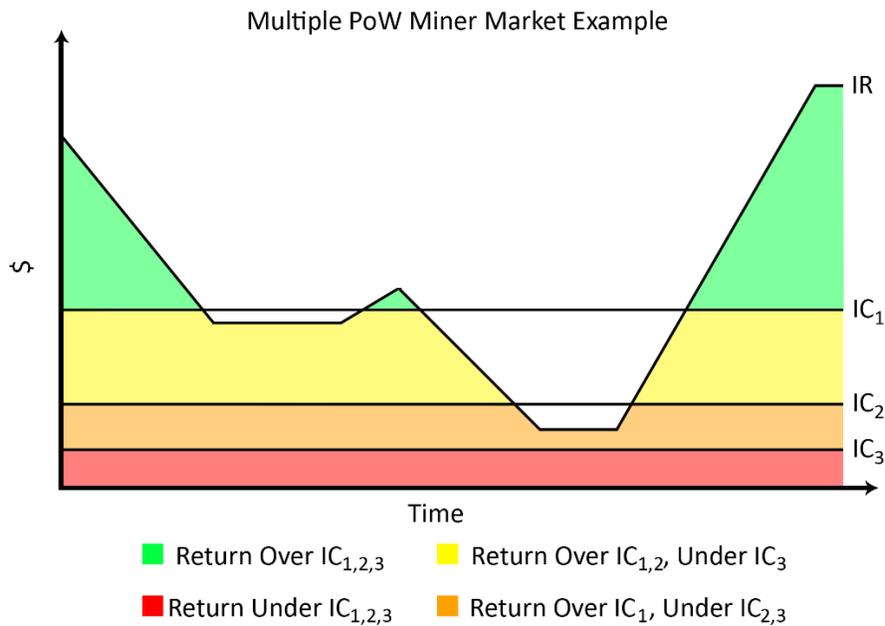


In the above graph (with profitability, operation timespan, IR jitter, etc. synthetically selected to demonstrate key behaviors in a normal single PoW miner’s cost-reward-market response), even when the instantaneous reward drops below CC (over an expected viability period of three years), mining continues (only ceasing when IR drops below IC).

PoW miners must choose a point on the  $\frac{initial\_cost}{operation\_cost}$  curve, where a higher initial\_cost generally results in a lower operation\_cost (higher-density and/or higher-power-efficiency), and, similarly, a lower initial\_cost generally results in a higher operation\_cost:



This (in addition to varying electrical, labor, etc. costs in different regions) results in many PoW miners having different instantaneous costs (IC) from each other, enabling some PoW miners to remain above operational break-even while other PoW miners fall below operational break-even (which reduces competition, causing IR to correct upwards):



Note that in the above graph, there is no period during which no PoW mining occurs (when  $IR > IC_1$ , all three PoW miners are active, when  $IC_1 > IR > IC_2$ , two out of three PoW miner are active, and when  $IC_2 > IR > IC_3$ , one out of three PoW miners are still active. When IR drops below  $IC_1$ , the PoW miner using IC curve  $IC_1$  will leave, allowing the PoW

difficulty to adjust (over time) to pay out a higher reward per hash computed, etc. Only extremely rapid changes in IR will result in no miners being profitable to operate.

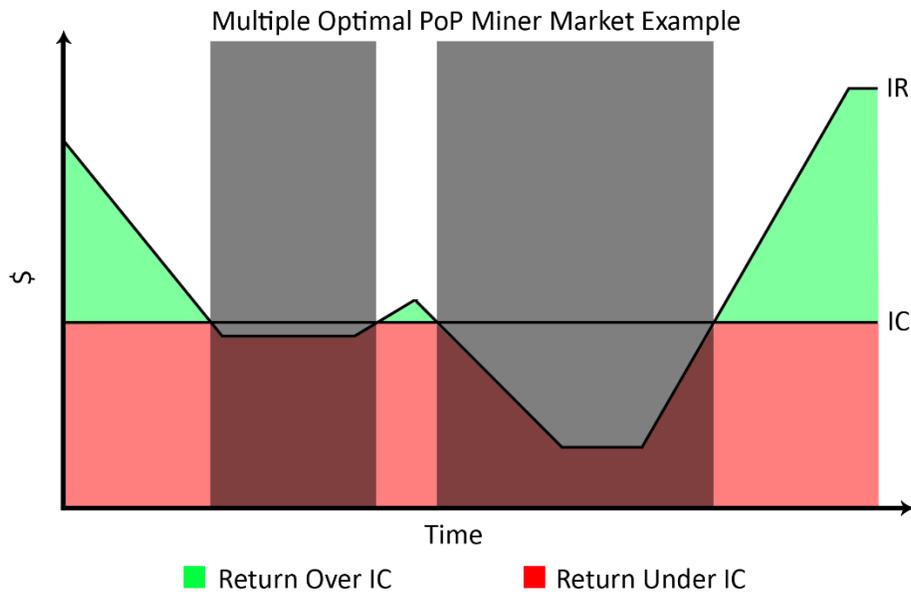
It should be noted that this simplified market approach does not factor in a PoW miner leaving when  $IR > IC$  in the event that the local IR is smaller than another external IR (such as mining a different blockchain), however such an exodus of mining power has the same result on the market: IR will adjust accordingly, offering additional incentive for those miners who continue to mine the blockchain in question). A more general approach would introduce another value (“Opportunity Cost”, or “OC”), and assume a PoW miner will leave when IR drops below the highest of either OC or IC.

As the result of commitments for short-term and semi-short-term costs (minimum electrical consumption contracts, employee compensation, floorspace rental, etc.), most large-scale PoW miners are subject to an alternative ‘semi-instantaneous’ cost market, where one or more interval costs compound into a cost curve which substitutes for the normal instantaneous cost market which small-scale PoW miners experience. The detailed exploration of such a semi-instantaneous cost market for PoW mining is beyond the scope of this document, as the optimal behavior of large-scale PoW mining operations (continuing operation when profitable) is even less similar to the instantaneous cost market as small-scale PoW mining, to which we are contrasting the nearly-completely-instantaneously-cost-market-based PoP mining.

## 5.2 PoP Reward Market

The PoP reward market is distinctly different than the PoW reward market; PoP miners invest an immediate cost—the price (transaction fee per byte) of a Bitcoin transaction—and receive a short-term return: PoP miners spend  $\frac{x}{PoP\ Publication}$  and receive  $\frac{y}{PoP\ Publication}$ . Therefore, their mining behavior is expected to closely model that of a PoW miner’s instantaneous profitability; a PoP miner is expected to only mine whenever  $x > y$ .

Because there is no initial cost, there is no initial-cost-vs-operating-cost relationship. As a result, the “most efficient” or most economically-viable PoP miners will all have an identical (for practical purposes) IC, and therefore will generally be expected to all start or stop mining simultaneously:



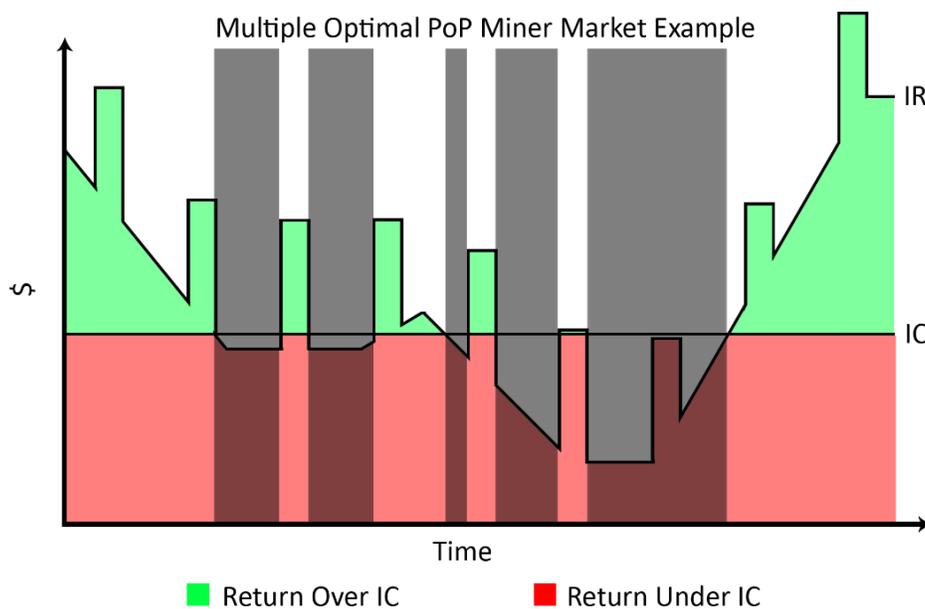
In the above graph, all optimal PoP miners share IC and IR (ignoring external incentives like deriving additional benefit from PoP mining at a particular time to protect their own transactions), so PoP mining can't expect to rely on some PoP miners ceasing mining before others (as is regular for PoW). The grey-shaded areas above demonstrate the times during which PoP mining is expected to temporarily cease. For illustration purposes IC is represented as a flat line, although changes in the price of Bitcoin and the fee level will result in changes to IR.

It should be noted that this simplification does not account for proprietary algorithms for predicting optimal Bitcoin transaction fees; it is assumed that optimal algorithms will become public for determining the minimum required Bitcoin (or alternate security-providing blockchain) transaction fee per byte (or per weight unit in the case of Segwit-enabled transactions) to maximize PoP rewards given the probabilities of inclusion in the next  $n$  Bitcoin blocks and the reward curve for compensation of the PoP publication's inclusion in the next  $n$  Bitcoin blocks. It should also be noted that it is assumed that Bitcoin blocks are full (thereby ensuring that PoW miners of Bitcoin don't experience a (practically) zero fee for including their own PoP publications), or that the minimum relay fee is zero or nearly zero. It should be noted that empty free block space would always be used up by PoP miners (who would begin to competitively bid with each other) as long as  $y > 0$ .

PoP also has a different security desire than PoW; while PoW generally aims to maximize the total reward paid in a particular short-term interval (maximize the total number of hashes performed on behalf of the blockchain in that time), PoP instead aims to minimize the length of periods of time during which PoP mining ceases to occur. While the VeriBlock blockchain's implementation of PoP (and PoP implementation suggestions for altchains) is resistant to multi-block publication gaps (see section 6.2 regarding keystones), these publications must occur at a certain frequency to maintain full security of the protocol. As a result, PoP gains a

higher benefit from the consistency of publications in a period of time, rather than the total quantity of publications.

In order to minimize the length (in blocks, or 'time') of each period where  $IR < IC$ , an artificial "jittering" mechanism is introduced which causes IR to "jitter" more than is simply caused by changes in the security-providing (Bitcoin, in the case of VeriBlock) blockchain fee market, fluctuations in the exchange rate between the native coin of the SI chain and the native coin of the SP chain, and the adjustment over time of the "PoP Difficulty" (which alters the expected reward per PoP based on recent historical data of the quantity of PoP publications in a similar fashion to how PoW difficulty alters the difficulty of finding a valid solution [or more simply for our purposes, the expected reward per hash] to the PoW challenge).



The above graph demonstrates the value of IR jitters in reducing the width of PoP unprofitability time periods (shaded grey), which provides PoW security inheritance with shorter maximum and average dark periods.

Different-sized jitters are introduced at varying intervals, which further ensure that periods of particularly low IR relative to IC will still often experience brief bursts of PoP mining to further increase the average timeliness of PoP publications. The details of the introduced jitter for VeriBlock, as well as suggestions for altchain PoP implementations, are explored in section 6.6.

## 6 Proof-of-Proof Technical Overview

At a high level, Proof-of-Proof involves incentivizing a new form of miner—a PoP miner—to publish data from the SI blockchain to the SP blockchain for the SI blockchain to reference in the future in the event of an alternate proposed history.

### 6.1 Desirable Proof-of-Proof Protocol Traits

Proof-of-Proof is designed to exhibit the following desirable traits:

1. The failure of one or several consecutive SI blocks to get published in the SP blockchain should not compromise the security of the SI blockchain
2. An attacker generating an alternate history fork of the SI blockchain should be required to regularly announce the state of their chain to the SP blockchain in a timely fashion
3. An attacker should not be able to “purchase” a higher rating for their competing SI fork than the legitimate SI fork receives based on publications in the SP blockchain (ex: the number of PoP publications of a particular SI block should not be a multiplier for the PoP score of that block)
4. The fork resolution algorithm (means by which the SI chain resolves consensus and determines the legitimacy of a proposed fork based on publications on the SP chain) should eventually and predictably reach finality, where the lack of publications of a competing SI chain up to SI height ‘n’ in the SP blockchain ensures that the SI chain cannot be forked back before ‘n’ without the attacker being required to reorganize the SP (and by extension, any other upstream SP) as well
5. The fork resolution algorithm should introduce limited overhead for SPV clients, and should ensure that an SPV client can maintain full security knowledge if connected to at least one non-byzantine node
6. The reward scheme for PoP miners should not incentivize SI (PoW, PoC, PoS, etc.) or SP PoW miners to censor PoP miner activity, even if they themselves are also engaged in the PoP mining process

Note that the PoP publications of an SI chain should contain the chain’s “Proof-of-X” in a self-contained fashion (for example, a PoW blockchain will publish an entire block header, not just a block hash, a PoS blockchain will publish a proof of funds or proof of validator set participation depending on the flavor of PoS, ...). This allows the difficulty of the Proof-of-X algorithm to act as an anti-spam mechanism and prevent a malicious party from publishing “spam” PoP transactions that could artificially trigger early-attack-detection metrics and lead users of the blockchain to falsely believe there may be a pending threat despite the attacker not actually producing the blocks they are announcing. See chapter 10 for additional countermeasures to false EAD triggering by minority hashrate actors.

## 6.2 Keystoning

To prevent a lack of publication for several contiguous SI blocks from compromising the SI blockchain’s security, we introduce a form of block reference braiding referred to as “keystoning” into the SI blockchain, and introduce a need for publication continuity in the fork resolution algorithm (see sections 6.3 and 6.4).

In keystoning, “keystone” blocks occur at a regular interval (unless experimental dynamic keystoning is employed, see section 10.5), and act as the backbone of long-range consensus resolution. All blocks (including keystones themselves) reference a number of previous keystones, allowing the publication of one block’s state data (which includes its keystone references) to provide context for a large section of the SI blockchain.

Three keystone parameters dictate the properties of a blockchain’s keystoning: the keystone interval, the number of referenced keystones, and the keystone finality delay.

| Keystone Parameters          |  |                     |
|------------------------------|--|---------------------|
| Parameter                    | Description  | Useful Range        |
| keystone_interval (i)        | The interval between keystone blocks. For example, a blockchain with a keystone interval of 20 would have keystones {0, 20, 40, ...}. This addresses the desirable protocol trait #2   | $2 \leq i < \infty$ |
| num_referenced_keystones (r) | The number of previous keystones which each block references.  | $2 \leq r < \infty$ |
| keystone_finality_delay (d)  | The maximum number of SP blocks that can occur without a publication of an additional SI keystone before “continuity” is lost (future publications of that fork are not considered for fork resolution, further explained in section 6.4). This addresses the desirable protocol trait #4. | $2 \leq d < \infty$ |

**The selected keystone parameters have a significant effect on the SI blockchain’s security profile.** If a SI blockchain has a keystone\_interval  $i$  and num\_referenced\_keystones  $r$ , the maximum number of blocks that the SI blockchain can go without a publication to the SP blockchain is approximately  $i*r$  (depends on whether the most recently published block was a keystone, just after a keystone, or farther into the keystone period) to maintain security.

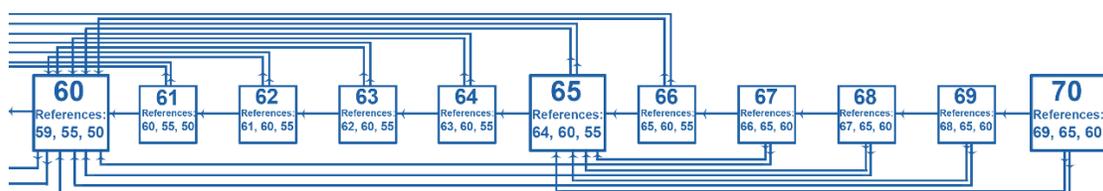
Given keystone\_interval  $i$  and num\_referenced\_keystones  $r$ , block references will follow the pattern (referencing the immediately preceding block, then  $r$  previous keystones):

| Keystoning Block Reference Pattern |   |
|------------------------------------|---|
| Block Number                       | Reference Pattern                                   |
| $ni$                               | $\{ni - 1,$<br>$(n - 1)i,$<br>...<br>$(n - r)i\}$   |
| $ni + 1$                           | $\{ni,$<br>$(n - 1)i,$<br>...<br>$(n - r + 1)i\}$   |
| $ni + m$ where $1 < m < ni$        | $\{ni + m - 1,$<br>$ni,$<br>...<br>$(n - r + 1)i\}$ |

For example, a blockchain with  $i=5$  and  $r=2$  would have the following keystone references:

| Keystoning Block Reference Pattern Example $i=5$ $r=2$ |                   |
|--|-------------------|
| Block Number   | Reference Pattern |
| 65   | 64, 60, 55        |
| 66   | 65, 60, 55        |
| 67   | 66, 65, 60        |
| 68   | 67, 65, 60        |
| 69   | 68, 65, 60        |
| 70   | 69, 65, 60        |
| 71   | 70, 65, 60        |
| 72   | 71, 70, 65        |
| 73   | 72, 70, 65        |

Or visually:



VeriBlock uses keystone parameters  $i=20$ ,  $r=2$ ,  $d=11$ . Note that these values *should not be used by an altchain; they're based on Bitcoin's (the SP blockchain for VeriBlock) properties. Because VeriBlock provides more storage space than Bitcoin does in its OP\_RETURN, it is reasonable to publish additional context headers and/or implement another proofing mechanism for preventing false Early Attack Detection [see chapter 10]]. The  $r=2$  value was chosen for VeriBlock due to the space constraints of OP\_RETURN on Bitcoin. There is a trade-off in the time until finality with longer keystone block reference patterns. Additionally, VeriBlock's 30-second blocktime must be taken into account.*

### 6.3 Reduced Publication View

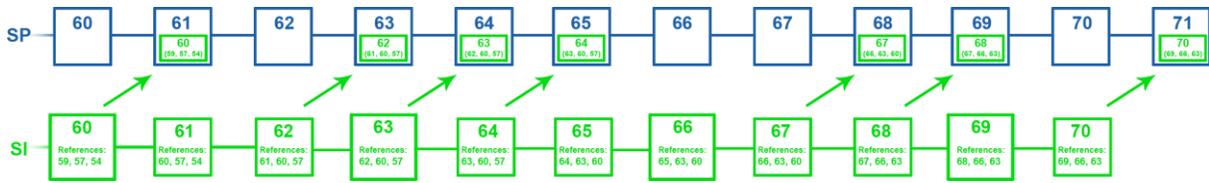
Based on the publications of the SI chain in the SP chain, a “reduced publication view” can be constructed with only the information relevant to consensus resolution.

The following algorithm is used for constructing a reduced publication view of a chain:

1. If calculating a reduced publication view in order to resolve a fork, ensure that the view of the SP blockchain accounts for knowledge contained in both forks (add all SP data from both chains to the SP SPV view, and remove any SP information not in the final main SI chain when finished with fork resolution)
2. Determine the last “stable” keystone (in the case of comparing two forks, this is the highest keystone shared by both chains)
3. For each keystone after the “stable” keystone:
  - a. gather all of the publications in the SI chain which connect said keystone to the previous keystone
  - b. Remove from the set of gathered publications any publications which are not in the best SP chain
  - c. Select the publication with the lowest SP block height (if multiple have the same SP block height, select one of them; the particular one selected is not relevant), this is the best publication for that particular keystone
  - d. Associate the keystone with the SP publication height of that best publication

Now the ordered list of associations of keystones with their lowest SP publication heights comprises the reduced publication view of that chain.

Take, for example, the following publication scenario (with SI  $i=3$ ,  $r=2$ ,  $d=(irrelevant)$  [arguments chosen for purposes of example; **not recommended**):



**Note that the keystone parameter  $i=3$  is unoptimal for most SI chains. Generally, shorter blocktimes will be accompanied by larger interval sizes.**

Constructing a reduced publication view of the above scenario would yield the following:

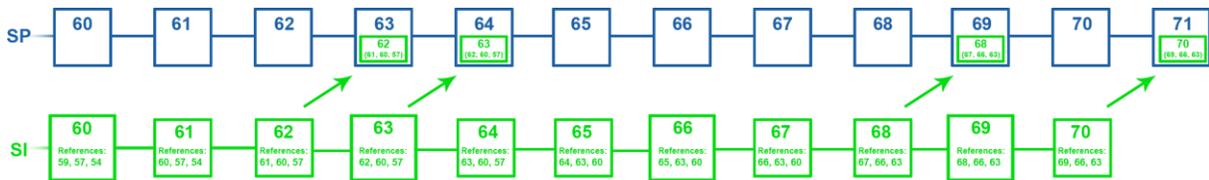


### 6.4 Publication Continuity Filtering

In order to force a potential attacker to give full public continuity visibility of their attacking chain while it's being built, the fork resolution algorithm (covered in 6.5) requires a chain to maintain publication continuity for its publications to be valid for fork resolution.

Continuity of an SI chain is achieved if the connectivity of every keystone in a chain can be resolved based purely on publications in the SP chain.

The example publication scenario in 6.3 can have a few publications removed while maintaining continuity. For example, the following publication scenario retains continuity:

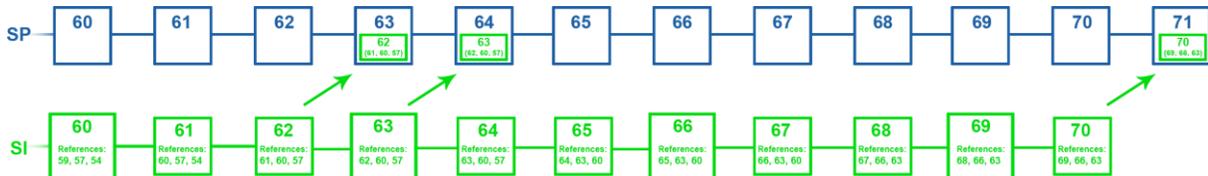


Based on publications in SP, SI keystone 60 is linked to 57 by publication of SI block 62's state info (in SP block 63), SI keystone 63 is linked to 60 by publication of SI block 63's state info (in SP block 64), SI keystone 66 is linked to 63 by publication of SI block 68's state info (in SP block 69), and SI keystone 69 is linked to 66 by publication of SI block 70's state info (in SP block 71). However, removing any of the publications displayed would cause SI to lose continuity. **Note: it is assumed that the blockchain's reward structure will incentivize publication beyond the "bare minimum" displayed above.**

The corresponding reduced publication view would be:



However, if a publication were to be removed, for example:



Then the reduced publication view would have a gap:



And the filtered reduced publication view would only consist of:



### 6.5 Fork Resolution

When a fork is proposed to a SI blockchain, it responds by comparing both potential forks (the proposed fork, and the existing chain) based on their publications to the SP blockchain. This comparison is based only on the timeliness and continuity of the publications, as the frequency/volume (beyond expected minimums) can be controlled by an attacker.

Fork resolution relies on a *fork\_resolution\_publication\_latency\_lookup\_table*, which will be specified for each SI blockchain based on the parameters of the particular SI and SP chains.

This lookup table should have a short plateau at the beginning (which allows blocks which were produced “around the same time” to share the same weighting, which prevents opportunistic attackers with minimal hashrate from attempting to generate the next keystone block slightly faster than the normal network and spam short reorgs), but then should aggressively weight keystone publications which fall outside of this grace period.

For example, an SI chain inheriting PoP security through VeriBlock might consider the following (where *amnesty\_period* represents the “grace period” of publications of the SI chain to VeriBlock, based on the SI difficulty adjustment algorithm, publication frequency, expected publication frequency, etc.):

| Example <i>fork_resolution_publication_latency_lookup_table</i>                                       |  |
|---|--|
| VBK Publication Timeliness  | Fork Resolution Weighting  |
| $0 \leq \textit{publication\_timeliness} < \textit{amnesty\_period}$                                  | 1  |
| $\textit{amnesty\_period} \leq \textit{publication\_timeliness} < \textit{keystone\_finality\_delay}$ | $\frac{1}{(\textit{publication\_timeliness} - \textit{amnesty\_period})^{1.03}}$ |
| $\textit{keystone\_finality\_delay} \leq \textit{publication\_timeliness}$                            | 0  |

The aggressiveness in the SI block production difficulty adjustment algorithm for preventing attackers with significantly more than 51% power from producing blocks significantly earlier than the “legitimate” network should be considered in determining how aggressive this lookup table is in lowering the score

The recommended PoP fork resolution algorithm consists of:

1. Process both forks (referred to as ‘left’ and ‘right’) into their filtered reduced publication view as explained in sections 6.3 and 6.4 (*sanity check: both filtered reduced publication views should start at the same keystone*)
2. Let *first\_ks* = the first keystone number of the filtered reduced publication views, and let *last\_ks* = the last keystone number of the longest filtered reduced publication view
3. Let *left\_fork\_score* and *right\_fork\_score* = 0
4. Let *continue\_scoring\_left\_fork* and *continue\_scoring\_right\_fork* = true
5. For each keystone *keystone\_for\_consideration* between *first\_ks* and *last\_ks*:
  - a. If *continue\_scoring\_left\_fork*, then let *left\_fork\_keystone\_publication* = the publication index in the SP blockchain for the *keystone\_for\_consideration* for the left chain (otherwise set it to a very high value, like 1,000,000,000). In the special case where the timestamp of the SP block the *keystone\_for\_consideration* is published to is lower than the timestamp of the *keystone\_for\_consideration*, then let *left\_fork\_keystone\_publication* = the first SP block which is both **higher than the block height of the first SP block**

**to contain a publication of the *keystone\_for\_consideration* and has a higher timestamp than that embedded in *keystone\_for\_consideration***

- b. If *continue\_scoring\_right\_fork*, then let *right\_fork\_keystone\_publication* = the publication index in the SP chain for the *keystone\_for\_consideration* for the right chain (otherwise set it to a very high value, like 1,000,000,000). In the special case where the timestamp of the SP block the *keystone\_for\_consideration* is published to is lower than the timestamp of the *keystone\_for\_consideration*, then let *right\_fork\_keystone\_publication* = the first SP block which is both **higher than the block height of the first SP block to contain a publication of the *keystone\_for\_consideration* and has a higher timestamp than that embedded in *keystone\_for\_consideration***
  - c. Let *lowest\_keystone\_publication* =  $\min(\text{left\_fork\_keystone\_publication}, \text{right\_fork\_keystone\_publication})$
  - d. Let *left\_fork\_keystone\_publication\_delay* =  $(\text{left\_fork\_keystone\_publication} - \text{lowest\_keystone\_publication})$ , and let *right\_fork\_keystone\_publication\_delay* =  $\text{right\_fork\_keystone\_publication} - \text{lowest\_keystone\_publication}$
  - e. If *left\_fork\_keystone\_publication\_delay* is greater than the SI chain's configured *keystone\_finality\_delay*, then set *continue\_scoring\_left\_fork* = false
  - f. If *right\_fork\_keystone\_publication\_delay* is greater than the SI chain's configured *keystone\_finality\_delay*, then set *continue\_scoring\_right\_fork* = false
  - g. Add the value from the publication latency lookup table corresponding to *left\_fork\_keystone\_publication\_delay* to *left\_fork\_score*
  - h. Add the value from the publication latency lookup table corresponding to *right\_fork\_keystone\_publication\_delay* to *right\_fork\_score*
6. If *left\_fork\_score* is greater than *right\_fork\_score* then the left fork is better. If *right\_fork\_score* is greater than *left\_fork\_score* then the right fork is better. If both scores are equal, then the chains are equally valid, and similar to most other consensus protocols, whatever blockchain is currently valid should remain valid unless another chain surpasses its score in the future

Note that the algorithm above uses the *keystone\_finality\_delay* for determining when one chain significantly outpaces another. Chains may optionally choose to adopt a *keystone\_finality\_delay* which is different from the maximum lag keystones at a particular height are allowed relative to a competing chain. In that event, the higher of the two numbers (measured in SP blocks) will represent when the lack of publications of an attacking chain can confirm finality for the challenged chain at a particular keystone height.

To maximize the security profile of Proof-of-Proof in a variety of adversarial conditions, the recommended SI fork resolution algorithm described above is based on the following favorable properties:

- Only the timeliness of publication of keystone block references is considered (all publications reference at least one keystone)
- Publications of all or most blocks are not assumed, and cannot be artificially done above fee market states to “buy” a high score for an attacking chain
- After the *keystone\_finality\_delay* is passed without publication of an attacking chain at a particular height, the fork resolution algorithm won’t possibly choose another chain as valid unless the SP blockchain (and in tandem, any upstream SP blockchains) experience forks
- The amount of data and processing required to run the full fork resolution algorithm is minimal and reasonable for SPV clients (and in the case of VeriBlock, altchains which are secured to VeriBlock and thus must validate VeriBlock in an SPV-like manner)

## 6.6 Proof-of-Proof Reward Scheme

While the magnitude of the rewards will vary widely based on the SI blockchain, the block mining algorithm(s) it employs, its maturity, and market projections, all SI blockchains will have some budget to allocate per unit-time.

Rather than allocate a fixed amount of rewards per-block, it is recommended that SI blockchains allocate their PoP security budgets such that they incentivize publication of blocks in a way that is most beneficial to the SI blockchain’s security, and that minimizes any potential conflicts-of-interest for people simultaneously engaged in SP block creation and SI PoP mining (see section 7.3 for additional discussion).

Due to the keystone structure, publications of the keystone block itself are the most valuable to the altchain’s consensus, and publications of the blocks following it are decreasingly valuable (they have a lower likelihood of being published to the earliest-possible SP block than the keystone block itself).

Because PoP transactions occur necessarily after the blocks they protect (including the block header they publish), PoP payouts are calculated after a sufficient delay (in SI blocks, generally set to take ~4+ hours after the occurrence of the published block) to ensure all SP transactions done on behalf of the SI chain have cleared, and that neither the SP chain nor any upstream SP chains are likely to experience a reorganization. An SI blockchain protocol will decide on the following variables to use for PoP score calculation:

| PoP Payout Parameters           |   |   |
|---------------------------------|---|---|
| Parameter                       | Description   | Value Recommendation                                  |
| pop_reward_settlement_interval  | The number of SI blocks following the PoP'd block for which a PoP endorsement of that block can occur | 4-8 hours of SI chain blocks                          |
| pop_reward_payment_delay        | The number of blocks after the PoP'd block to wait before performing PoP payout                       | Generally, $1.25 * pop\_reward\_settlement\_interval$ |
| pop_relative_score_lookup_table | An associative function scoring PoP publications by their timeliness in the SP chain                  | <b>See below</b>                                      |

The *pop\_relative\_score\_lookup\_table* is a function which maps a particular *publication\_timeliness* delay in SP blocks to a score for the PoP transaction. Note that each value is based off of the offset from the first publication (ex: if the first PoP publication of SI block 100 is in SP block 1000, then each key in the lookup table will start at 0 and be based on tardiness in the SP chain relative to SP block 1000). Also note that this is the relative scoring for payout calculation, not for the relative weighting used in the fork resolution algorithm. Generally, this curve should be more aggressive than the fork resolution lookup table, as the purpose of the payout curve is to incentivize rapid publication, rather than determine SP-finality for the SI chain.

Generally, something similar to the following table is recommended for altchains inheriting security from the VeriBlock blockchain:

| Example <i>pop_relative_score_lookup_table</i> |   |
|--|---|
| VBK Publication Timeliness                     | PoP Reward Score                                  |
| $0 \leq publication\_timeliness < 10$          | 1.000   |
| $10 \leq publication\_timeliness < 50$         | $\frac{1}{(publication\_timeliness - 10)^{1.05}}$ |
| $50 \leq publication\_timeliness$              | 0   |

This allows any publication in the first ~5 minutes (10 VBK blocks) to receive the same reward (and disincentivizes any VBK PoW miners who are also PoP mining for the particular altchain from censoring others' PoP transactions in an attempt to increase their reward), while still providing a "long tail" of smoothed but rapidly-deteriorating rewards for those with less

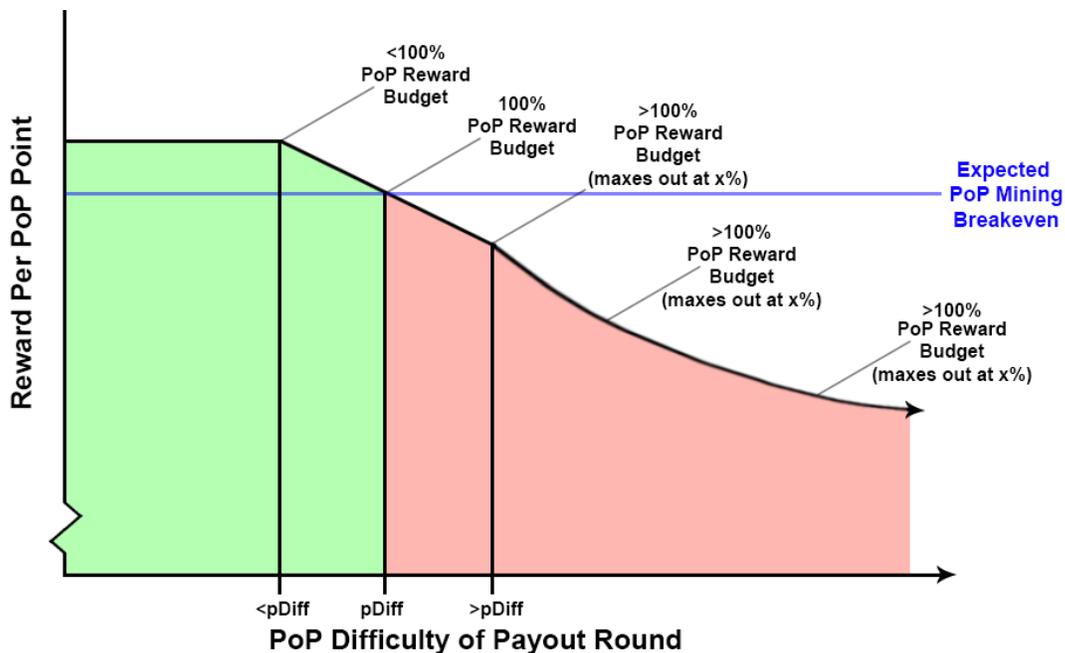
competitive VBK fees. Note that the sum of all PoP reward scores will be used later in calculating the payouts, which will reward PoPs based on the total number of PoPs for a particular block and the number of PoPs for previous recent blocks.

In addition to the existing difficulty (PoW, PoC, PoS, etc.) for the block mintage protocol, PoP introduces an additional PoP difficulty (*pop\_difficulty* or *pDiff*). This PoP difficulty reflects the recent PoP scores of blocks in the SI blockchain, and functions as a means of regulating PoP payouts and allowing fluctuations in the SP fee market to be priced into the SI blockchain's PoP mining rewards.

After *pop\_reward\_settlement\_interval* SI blocks pass after a particular block, the PoP score for that particular block can be calculated by summing up the values from *pop\_relative\_score\_lookup\_table* corresponding to each PoP transaction (which proofs to the best-known VeriBlock blockchain) endorsing the particular block. This PoP score is itself fed into a function which determines the altchain coin-per-PoP-score which will be paid out for that particular block. Note that many SI chains may opt to pay larger PoP rewards in particular blocks that are more valuable for consensus, and pay smaller (or zero) PoP rewards for other blocks.

For blocks which have a lower PoP publication score than the current *pop\_difficulty*, the total reward paid out will be lower than the expected average payout-per-block. Accordingly, blocks with a higher PoP publication score than the current *pop\_difficulty* will pay out a total reward that is larger than the expected average payout-per-block. Over time, the rewards paid for PoP will average out to the expected per-block payout.

A recommended reward schedule is shaped as follows:



Note that *pDiff* is short for *pop\_difficulty*. Three distinct sections to this rewards curve exist.

The first section of the reward curve is a horizontal reward curve, which pays out a fixed expected-higher-than-market (for blocks paying a high PoP reward) amount. For example, if the *pop\_difficulty* is 100, then a block with a PoP reward score of 1 would pay out the same coins-per-PoP-point as a block with a PoP reward score of 2. This disincentivizes block miners on the SP and the SI chains from attempting to prevent competing SI PoP endorsements from completing. In most cases (SI blocks that are priced to pay for endorsements of themselves, rather than just act as an extreme market condition detector), rewards-per-point on this horizontal section will be above the breakeven of the transaction fees an SI blockchain PoP miner would pay on the SP chain.

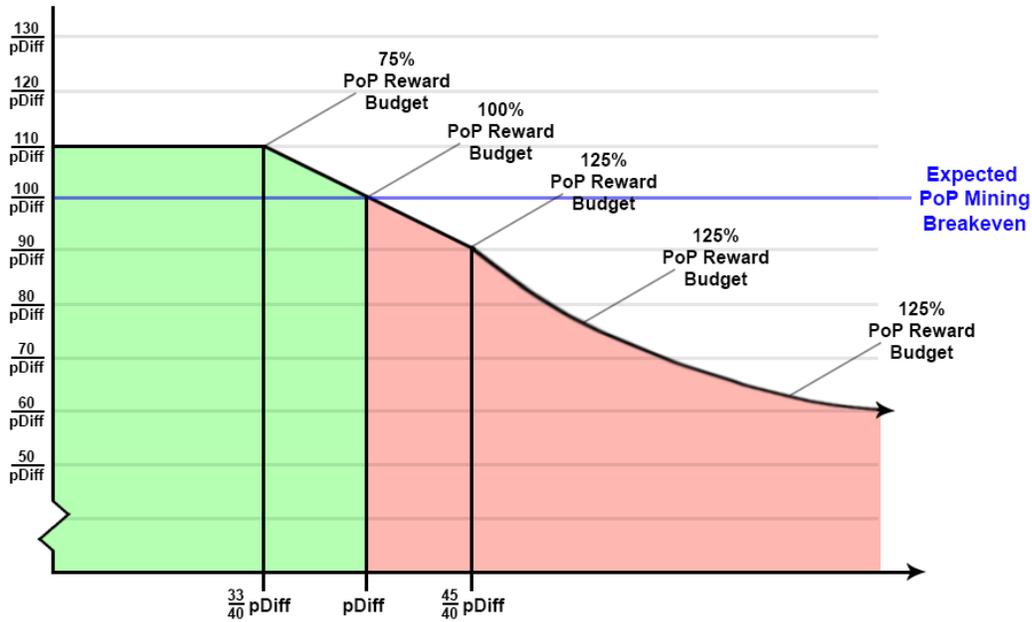
The second section of the reward curve provides a linearly-decreasing reward-per-PoP-point, which allows the total transaction fees paid by SI blockchain PoP miners to naturally settle based on the current SP blockchain fee market situation. In most cases (SI blocks that are priced to pay for endorsements of themselves), the PoP mining breakeven (based on transaction fees SI blockchain miners pay on VeriBlock) will occur somewhere in this section (and during stable market conditions, will be very close to *pDiff*). The total “pie” of shared rewards by PoP miners of the particular rewarded block continues to increase throughout this section.

The third section will generally only be encountered in times of extreme market fluctuations (in the VeriBlock fee market and/or the value ratio between VBK and the SI blockchain’s native coin), and places a cap on the total reward a block will pay out (the total “pie” of shared rewards by PoP miners of the particular rewarded block stays fixed at any point in this section).

SI blockchains should have different blocks which have different reward curves (either just shifted/stretched versions of the same curve, or curves with different overall shapes). This introduces the jittering discussed in section 5.2. A reward curve which exists entirely below the expected PoP mining breakeven line is expected to only be published in the event that the SP fee market drops in price drastically and/or the SI blockchain’s native coin appreciates significantly in value relative to the SP native coin in a short period of time. This allows the system to adjust to external market fluctuations more quickly (as publications of these blocks will cause *pDiff* to increase, lowering the per-PoP-reward for all blocks).

Blockchains, particularly those with fast block times, may opt to only offer non-zero PoP payouts on a fraction of their blocks. This allows the altchain to provide a large reward pool less frequently, reducing the risk that the SI blockchain will be entirely priced out of the PoP market.

An example of a normal PoP block reward curve (on a blockchain wishing to reward 100 SI coins for the particular rewarded block, never to exceed 125 SI coins):



In order to calculate the PoP rewards to pay out in a particular block (which rewards PoP miners who published the block  $pop\_reward\_payment\_delay$  blocks ago):

1. Let  $block\_to\_calculate\_rewards\_for = current\_block - pop\_reward\_payment\_delay$
2. Let  $pop\_transactions\_to\_consider$  = the set of all PoP transactions (later referred to as ATV transactions; ones that demonstrate publication of a SI blockchain's state to VeriBlock) which endorse  $block\_to\_calculate\_rewards\_for$  which exist in the interval  $(block\_to\_calculate\_rewards\_for, block\_to\_calculate\_rewards\_for + pop\_reward\_settlement\_interval]$
3. Based on the altchain's view of the VeriBlock blockchain current to the last applied (immediately-preceding) altchain block, remove all altchain PoP transactions from  $pop\_transactions\_to\_consider$  which don't publish to the best-known VeriBlock blockchain
4. Let  $first\_publication\_index\_in\_veriblock$  = the index in the VeriBlock blockchain of the first PoP publication in  $pop\_transactions\_to\_consider$
5. Let  $total\_pop\_score\_for\_rewarded\_block = \text{sum}(\text{each PoP transaction in } pop\_transactions\_to\_consider \text{ score based on its relative timeliness to } first\_publication\_index\_in\_veriblock)$
6. Let  $reward\_per\_pop\_point$  = the corresponding reward-per-point from the PoP reward curve for  $current\_block$
7. Award (generally through the coinbase transaction, **see section 6.7 for a discussion on alternative options**) the PoP miner of each PoP transaction in  $pop\_transactions\_to\_consider$  based on the corresponding timeliness of publication of the particular PoP transaction relative to  $first\_publication\_index\_in\_veriblock$

It is generally recommended that altchains have one block in each keystone period (preferably towards the beginning of the keystone period) which splits out a fixed-size reward equally amongst all participants. While this does create an incentive for SP block miners to potentially censor SI transactions endorsing this block and collect the majority of the entire reward themselves (the censored SI transactions are still likely to be included in a future SP block), it acts as a hedge against extremely volatile fee market and price activity which could potentially exceed the protocol's SP fee market discovery done by the PoP difficulty calculations.

### **6.7 Alternatives to Coinbase PoP Payouts**

While most blockchains will generally opt to provide PoP payouts through the coinbase transaction, PoP payouts can also be done via a general smart contract on blockchains which support it. This "coinbase-like" smart contract could allow users to "top-up" the blockchain's security budget, amongst other interesting features.

Do note, however, that smart contracts may also pose a vulnerability because they could be used to artificially change or flatten the reward jittering and reduce the chain's security profile by, paradoxically, providing additional rewards to PoP miners of low-reward rounds. Mitigation strategies for this involve blinding general smart contracts from being able to read the PoP information or having PoP mining occur with a special type of address only able to receive coins from a the PoP payout smart contract. This attack vector is one of the reasons the VeriBlock blockchain does not offer turing-complete smart contract scripts.

## **7 VeriBlock Blockchain PoP Parameter Selection**

The VeriBlock blockchain is a concrete example of selecting the PoP protocol parameters explained in chapter 6.

PoP miners acting on behalf of the VeriBlock blockchain network must be properly incentivized to: publish PoP data regarding the most recent VeriBlock block available to them, get their PoP publication in the Bitcoin blockchain in the next block, and return their PoP publication proof back to the VeriBlock blockchain as quickly as possible.

The PoP mining market is permissionless and elastic. Anyone can elect to PoP mine at any time, and the direct cost associated with PoP mining (the paid Bitcoin transaction fee) can be configured by the PoP miner, with a direct effect on the PoP miner's expected payout probability.

Publication of a VeriBlock block header to Bitcoin as part of a PoP payload is incentivized with the payout scheme described in section 6.6.

### **7.1 Incentivizing Rapid Publication**

The ideal VeriBlock PoP miner publishes VeriBlock PoP data such that it occurs in the next Bitcoin block. While some edge-case scenarios exist in which this is impossible (due to a lack of global information availability because of propagation and processing latency, and Bitcoin PoW miners' unique software and associated weightings and refresh rates for selecting a subset of their mempool for mining), there generally exists a known fee market within which all Bitcoin transaction originators (including VeriBlock PoP miners) participate.

Fundamentally, this fee market provides a probability  $p$  with which a transaction with a fee  $f$  will make it into the next block (or the 2<sup>nd</sup>, or 3<sup>rd</sup>, ... next blocks). To properly incentivize VeriBlock PoP miners to pay sufficient Bitcoin transaction fees, VeriBlock rewards are weighted such that the ratio of VeriBlock coins rewarded to Bitcoin fee spent is maximized when a PoP miner pays a sufficient (but not excessive) fee to make it into the next block.

### **7.2 Incentivizing Return of PoP Data**

To receive compensation for a PoP publication to Bitcoin, the PoP miner must construct a proof that the PoP publication transaction has been included in the Bitcoin blockchain, and return that proof to the VeriBlock blockchain.

Even if the PoP publication transaction does not get included in the next possible Bitcoin block (or, more formally, the first Bitcoin block in which any other PoP publication transaction which proves the same VeriBlock block occurs), a PoP miner is incentivized to return the PoP publication proof back to the VeriBlock blockchain if the block which the PoP publication is close enough to the first possible block to still be relevant to VeriBlock security.

### **7.3 Disincentivizing Bitcoin PoW Miners from Censoring PoP Transactions**

Bitcoin PoW miners are modeled as rational economic actors whose goal is to maximize short-term profits. They are incentivized by the Bitcoin block reward, Bitcoin transaction fees, and additional external benefits they can acquire by selecting the data they include in their block.

Because PoP publication transactions provide a monetary or pseudo-monetary reward external to the Bitcoin blockchain itself, they are considered to provide external incentives.

Because Bitcoin PoW miners can also perform PoP mining for the VeriBlock blockchain, it's possible that they would want to prevent publication of any PoP data in their Bitcoin block other than their own PoP publication if doing so would make their PoP publication more valuable. For example, if a fixed block reward was offered and split amongst the miners responsible for all valid timely publications of a particular VeriBlock block to Bitcoin, then a particular Bitcoin PoW miner would be incentivized to censor all other VeriBlock PoP miners for a range of the most recent VeriBlock blocks from entry into their Bitcoin block, allowing

them to be the only successful PoP miner at the first Bitcoin block index (and collect the majority of the reward).

To prevent this behavior, VeriBlock’s reward mechanism is balanced to make a Bitcoin PoW miner’s most incentivized action to generally be to include most or all PoP transactions which are competitive in the regular Bitcoin fee market.

The total PoP coinbase reward at a particular VeriBlock block  $n$  pays the PoP miners who endorsed VeriBlock block  $(n - pop\_reward\_delay\_period)$ . The  $(pop\_reward\_delay\_period)$  is 500 VeriBlock blocks. These endorsements must be contained within the  $pop\_settlement$  period of the block in question (400 VeriBlock blocks). For the first  $payment\_delay\_period$  blocks in the VeriBlock blockchain, there is no PoP coinbase.

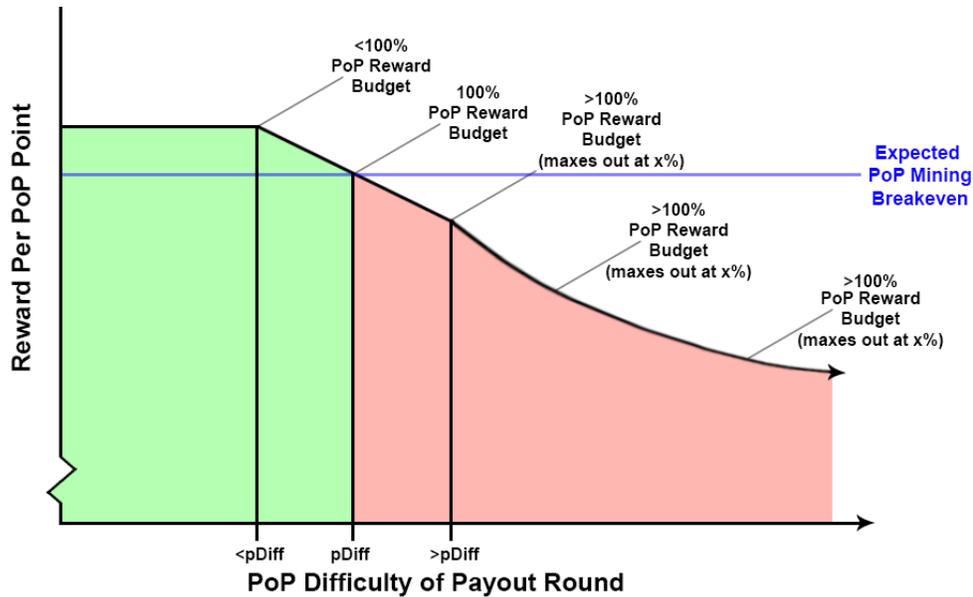
The VeriBlock blockchain has a floating reward mechanism which alters the payout for each VeriBlock block based on the volume of PoP transactions which endorsed recent VeriBlock blocks. The intuition for this process is similar to the difficulty of Bitcoin; in this case the “hashing power” is the number and timeliness of transactions, and the ‘difficulty’ alters the reward per publication with weight based on its timeliness (in the same way that difficulty in a PoW sense alters the reward per hash). Note that VeriBlock pays rewards to PoP publications in the first 8 Bitcoin blocks that PoPs for a particular VeriBlock block are published in (according to the following weight chart for reward distribution).

| <b>PoP Difficulty Contribution and Reward Ratio (Based on <math>1/x^2</math>)</b> |                   |
|---|-------------------|
| <b>Relative Bitcoin Block</b>   | <b>Multiplier</b> |
| 1   | 1.000             |
| 2   | 0.250             |
| 3   | 0.111             |
| 4   | 0.063             |
| 5   | 0.040             |
| 6   | 0.028             |
| 7   | 0.020             |
| 8   | 0.016             |

As explained in section 6.6, a reward curve converts a PoP score (relative to its trailing average, which behaves like a difficulty) to a total block reward. This curve provides a maximum payout per PoP point, and ensures that that the total block reward for PoP miners stays at or below a particular value.

Section 5.2 explains a jittering mechanism used for price discovery and to strategically incentivize publication of sufficient context for the chain to maintain continuity.

Recall:



This curve (along with the PoP score reward weightings, finality periods, and the PoP reward share) produces a number of desirable traits:

- Bitcoin PoW miners who censor others' PoP transactions generally do so at a loss, even if they themselves are VeriBlock PoP miners
- VeriBlock PoW miners are incentivized to include as many PoP transactions as possible, even if they themselves are VeriBlock PoP miners
- PoP miners are incentivized to continue to PoP mine until roughly breakeven

This reward scheme:

- Incentivizes Bitcoin PoW miners who also participate in VeriBlock PoP Mining
  - Their own PoP transaction(s) earns a higher reward for including additional PoP transactions up to approximately the trailing difficulty
  - The slow descent of per-PoP reward after the trailing difficulty allows the Bitcoin blockchain fee market to function; PoP miners (particularly those who are using the OP\_RETURN on a transaction they already planned to send) can pay fees which are higher than the Bitcoin PoW miner's marginal losses from PoP for including their PoP transaction
- Incentivizes standalone VeriBlock PoP miners

- Based on the current state of the mempool and efficiency of the Bitcoin fee market, they can make informed decisions regarding profitability of PoP mining
- The ratio of reward:cost should be highest for PoP transactions in the next Bitcoin block (a fee would have to be 87.5% cheaper for 2<sup>nd</sup> block inclusion than 1<sup>st</sup> block inclusion), incentivizing PoP miners to pay adequate fees for speedy inclusion
- Even if the VeriBlock PoP miner's transaction does not make it into the next block, they still have a non-trivial incentive to return the publication to VeriBlock if it enters Bitcoin within the 6-block limit

#### 7.4 PoP Payout Round Multipliers / Reward Levels

In order to introduce jittering to prevent long periods of profitability and unprofitability with no interruption (and to also offer higher incentives for blocks whose publication to Bitcoin results in a higher security profile, such as keystone blocks), different blocks have different payout multipliers.

These payout multipliers double as a discovery mechanism for the VeriBlock blockchain to determine the profitability threshold for current PoP miners, which have the effect of properly setting the reward curves to near-cost to prevent censorship by Bitcoin PoW miners. Four levels of block payouts exist (1 through 4), with higher numbers corresponding to higher rewards.

Rounds 1-3 use the same curve but apply different multipliers to it, and round 4 uses a similarly-shaped curve but with different segment length ratios. Rounds 2, 3, and 4 reward curves all offer a starting reward which will be above the Bitcoin fee market price when the fee market and exchange rates are stable. During times of increasing Bitcoin transaction costs relative to VBK, it is expected that rounds 2 and 3 may fail to be published, causing *pDiff* to drop (and thus, increasing the starting reward for future rounds). During times of decreasing Bitcoin transaction costs relative to VBK, it is expected that round 1 would get published, causing *pDiff* to increase, thus lowering the starting reward for future rounds.

These payout multipliers are strategically placed to encourage several behaviors:

- Always perform publications of keystone blocks, which operate in at Round-4 payouts
- Perform publications of alternating blocks (increasing the likeliness that a jagged Bitcoin fee market intersects with the VeriBlock jittering to produce profitable publications at a higher frequency)

- Alternate between expected up and down rounds for faster external 'cost of proof' discovery, along with Round-4 payouts every 20 blocks for keystone block rewards.

Note that the adjustment of the reward level schedule does not happen immediately; the PoP difficulty changes on a delay to allow for all PoP transactions to come in; it is assumed that PoP miners will price in the expected changes based on other PoP transactions in Bitcoin and in the mempool (as the adjustment can be predicted with reasonable accuracy once one or two Bitcoin blocks occur).

In order to robustly handle significant Bitcoin fee market changes or VBK:BTC trade ratio swings, every 4<sup>th</sup> block in a keystone period (blocks in the set {3, 23, 43...}) splits its total block reward amongst all PoP miners who publish it to Bitcoin (still using PoP scoring to rate the value of transactions relative to one another); if only one PoP transaction for this block occurs the PoP miner receives the full block reward.

While publications of this block are subject to potential selective censorship by Bitcoin PoW miners (who want to claim the majority of the reward for themselves), it results in at least one block per keystone period being economically viable to PoP mine, even during times of significant Bitcoin fee market or exchange rate volatility.

## 7.5 VeriBlock Bitcoin Finality

Due to the random nature of block generation time and asynchronous nature of distributed networks, simply the inclusion of the header of a VeriBlock block in the Bitcoin blockchain cannot be used to determine “Bitcoin-finality” (finality that is guaranteed unless a reorganization occurs on Bitcoin itself) for a particular VeriBlock block and all of its ancestors.

As a result, an algorithm for relative score calculation must be introduced which determines which of a set of two or more possible chains is the most valid one. In order to determine the most valid chain (or the set of equally-valid chains for which additional information in the way of future blocks on one of the chains is required for tie-breaking), any two chains (each defined by their most recent block) must be comparable.

One of the most important utilities of the VeriBlock blockchain’s consensus algorithm is to produce Bitcoin-finality for the VeriBlock blockchain (which in turn leads to Bitcoin-finality for blockchains secured to VeriBlock); at a certain point, a blockchain up to a particular height which has no competitors published to Bitcoin for a period of ‘time’ should become unbeatable unless Bitcoin itself experiences a reorganization. As such, a grace period (measured in Bitcoin blocks) must be provided such that a chain which is uncontested by publications in Bitcoin from a particular block height down should be immortalized as final (and chains which were once considered valid competitors should eventually become impossible if they do not produce new publications to Bitcoin within the finality period). Such a system forces any attacker to frequently publish the present state of an alternate chain they are building in order for their attacking chain to possibly be accepted by the network.

However, such a finality window also creates the possibility for an attacker to produce a valid chain offline which, in the event that the “main” blockchain fails to make a presence in Bitcoin within the finality closure window, becomes a nuclear weapon; such a chain will always be considered more valid than the main chain, no matter how many blocks are built on top of the “main chain” and how many PoP publications of that blockchain get into Bitcoin at a later point in time.

In order for such an attack to be successful, one of the following three incredibly unlikely events must occur:

- Bitcoin miners demonstrate complete censorship of PoP transactions of the main VeriBlock chain while permitting PoP transactions which endorse an attacking VeriBlock chain to be included in the Bitcoin blockchain for the duration of the finality delay period
- VeriBlock PoW miners successfully censor the VeriBlock main chain continually and prevent all valid PoP endorsements of the main chain from making it into later blocks

in the main chain (attack eventually fails if the attacker fails to maintain complete censorship control of the main chain)

- No new VeriBlock keystone period on the main chain is produced during the entire duration of the finality period of Bitcoin blocks, but at least one new VeriBlock keystone period is produced privately and proofed to Bitcoin during the finality period

The first scenario would demonstrate a complete failure of the Bitcoin blockchain to maintain a censorship-free transactional system; analogous to a majority hashrate attack wherein a single agent or multiple colluding agents control a majority of the hashrate. As a result, Bitcoin can no longer be relied upon for security which adheres to the principles of DTTP.

The second scenario would only work for the duration of time that the VeriBlock PoW 51% attackers are able to maintain complete censorship over the main blockchain; eventually the valid PoP endorsements would become part of the main blockchain and the chain would no longer appear to the fork resolution algorithm to have a gap larger than the finality delay.

The third scenario is a driving force between the block time and finality delay parameters of the VeriBlock protocol, which can be tweaked to make the probability of such an event occurring arbitrarily small (and have been selected to all-but-eliminate the possibility of such an anomaly).

## 7.6 Block Time and Finality Delay Parameter Selection

The probability  $p$  that  $n$  Bitcoin blocks occur before  $m$  VeriBlock blocks with a VeriBlock block time of  $t$  seconds (assuming neither network is in a period of hashrate fluctuation above or below that communicated by present difficulty) is:

$$p = \sum_{k=n}^{n+m-1} \binom{n+m-1}{k} \left( \frac{\frac{1}{600}}{\frac{1}{600} + \frac{1}{t}} \right)^k \left( \frac{\frac{1}{t}}{\frac{1}{600} + \frac{1}{t}} \right)^{n+m-1-k}$$

A few tables demonstrating the probability of an entire finality period ( $n$ ) passing before a particular number of VeriBlock blocks are produced ( $m=\{1, 3, 5, 10\}$ ) at particular block times ( $t$ ) help with choosing acceptable values for all three variables when paired with realistic security delay expectations and real-world propagation performance on modern cryptocurrency networks.

|                     |   |   |   |   |   |
|---------------------|---|---|---|---|---|
| Color               |  |  |  |  |  |
| Frequency (minimum) | 0   | 16.6 hrs  | 69 days   | 19 years  | 1.9 millennia   |
| Frequency (maximum) | 16.6 hrs  | 69 days   | 19 years  | 1.9 millennia   | $\infty$  |

| Probability of $n$ Bitcoin Blocks Before $m=1$ VeriBlock Blocks vs VeriBlock Block Time |     |                                  |         |         |         |         |         |         |         |         |         |
|---|-----|----------------------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
|   |     | Number of Bitcoin Blocks ( $n$ ) |         |         |         |         |         |         |         |         |         |
| VBK Block Time (t, sec)   |     | 1                                | 2       | 3       | 4       | 5       | 6       | 7       | 8       | 9       | 10      |
|   | 15  | 1.2E-02                          | 1.5E-04 | 2.0E-06 | 3.9E-07 | 3.1E-10 | 3.8E-12 | 4.8E-14 | 6.0E-16 | 7.5E-18 | 9.3E-20 |
|   | 30  | 2.5E-02                          | 6.3E-04 | 1.6E-06 | 2.0E-06 | 9.8E-09 | 2.4E-10 | 6.1E-12 | 1.5E-13 | 3.8E-15 | 9.5E-17 |
|   | 45  | 3.8E-02                          | 1.4E-03 | 5.3E-05 | 6.3E-06 | 7.4E-08 | 2.8E-09 | 1.0E-10 | 3.9E-12 | 1.5E-13 | 5.5E-15 |
|   | 60  | 5.0E-02                          | 2.5E-03 | 1.3E-04 | 6.3E-06 | 3.1E-07 | 1.6E-08 | 7.8E-10 | 3.9E-11 | 2.0E-12 | 9.8E-14 |
|   | 90  | 7.5E-02                          | 5.6E-03 | 4.2E-04 | 3.2E-05 | 2.4E-06 | 1.8E-07 | 1.3E-08 | 1.0E-09 | 7.5E-11 | 5.6E-12 |
|   | 120 | 1.0E-01                          | 1.0E-02 | 1.0E-03 | 1.0E-04 | 1.0E-05 | 1.0E-06 | 1.0E-07 | 1.0E-08 | 1.0E-09 | 1.0E-10 |

| Probability of $n$ Bitcoin Blocks Before $m=3$ VeriBlock Blocks vs VeriBlock Block Time |     |                                  |         |         |         |         |         |         |         |         |         |
|---|-----|----------------------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
|   |     | Number of Bitcoin Blocks ( $n$ ) |         |         |         |         |         |         |         |         |         |
| VBK Block Time (t, sec)   |     | 1                                | 2       | 3       | 4       | 5       | 6       | 7       | 8       | 9       | 10      |
|   | 15  | 3.6E-02                          | 9.0E-04 | 1.9E-05 | 3.5E-07 | 6.1E-09 | 1.0E-10 | 1.6E-12 | 2.6E-14 | 3.9E-16 | 5.9E-18 |
|   | 30  | 7.0E-02                          | 3.5E-03 | 1.4E-04 | 5.4E-06 | 1.9E-07 | 6.3E-09 | 2.0E-10 | 6.3E-12 | 2.0E-13 | 5.7E-15 |
|   | 45  | 1.0E-01                          | 7.5E-03 | 4.7E-04 | 2.6E-05 | 1.4E-06 | 6.8E-08 | 3.3E-09 | 1.5E-10 | 7.0E-12 | 3.2E-13 |
|   | 60  | 1.3E-01                          | 1.3E-02 | 1.1E-03 | 7.9E-05 | 5.5E-06 | 3.7E-07 | 2.4E-08 | 1.5E-09 | 9.0E-11 | 5.4E-12 |
|   | 90  | 1.9E-01                          | 2.7E-02 | 3.3E-03 | 3.7E-04 | 3.9E-05 | 3.9E-06 | 3.7E-07 | 1.5E-08 | 3.2E-09 | 2.9E-10 |
|   | 120 | 2.3E-01                          | 4.5E-02 | 7.3E-03 | 1.1E-03 | 1.5E-04 | 2.0E-05 | 2.6E-06 | 3.2E-07 | 3.9E-08 | 4.7E-09 |

| Probability of $n$ Bitcoin Blocks Before $m=5$ VeriBlock Blocks vs VeriBlock Block Time |                                  |         |         |         |         |         |         |         |         |         |         |
|---|----------------------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
|   | Number of Bitcoin Blocks ( $n$ ) |         |         |         |         |         |         |         |         |         |         |
| VBK Block Time (t, sec)   |                                  | 1       | 2       | 3       | 4       | 5       | 6       | 7       | 8       | 9       | 10      |
|   | 15                               | 5.8E-02 | 2.2E-03 | 6.3E-05 | 1.6E-06 | 3.5E-08 | 7.3E-11 | 1.4E-11 | 2.7E-13 | 4.9E-15 | 8.5E-17 |
|   | 30                               | 1.1E-01 | 8.0E-03 | 4.6E-04 | 2.3E-05 | 1.0E-06 | 4.3E-08 | 1.7E-09 | 6.3E-11 | 2.3E-12 | 7.9E-14 |
|   | 45                               | 1.5E-01 | 1.7E-02 | 1.4E-03 | 1.1E-04 | 7.2E-06 | 4.5E-07 | 2.6E-08 | 1.5E-09 | 8.0E-11 | 4.2E-12 |
|   | 60                               | 1.9E-01 | 2.8E-02 | 3.2E-03 | 3.1E-04 | 2.8E-05 | 2.3E-06 | 1.8E-07 | 1.4E-08 | 9.8E-10 | 6.8E-11 |
|   | 90                               | 2.6E-01 | 5.4E-02 | 9.2E-03 | 1.4E-03 | 1.8E-04 | 2.3E-05 | 2.6E-06 | 2.9E-07 | 3.2E-08 | 3.3E-09 |
|   | 120                              | 3.1E-01 | 8.5E-02 | 1.9E-01 | 3.7E-03 | 6.6E-04 | 1.1E-04 | 1.7E-05 | 2.5E-06 | 3.6E-07 | 5.0E-08 |

| Probability of $n$ Bitcoin Blocks Before $m=10$ VeriBlock Blocks vs VeriBlock Block Time |                                  |         |         |         |         |         |         |         |         |         |         |
|--|----------------------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
|  | Number of Bitcoin Blocks ( $n$ ) |         |         |         |         |         |         |         |         |         |         |
| VBK Block Time (t, sec)  |                                  | 1       | 2       | 3       | 4       | 5       | 6       | 7       | 8       | 9       | 10      |
|  | 15                               | 1.1E-01 | 7.2E-03 | 3.5E-04 | 1.4E-05 | 5.0E-07 | 1.6E-08 | 4.4E-10 | 1.2E-11 | 2.9E-13 | 7.0E-15 |
|  | 30                               | 1.8E-01 | 2.4E-02 | 2.4E-03 | 1.9E-04 | 1.3E-05 | 8.1E-07 | 4.6E-08 | 2.5E-09 | 1.2E-10 | 5.8E-12 |
|  | 45                               | 2.4E-01 | 4.6E-02 | 6.6E-03 | 7.9E-04 | 8.2E-05 | 7.7E-06 | 6.5E-07 | 5.2E-08 | 3.9E-09 | 2.7E-10 |
|  | 60                               | 2.7E-01 | 6.9E-02 | 1.3E-02 | 2.1E-03 | 2.9E-04 | 3.6E-05 | 4.0E-06 | 4.3E-07 | 4.2E-08 | 4.0E-09 |
|  | 90                               | 3.2E-01 | 1.2E-01 | 3.2E-02 | 7.6E-03 | 1.5E-03 | 2.8E-04 | 4.8E-05 | 7.6E-06 | 1.1E-06 | 1.6E-07 |
|  | 120                              | 3.4E-01 | 1.6E-01 | 5.7E-02 | 1.7E-02 | 4.7E-03 | 1.1E-03 | 2.6E-04 | 5.3E-05 | 1.1E-05 | 2.0E-06 |

Data from the last year shows Bitcoin block propagation times to 50% of publicly accessible nodes ranging from roughly 1000ms to 2000ms<sup>[1]</sup>, making a 30-second block time (assuming slightly faster propagation speeds for VeriBlock due to the smaller individual block size) reasonable without significantly affecting the orphan rate of the network.

The probability of particular strengths of non-majority-hashrate miners successfully censoring all VeriBlock PoP transactions for a period of time is also relevant to selecting the minimum number of Bitcoin blocks required for the finality delay:

| Probability of $h\%$ Bitcoin Hashrate Miner Completely Censoring PoP for $m$ Bitcoin Blocks |                                  |         |         |         |         |         |         |         |         |         |         |
|---|----------------------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
|   | Number of Bitcoin Blocks ( $n$ ) |         |         |         |         |         |         |         |         |         |         |
|   | 1                                | 2       | 3       | 4       | 5       | 6       | 7       | 8       | 9       | 10      |         |
| Hashrate of Attacker ( $h\%$ )  | 1%                               | 1.0E-02 | 1.0E-04 | 1.0E-06 | 1.0E-08 | 1.0E-10 | 1.0E-12 | 1.0E-14 | 1.0E-16 | 1.0E-18 | 1.0E-20 |
|   | 5%                               | 5.0E-02 | 2.5E-03 | 1.3E-04 | 6.3E-06 | 3.1E-07 | 1.6E-08 | 7.8E-10 | 3.9E-11 | 2.0E-12 | 9.8E-14 |
|   | 10%                              | 1.0E-01 | 1.0E-02 | 1.0E-03 | 1.0E-04 | 1.0E-05 | 1.0E-06 | 1.0E-07 | 1.0E-08 | 1.0E-09 | 1.0E-10 |
|   | 15%                              | 1.5E-01 | 2.3E-02 | 3.4E-03 | 5.1E-04 | 7.6E-05 | 1.1E-05 | 1.7E-06 | 2.6E-07 | 3.8E-08 | 5.8E-09 |
|   | 20%                              | 2.0E-01 | 4.0E-02 | 8.0E-03 | 1.6E-03 | 3.2E-04 | 6.4E-05 | 1.3E-05 | 2.6E-06 | 5.1E-07 | 1.0E-07 |
|   | 25%                              | 2.5E-01 | 6.3E-02 | 1.6E-02 | 3.9E-03 | 9.8E-04 | 2.4E-04 | 6.1E-05 | 1.5E-05 | 3.8E-06 | 9.5E-07 |
|   | 30%                              | 3.0E-01 | 9.0E-02 | 2.7E-02 | 8.1E-03 | 2.4E-03 | 7.3E-04 | 2.2E-04 | 6.6E-05 | 2.0E-05 | 5.9E-06 |
|   | 35%                              | 3.5E-01 | 1.2E-01 | 4.3E-02 | 1.5E-02 | 5.3E-03 | 1.8E-03 | 6.4E-04 | 2.3E-04 | 7.9E-05 | 2.8E-05 |
|   | 40%                              | 4.0E-01 | 1.6E-01 | 6.4E-02 | 2.6E-02 | 1.0E-02 | 4.1E-03 | 1.6E-03 | 6.6E-04 | 2.6E-04 | 1.1E-04 |
|   | 45%                              | 4.5E-01 | 2.0E-01 | 9.1E-02 | 4.1E-02 | 1.9E-02 | 8.3E-03 | 3.7E-03 | 1.7E-03 | 7.6E-04 | 3.4E-04 |

Note that attempting such an attack requires that the attacking miner forfeits all of the transaction fees offered by VeriBlock PoP miners and must generate at least one valid VeriBlock block at the current VeriBlock difficulty each time the censorship attack is attempted.

Furthermore, unless the attacker is willing to continually forfeit their block rewards and transaction fees on Bitcoin in the event of a failed attack, the attacker would still publish their Bitcoin blocks in the Bitcoin blockchain, and the censorship of PoP transactions would be publicly visible whenever attempted (as a period of several Bitcoin blocks would lack any PoP validations of the public VeriBlock chain and would contain an endorsement of an unknown VeriBlock block instead). The frequency and average length of these published unsuccessful attacks can be used to approximate the hashrate controlled by the attempting attacker.

As a result of analyzing the probabilities of a 30% hashrate attacker successfully executing a censorship attack along with the probabilities for more than  $n$  Bitcoin blocks occurring before  $m=10$  VeriBlock blocks (the period over which the a full reward multiplier cycle runs) and determining that 30 seconds was the lowest our block time could go before threatening the decentralization of mining given recent relay performance of Bitcoin,  $n=11$  Bitcoin blocks was selected as the Bitcoin finality period.

## 8 VeriBlock Blockchain Specifications

A number of design decisions were made in architecting the VeriBlock blockchain to optimize the speed, scalability, and additional benefits that it provides to blockchains which leverage it for security, as well as make secure SPV implementations require minimal bandwidth and processing power. Due to the significant divergence of the VeriBlock blockchain architecture from any particular blockchain daemon software, we opted to build the VeriBlock codebase from the ground up.

### 8.1 Merkle Patricia Tree Ledger

VeriBlock's balance sheet (which associates each address to a balance and signature index) is implemented as a Merkle Patricia Tree, which allows compact balance proofs which offer  $O(\log(n))$  complexity for lookups, insertions, and deletions<sup>[4]</sup>. Note that this implementation detail isn't relevant to altchains using VeriBlock for security; a UTXO model could have been equivalently used.

### 8.2 Blocksize Calculation and Allowances

In order to properly align incentives of VeriBlock PoW miners with the health of the chain, VBK->BTC PoP transactions are not counted in the blocksize calculation (elsewise VeriBlock PoW miners would favor including less or zero VBK->BTC PoP transactions in their blocks in order to collect more fees from fee-paying transactions).

The size of standard VeriBlock transactions (which may contain PoP payloads from altchain PoP miners) is based off of the serialized format used for TxID calculation. The size of multisig VeriBlock transactions is based off of a fixed size calculation per signature present in the transaction.

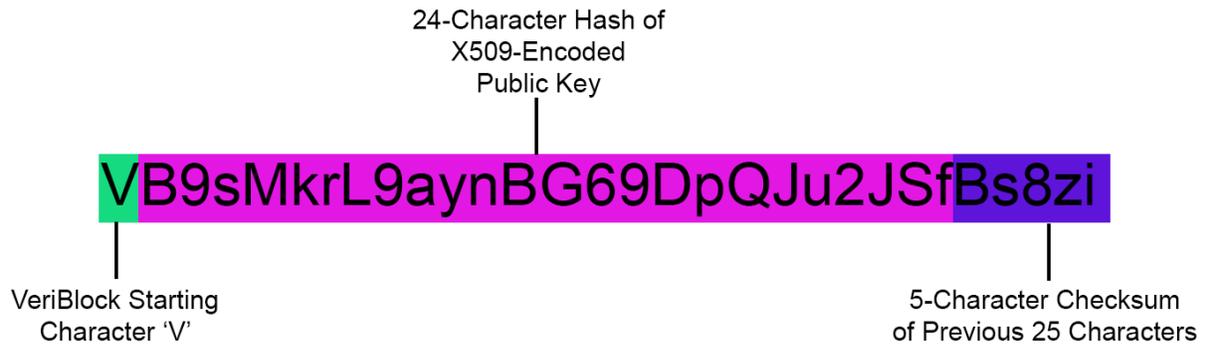
VBK blocks are limited to 256KB of normal transactions (standard and multisig) unless they contain VBK->BTC PoP transactions, at which point they are allowed up to 384KB, based on the ratio of the number of VBK->BTC PoP transactions the block contains relative to the network's current PoP difficulty. This acts as an additional incentive for VeriBlock PoW miners to include VBK->BTC PoP transactions in their blocks; while VBK->BTC PoP transactions don't directly pay fees, they allow the VBK PoW miner to include more fee-paying transactions in their blocks.

### 8.3 Standard Addresses and Standard Transactions

Signatures on the VeriBlock network use ECDSA with the secp256k1 curve. A standard address represents one public/private keypair, and is generated by the following algorithm:

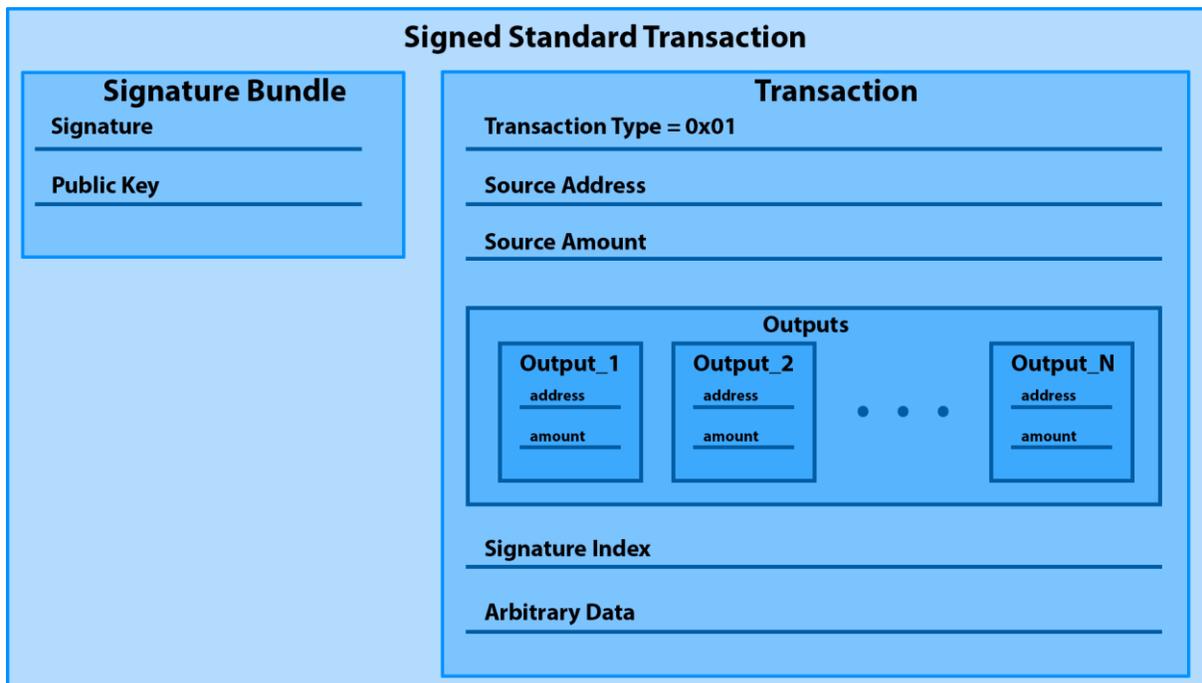
1. Generate an ECDSA public/private keypair

2. Hash the X509-encoded public key with SHA-256, represent as Base58
3. Concatenate 'V' + (first 24 characters of hash generated in step 2)
4. Hash the result of step 3 with SHA-256, represent as Base58
5. Concatenate the first 5 characters of the hash created in step 4 at the end of the result of step 3. The result is the standard address.



Standard transactions on VeriBlock send coins from one source address (which is a standard address) to one or more output addresses, and can optionally include additional arbitrary data, which altchain PoP miners use to publish altchain state data to VeriBlock. While no validation rules (other than a maximum size of 65,535 bytes) are enforced on the arbitrary data, there is an encoding format used by VeriBlock which, if followed by the altchain, will allow it to take advantage of altchain ID tracking, further explained in section 8.9.

The format of a standard transaction is as follows (without the signature, which signs the TxID generated by the serialized transaction):



## 8.4 Multisig Addresses and Transactions

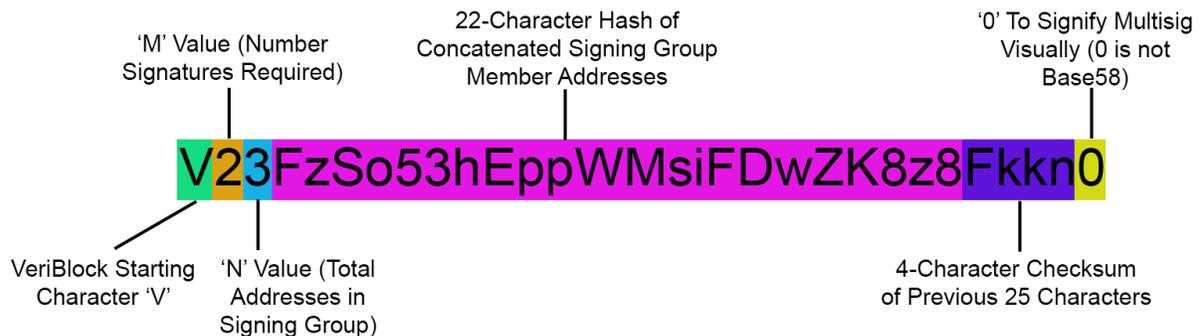
VeriBlock supports M-of-N multisig transactions, up to M and N values of 58. Multisig addresses follow a special format which allows the M and N values to be obtained by inspection.

A multisig address on VeriBlock is composed of and controlled by multiple standard VeriBlock addresses.

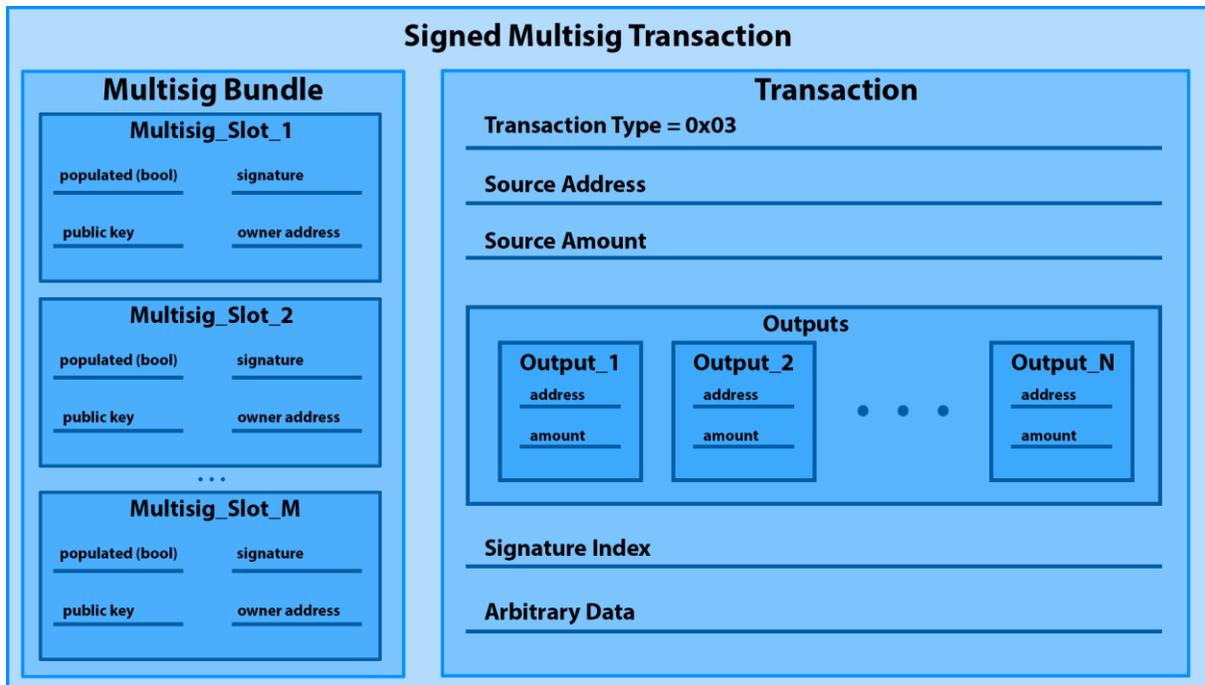
Conveniently, this allows easy generation of additional multisig addresses with different M values (up to N) which share the same signing group by just changing the M value and re-calculating the checksum.

The algorithm for generating a VeriBlock multisig address is:

1. Generate or Select N (where  $1 < N \leq 58$ ) standard VeriBlock addresses which represent the desired members of the signing group
2. Decide on a value M (where  $0 < M \leq N$ ), which represents the threshold of unique members who have to sign a transaction for it to be considered valid
3. Concatenate all String representations of the VeriBlock addresses (and store the order in which they were concatenated, note that the same standard address can be used multiple times)
4. Hash the concatenation created in step 3 with SHA-256, represent as Base58
5. Concatenate: 'V' + Base58Encode(M - 1) + Base58Encode(N - 1) + (first 22 characters of hash generated in step 4)
6. Hash the result of step 5 with SHA-256, represent as Base58
7. Concatenate the first 4 characters of the hash created in step 6 at the end of the result of step 5
8. Append a '0' to the end of the result of step 7. The result is the multisig address.



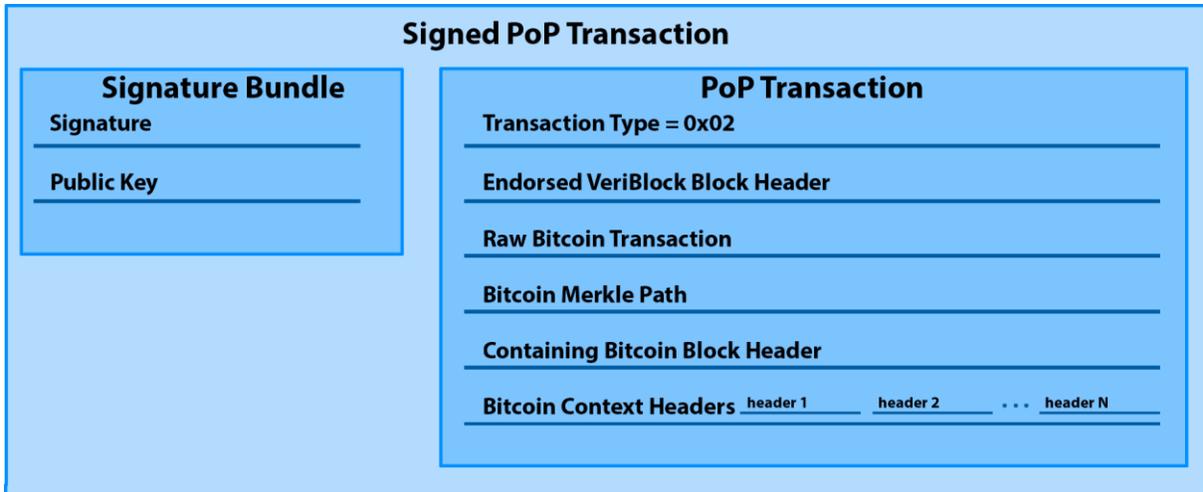
The format of a signed multisig transaction is as follows:



Exactly M MultisigSlots in the MultisigBundle must be populated and validly sign the TxID of the transaction's simple serialization. These must be populated in order such that the multisig source address can be recomputed.

### 8.5 PoP Transactions

Only standard addresses can generate VeriBlock PoP transactions. These PoP transactions include the endorsed VeriBlock block, the full Bitcoin transaction containing the VeriBlock block header and PoP payout information, the Merkle path proving the Bitcoin transaction is in a valid Bitcoin block, the header of the Bitcoin block which includes the Bitcoin transaction (and validates the Merkle root), and 0 or more context Bitcoin block headers required to connect the containing Bitcoin block to the known state of the Bitcoin blockchain at the time of the endorsed VeriBlock block. They follow the format:

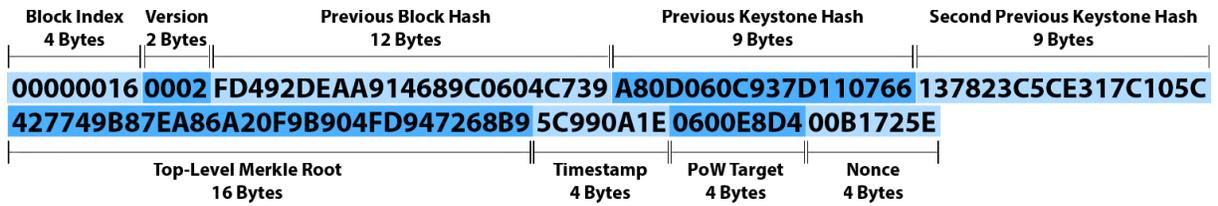


## 8.6 Block Format

The desire to fit alongside PoP payout information into a Bitcoin 80-byte OP\_RETURN (for efficiency sake, see section 8.10 for the alternate arbitrary data encoding format) along with keystoning (explored in section 6.2) and the desire to have the block index encoded directly in the header (such that PoP publications of non-continuous headers can be ordered appropriately for Early Attack Detection) dictates a divergence from the traditional block header formats of other blockchains. The block header is 64 bytes long, and is formatted as follows:

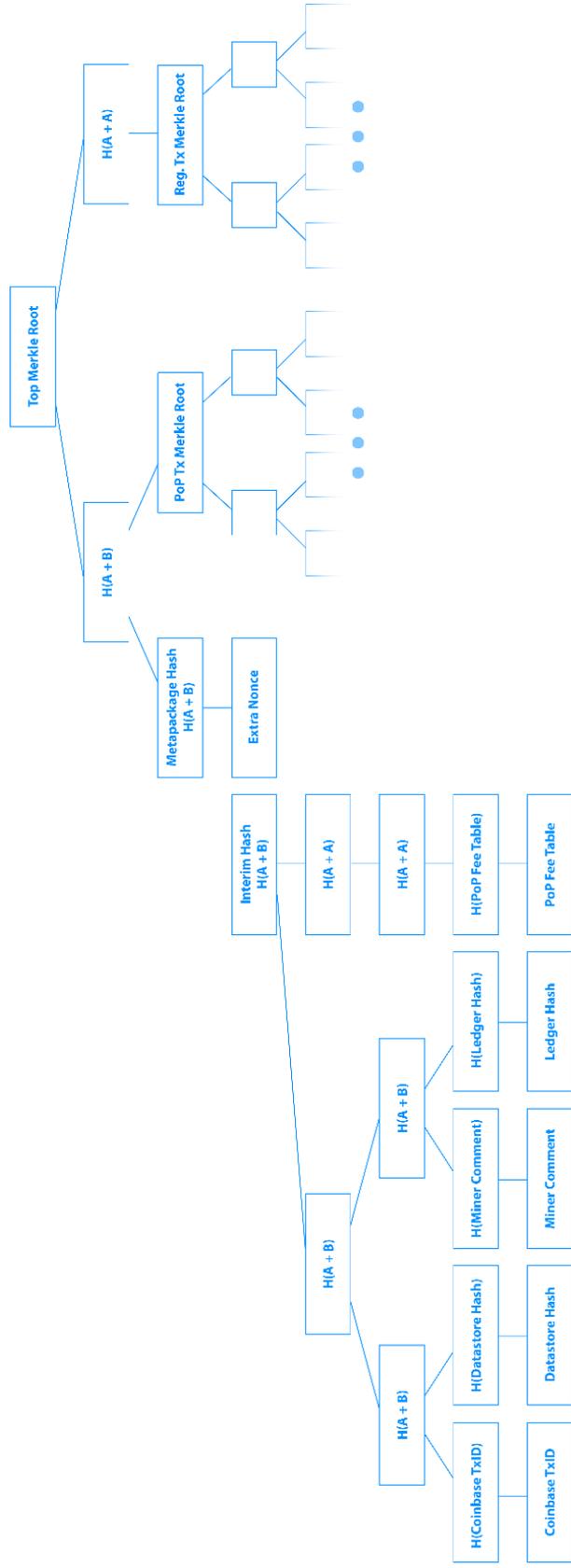
| Block Header Format (64 Bytes) |       |   |
|--------------------------------|-------|---|
| Size (Bytes)                   | Type  | Data                                      |
| 4                              | int32 | Block Height                              |
| 2                              | int16 | Version                                   |
| 12                             | bytes | Previous Block Hash (End)                 |
| 9                              | bytes | First Previous Keystone Block Hash (End)  |
| 9                              | bytes | Second Previous Keystone Block Hash (End) |
| 16                             | bytes | Merkle Root Hash                          |
| 4                              | int32 | Timestamp                                 |
| 4                              | int32 | Difficulty                                |
| 4                              | int32 | Nonce                                     |

Or visually (block #22 on the VeriBlock mainnet used for illustration purposes):



The PoW Target is encoded using the same ‘nBits’ compressed target format used by Bitcoin<sup>[2]</sup>. The bytes of the previous block hash, previous keystone hash, and second previous keystone hash are the ending bytes of the hash (the minimum PoW difficulty of the network ensures a minimum number of zeros at the beginning of the hash which ensure significantly more protection against collisions than 9 or 12 bytes would ordinarily provide).

Similar to Bitcoin’s Merkle root, the VeriBlock top-level Merkle root is the only piece of data directly resulting from the contents of the block itself:



## 8.7 Difficulty Algorithm

In order to ensure that an attacker with a high hashrate can't produce blocks at a fast enough rate to significantly outrun the legitimate network blocks (and thus publish VBK block headers to Bitcoin significantly earlier than the "legitimate" block headers of the public network would be published to Bitcoin which would make it possible for the attacker to beat the Bitcoin finality delay guarantees), the properties of the difficulty algorithm were carefully selected.

The difficulty algorithm is closely based on LWMA<sup>[3]</sup>. It has a retarget period of 100 blocks, which allows rapid adjustments to difficulty increases. As a result, even an attacker controlling orders of magnitude more vBlake hashing power than the VeriBlock network will be unable to make a fork which produces blocks early enough to cause the PoP score of blocks on the "legitimate" public VeriBlock chain to be zero.

## 8.8 Segwit-Like Transactions

In order to eliminate the possibility of signature malleability being used to manipulate TxIDs, VeriBlock TxIDs are calculated based off an encoded format containing only the source address, source amount, destination addresses and amounts, signature index, and "extra data" (the "OP\_RETURN" equivalent on VeriBlock, which is where altchain PoP miners put their PoP payloads).

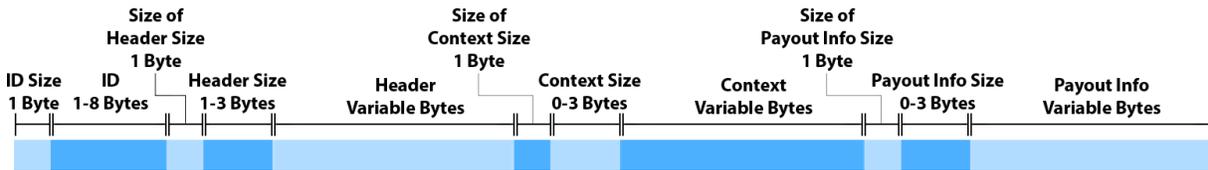
This protects users from signature malleability attacks which could trick them into thinking a transaction did not complete successfully, which is helpful to any software performing transactions on VeriBlock, including altchain PoP mining.

## 8.9 Altchain Data Publication Format and ID Tracking

On Bitcoin, it is impossible to verify that a transaction does *not* exist in a block without analyzing the entire block. As a result, Early Attack Detection for VeriBlock needs to digest every transaction on the Bitcoin blockchain to determine whether it has any implications for VeriBlock's PoP consensus.

In order to make PoP easier to implement for altchains, each altchain adopts a unique identifier which all PoP transactions done on its behalf must carry. When VeriBlock blocks are produced, they contain a metapackage which records the number of PoP transactions which are relevant to that altchain which exist in that particular VeriBlock block. As a result, providing the Merkle proof that the altchain's ID pairs to a particular number in the metacontent package along with that number of Merkle proofs of transactions containing the altchain ID provides full knowledge of information relevant to altchain consensus without requiring exhaustive searches of all VeriBlock blocks.

Altchains which wish to take advantage of this ID tracking will need to serialize their PoP data in the following format:



Note that the context and payout info sections can both be omitted (size of context size and size of payout info size both set to zero). The maximum total size of the serialized payload is 65535 bytes.

### 8.10 Alternate PoP Data Encoding Format in Bitcoin

The way the VeriBlock blockchain looks for data in a Bitcoin transaction is designed such that changes to the Bitcoin scripting language functionality, modifications to the set of standard transaction script formats for relay, changes or removal of OP\_RETURN, introduction of new signature schemes, etc. do not require VeriBlock to hard-fork to interpret the new data.

The algorithm for extracting VeriBlock data from a Bitcoin transaction first looks for a 64-byte contiguous section of the provided Bitcoin transaction which contains a valid VeriBlock block header (which hashes to a value below its embedded target), and if found will use the discovered header as the PoP-published block header, and the following 16 bytes as the miner identification data.

In the event that a contiguous valid block header is not found, an alternate encoding format is used. This encoding format uses a magic number to indicate the start of the encoding guide, which specifies how many chunks the data is broken up into, where each chunk is, and the length of each chunk. The encoding guide, when interpreted, produces a list of skips and reads (start at the beginning of the raw transaction, skip  $n_1$  bytes, read  $m_1$  bytes, skip  $n_2$  bytes, read  $m_2$  bytes, ... skip  $n_{num\_chunks}$  bytes, read  $m_{num\_chunks}$  bytes).

First, The algorithm searches for the magic bytes 0x927A59. This magic value was selected as it is tied for the fewest organic occurrences of any 24-bit sequence in transactions on the Bitcoin blockchain as of December 2018. It should be noted that false-positives of the magic bytes do not result in any consensus problems for the VeriBlock blockchain, and their (likely unintentional) inclusion in a PoP-data-carrying transaction on Bitcoin will not prevent the transaction from legitimately publishing and proving PoP data; any false-positive magic number occurrences which aren't followed by a valid encoding guide which recovers a valid VeriBlock block header is ignored. If the transaction includes a valid encoding guide (prefaced by the magic bytes) later on, the transaction is still valid.

If the magic bytes are found, the next 4 bits are read and interpreted as an unsigned 4-bit integer which represent the number of chunks the data is split up into ( $num\_chunks$ ). The

next 2 bits are read and interpreted as an unsigned 2-bit integer which represents the length of each encoded offset distance descriptor (*offset\_descriptor\_size*), with the following lookup table representing the respective values:

| Encoded Offset Distance Descriptor Size Lookup Table |        |
|--|--------|
| Binary Value   | Offset |
| 00   | 4      |
| 01   | 8      |
| 10   | 12     |
| 11   | 16     |

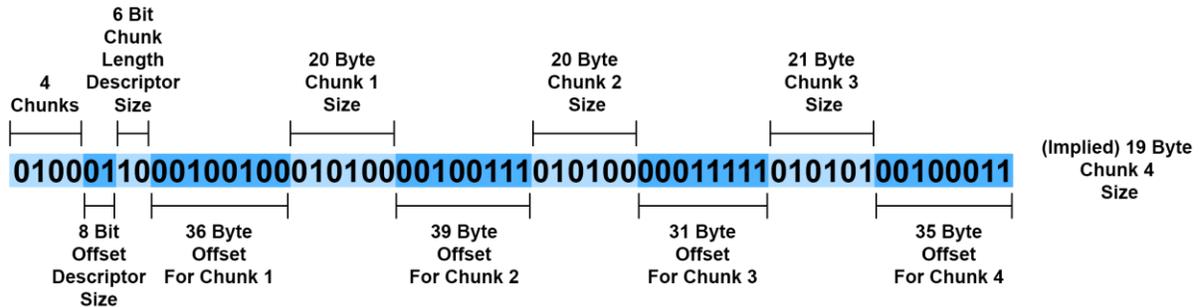
Then, the next two bits are read and interpreted as an unsigned 2-bit integer which specifies the length of each encoded chunk size descriptor (*chunk\_length\_descriptor\_size*), with the following lookup table representing the respective values:

| Encoded Chunk Length Descriptor Size Lookup Table |        |
|---|--------|
| Binary Value                                      | Offset |
| 00  | 4      |
| 01  | 5      |
| 10  | 6      |
| 11  | 7      |

Then, (*num\_chunks* - 1) times, the next *offset\_descriptor\_size* bytes are read and interpreted as a *offset\_descriptor\_size*-byte unsigned integer which represents the number of bytes to skip, and then the next *chunk\_length\_descriptor\_size* bytes are read and interpreted as a *chunk\_length\_descriptor\_size*-byte unsigned integer which represents the number of bytes to read for the chunk (*chunk\_size*). Then, the next *offset\_descriptor\_size* bytes are read and interpreted as an *offset\_descriptor\_size*-byte unsigned integer which represents the final number of bytes to skip before reading the final chunk, and the size of the last chunk is implied by the formula:

$$chunk\_size_{num\_chunks} = 80 - \sum_{i=1}^{num\_chunks - 1} chunk\_size_i$$

Here is a concrete example of an encoding guide which describes a skip-read list of: ***skip36,read20,skip39,read20,skip31,read21,skip35,read19.***



It should be noted that the data comprising the encoding guide can be accessed by the skip-read-list specified by the encoding guide.

In the special case of Bitcoin Segwit transactions (or any other future transaction type where some data ‘required’ to properly validate the transaction is not included in the data used for TxID calculation), the transaction must be filtered down to only the bytes used for TxID calculation for VeriBlock PoP data validation purposes. As such, no part of the encoding guide, or the chunks the encoding guide points to, can be included in the filtered data.

Additionally, the VeriBlock blockchain network will not extract multiple endorsements from a single Bitcoin transaction, even if multiple endorsements are included. Once the VeriBlock BTC transaction search/validation algorithm finds a valid VeriBlock header (along with corresponding PoP miner payout information), it stops. This prevents game theory issues where PoP miners could delay publication of a VeriBlock block header until they have two (or more) headers they believe they will benefit from publishing, and include these multiple header publications in a single transaction.

If a Bitcoin transaction contains one or more contiguous 64-byte segments which constitute a valid VeriBlock header, the first header published in the transaction is the only one the VeriBlock network will accept as being endorsed (and any headers published using an encoding guide are ignored). If no valid contiguous 64-byte segments are a valid VeriBlock header, then the first encoding guide which describes a valid 64-byte VeriBlock block header will be used to extract the endorsed header, and any subsequent encoding guides, regardless of described header validity, are ignored.

## 9 Integrating VeriBlock Security Into Altchains

Altchains will need to store and validate the PoP data altchain PoP miners produce (both altchain PoP transactions and their associated context, and VeriBlock PoP transactions and their associated context). This allows the altchain to have a complete SPV-level view of the VeriBlock (and by extension, Bitcoin) blockchains which it can reference in the event that an altchain fork is proposed.

Two types of the integration are possible, and they are compatible with each-other on a consensus level (so moving from one type of integration to the other does not require a hard/soft fork). The first is a standalone service/process which handles calls over gRPC through a bound port, and the second is an integration library built in the native language of the blockchain daemon. Both serve the same purpose, and are collectively referred to as the “VeriBlock integration point.”

At a high level, altchains will need to introduce a new transaction type which carried Proof-of-Proof data created by Altchain PoP miners, be able to validate the data contained within these transactions, be able to store these transactions in a protected section of their blocks (which doesn't count towards the blocksize limitations enforced on altchain PoW miners), be able to reward PoP miners based on the data contained in these transactions, and be able to reference the views of VeriBlock and Bitcoin provided by the data in these transactions when resolving forks.

| Summary of SI Blockchain Modifications by Section |   |
|---|---|
| Section   | Modification(s)   |
| 9.1   | The altchain must decide on the PoP security parameters discussed in chapter 6  |
| 9.2   | The blockchain's difficulty algorithm must prevent a high-hashrate attacker from producing blocks fast enough to violate the finality delay of the chain with respect to VeriBlock publication height                       |
| 9.3   | One or two (depending on security goals) new transaction types/script format(s) must be introduced to store PoP payloads, and the block size calculation algorithm should not include these transactions in its calculation |
| 9.4   | The coinbase transaction must provide PoP payouts to altchain PoP miners  |
| 9.5   | The block validation code must be updated to account for any block structure changes made to accommodate PoP, as well as to validate all altchain PoP transactions  |
| 9.6   | The code which resolves which fork to take in the event of multiple proposed versions of SI blockchain history must use ATVs to determine the better fork   |

### 9.1 Altchain PoP Parameters

As covered in chapter 6 (with a worked-out version for VeriBlock's own selection of these parameters in chapter 7), several PoP parameters have to be decided on, based on the altchain's security timeliness requirements, minimum tolerable publication frequency, and block time.

The following parameters need to be decided on:

- keystone\_interval
- num\_referenced\_keystones
- keystone\_finality\_delay
- pop\_reward\_settlement\_interval
- pop\_reward\_payment\_delay
- fork\_resolution\_relative\_score\_lookup\_table
- pop\_relative\_score\_lookup\_table
- Per-block PoP payout reward curves

### 9.2 Difficulty Algorithm

For blockchains where the difficulty of block creation is governed by a difficulty adjustment algorithm based on the timestamp of blocks (ex: PoW), the algorithm needs to be tweaked

to ensure an attacker can't produce blocks quickly enough to violate the *keystone\_finality\_delay* of the altchain. The fork resolution algorithm of the altchain will weight altchain keystone blocks based on the index of publication in VeriBlock. The difficulty algorithm should ensure that attackers (even those with orders of magnitude more hashing power than the main network) are unable to produce blocks fast enough to publish them to VeriBlock early enough such that blocks from the main chain would lose continuity.

### **9.3 New Transaction Types / Script Templates, Modified Block "Size" Calculation**

In order for an altchain to secure itself to VeriBlock, it must track the current state of the VeriBlock blockchain (and by proxy, the Bitcoin blockchain), and be able to validate proofs from altchain PoP miners that a particular altchain block header (+ any additional information) was published to VeriBlock. To facilitate this, one or more new transaction types need to be introduced. As a result, the altchain will maintain (delayed) SPV-level knowledge of VeriBlock and Bitcoin, constructed entirely from data returned by altchain PoP miners.

Two types of data payloads exist—VeriBlock-to-Bitcoin proofs (*VTBs*), and Altchain-to-VeriBlock proofs (*ATVs*). *VTBs* provide sufficient data for the altchain to track the state of the VeriBlock and Bitcoin blockchains with SPV-level security, including full PoP weighting fork resolution for VeriBlock. *ATVs* prove that a transaction exists in a VeriBlock block which is plausibly part of the main chain (incorporated into a block whose header builds on a set of zero or more headers that connect to the "known" VeriBlock chain according to a previous view of the VeriBlock chain known by the altchain based on *VTBs* and, in some cases, the VeriBlock block header context contained in *ATVs*).

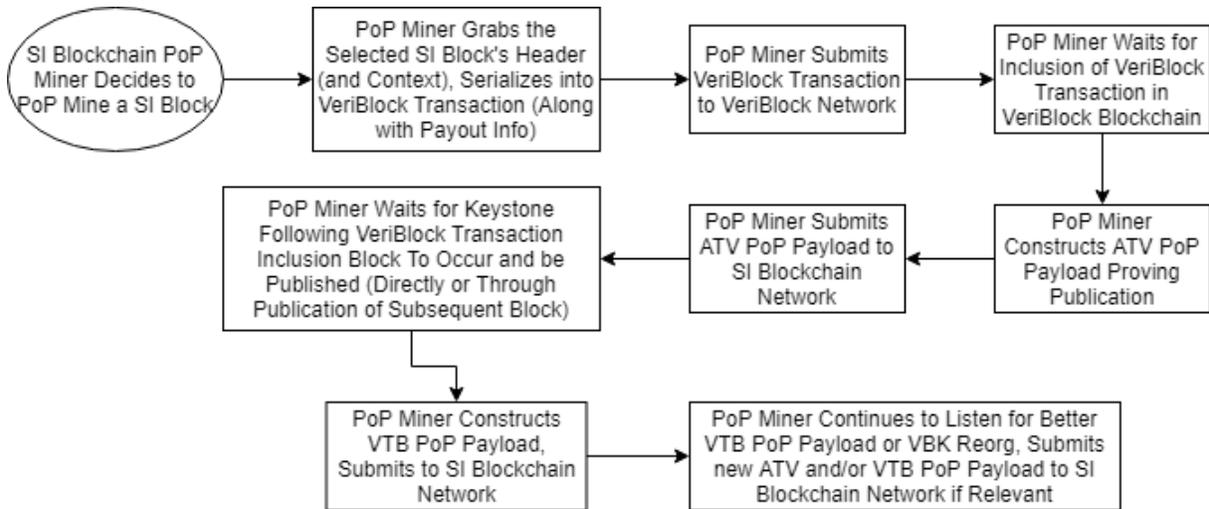
The content of *ATV* and *VTB* payloads can be considered as a black-box from the altchain's perspective: all of the work required to process them and maintain the SPV-level view of VeriBlock is done by the integration point on behalf of the altchain.

Altchains may either opt to bundle *ATV* and *VTB* payloads together in a single altchain PoP transaction (such that an altchain PoP miner doesn't return any data to the altchain until a *VTB* which publishes a VeriBlock keystone period the *ATV* proves to is available), or introduce two altchain PoP transactions; one which returns an *ATV* as soon as it is available, and another which follows up with one or more *VTB* publications which provide Bitcoin-based security context for the VeriBlock keystone period which includes the *ATV* transaction.

Having a single transaction type which contains both the *ATV* and one or more *VTBs* (as needed) provides an implementation complexity reduction, along with small space savings (the "scaffolding" for an additional transaction isn't required). Alternately, having separate *ATV* and *VTB* PoP transactions allows the altchain to use VeriBlock's native PoW as a middle-ground protection against reorganization attacks more quickly (particularly if the altchain *fork\_resolution\_publication\_delay\_latency\_lookup\_table* is aggressive enough that the

weighting cliff “often” ends before a VeriBlock keystone period finishes, publications of it get to Bitcoin, and the proof is returned to VeriBlock).

For the two-transaction-type implementation, the general workflow for an altchain PoP miner is as follows:



#### 9.4 Altchain PoP Payouts

In order to pay out rewards to the altchain PoP miners, some type of blockchain-enforced payout mechanism must exist. Generally, this will take the form of a portion of the coinbase transaction paying out new coins (plus delayed fees) to PoP miners. As discussed in section 6.6, the payout for PoP miners who successfully endorsed block  $n$  will occur at block  $n + pop\_reward\_payment\_delay$ . A coinbase transaction is only valid if it contains the correct PoP payouts based on altchain PoP transactions which are in the altchain before block  $n + pop\_reward\_settlement\_delay$ .

#### 9.5 Additional Block Validation Rules

When a new block is added onto the altchain, all of the PoP transactions (carrying either ATV, VTB, or ATV + VTB payloads) must be checked for validity.

ATV and VTB payloads have their own internal consistency rules independent of network state (provided block headers are contiguous with one another and validate at their encoded difficulties, Merkle proofs are correct, etc.), which are handed off to the integration point. Additionally, ATV and VTB payloads have additional validity requirements based on the current view of the VeriBlock blockchain held by the integration point (which is based on previous VTB payloads handed to the integration point by PoP transactions in previous altchain blocks, except in the trivial case of the first VTB on the altchain network connecting

to the hard-coded VeriBlock starting header). VTB payloads must provide VBK block headers which connect to the previously-known VBK chain, and the included VeriBlock PoP transactions must contain sufficient Bitcoin headers to connect to the known state of the BTC chain. ATV payloads must connect to the known VeriBlock chain. These validations are also handled by the integration point.

It follows that these transaction validity rules for transactions in a block should also be applied to transactions a miner attempts to put into a block; new altchain PoP transactions propagating on the altchain network should be validated in a similar fashion before being allowed entry into the mempool, and because a reorganization of the altchain can cause the integration point's delayed SPV view of VeriBlock and Bitcoin to change, reorganizations should trigger a re-validation of all pending altchain PoP transactions in the mempool (unless VTB and ATVs are required to connect to the VeriBlock SPV view at the time of the PoP'd altchain block, at which point the only validity check would be that the endorsed altchain block is still part of the main chain).

As previously stated, PoP transactions should not count towards the blocksize / gas limit that altchain block miners are beholden to. As a result, the block validation rules should also be tweaked accordingly, in addition to being updated to handle any format changes made to the block to accommodate PoP transactions.

## **9.6 Altchain Fork Resolution**

When an altchain fork occurs over the boundary of at least one keystone period, the correct chain is chosen based on the PoP scores of the two chains. As such, the altchain must implement the algorithms described in sections 6.3-6.5, including an efficient way to quickly gather all altchain PoP publications containing ATVs of blocks in each keystone period for any chain fork.

In order to ensure the SPV view of VeriBlock (and by extension, Bitcoin) correctly reflect knowledge contained in both forks, the altchain integration point offers a mechanism of temporarily adding data to the view. From the block height of divergence for the two chains, any VTBs in the chain which isn't currently the main chain must be provided to the integration point in the "temporary" addition format, and then cleared after the correct fork is determined. If the fork resolution algorithm determines that the currently-applied chain is inferior, the current chain will be unapplied (unapplying all of the changes VTB and ATV payloads in the current chain back to the forking point, and then applying all of the VTB and ATV payloads in the new chain).

## 10 False EAD Triggering Prevention With PoW Proofs

Because not all headers of a SI chain are required to be published to an SP chain, an attacker on a Proof-of-Work SI chain with less than 51% of the SI chain's hashing power could produce publications to the SP chain which signal an attack. While the attacker will never be able to actually cause a reorganization, these false-positive warnings for Early Attack Detection could be used as a means to cause disruption to services built around the SI blockchain. The exact amount of hashing power required to generate these false positives depends on the required frequency of full PoW-valid headers (which are a result of keystone  $i$  and  $r$  values).

In order to force an attacker to control and utilize a significant portion of the SI chain's hashrate to generate valid EAD warnings, PoW Proofs can be used. These consist of a number of extra headers included in a blockchain's PoP publications which demonstrate statistical evidence that a sufficient amount of Proof-of-Work has been done to generate an equally valid blockchain.

For an SI chain publishing to VeriBlock, these extra headers can be published in the `context_info` section of the Altchain PoP Endorsement.

### 10.1 Background

In Proof-of-Work, the target refers to the maximum value a resultant hash can have and still be a valid solution. For example, if a theoretical hashing function outputs random numbers between 0 and 999 inclusive (base-10 numbers used for simplicity), and the target is 100, then inputs which produce a hash output [0, 100) will be valid solutions.

However, any solution with a hash below 100 is "above difficulty," meaning that the solution would have also been valid at a higher difficulty (lower target). Because of the uniform distribution of hashing functions, any solution between 0 and 100 is equally likely, despite a solution of 10 being valid at a roughly 10x difficulty (thus, having a *difficulty multiple* ( $dm$ ) of 10). As a result, every PoW solution except one (which exists exactly at the target) will actually have a higher valid difficulty than the network requires.

For the range [0, 100) with a *target* ( $tgt$ ) of 100, solutions will have a  $dm$  ranging from 1 to 100 (ignoring the solution 0, which has an infinite  $dm$ ), and an *expected difficulty multiple* ( $edm$ ) of  $\sim 2.9290$ .

As the range grows larger, the  $edm$  grows logarithmically. The  $edm$  can be calculated as follows:

$$edm = \frac{\sum_{x=1}^{x=tgt} \frac{tgt}{x}}{tgt} = \sum_{x=1}^{x=tgt} \frac{1}{x}$$



The probability that at least  $n$  out of  $m$  blocks occur with  $dm \geq mindm$  is:

$$\sum_{k=n}^m \binom{m}{k} \left(\frac{1}{mindm}\right)^k \left(1 - \frac{1}{mindm}\right)^{m-k}$$

## 10.2 PoW Proofs Introduction

PoW Proofs take advantage of the properties explored in section 10.1 by allowing the publication of a small number of recent headers with high  $dm$  values to demonstrate with high statistical probability that sufficient work was done to generate a particular number of blocks at a given difficulty. Other technologies like NiPoPoW<sup>[6]</sup> use similar properties of Proof-of-Work to make more efficient probabilistic PoW validation.

A normal Proof-of-Work chain at difficulty  $d$  would expect to have 50% of its blocks with a  $dm$  greater than or equal to 2, 25% of its blocks with a  $dm$  greater than or equal to 4, etc. As a result, two valid PoW solutions with  $dm \geq 4$  are convincing evidence that 8 valid PoW solutions were generated at difficulty  $d$  (in order for an attacker to create these two PoW solutions where  $dm \geq 4$ , they would, on average, have to do the same amount of work as to generate 8 blocks at difficulty  $d$ ).

As a result, a number of *top blocks* ( $topb$ ) based on their  $dm$  out of a number of *total blocks* ( $totb$ ; generally the maximum number of block headers which can be skipped while maintaining full publication continuity of the SI chain) can be selected and have their full PoW solution (generally the complete header) published.

Because the *topb* blocks are not guaranteed to have the appropriate  $dm$  values, a failure of *topb* blocks to satisfy the requirements will require alternate proofs, covered in section 10.4.

## 10.3 Parameter Selection

Selection of *topb* and *mindm* variables depend on *totb* and poses a trade-off between the amount of extra data published to the SP chain and the percentage of the hashing power of the SI blockchain an attacker requires to trigger EAD.

The percentage of hashpower an attacker needs to falsely trigger EAD is based on the following ratio:

$$\text{false\_EAD\_hashpower\_requirement} = \frac{topb * mindm}{totb}$$

The following table demonstrates the probabilities that at least the given *topb* number of blocks in a period of *topt* will have the specified *dm* values:

| Color                            |     |     |     |     |      |
|----------------------------------|-----|-----|-----|-----|------|
| Probability (minimum, inclusive) | 0%  | 60% | 70% | 80% | 95%  |
| Probability (maximum, exclusive) | 60% | 70% | 80% | 95% | 100% |

| Probability of <i>topb</i> Out of <i>topt</i> Blocks Meet <i>dm</i> Threshold |    |                                   |         |         |         |         |                 |         |         |         |         |
|---|----|-----------------------------------|---------|---------|---------|---------|-----------------|---------|---------|---------|---------|
|   |    | Difficulty Multiple ( <i>dm</i> ) |         |         |         |         |                 |         |         |         |         |
|   |    | <i>topb</i> = 1                   |         |         |         |         | <i>topb</i> = 2 |         |         |         |         |
|   |    | 2                                 | 5       | 8       | 10      | 15      | 2               | 5       | 8       | 10      | 15      |
| Blocks in Period ( <i>topt</i> )  | 5  | 96.875%                           | 67.232% | 48.709% | 40.951% | 29.175% | 81.250%         | 26.272% | 12.073% | 8.146%  | 3.881%  |
|   | 8  | 99.609%                           | 83.222% | 65.639% | 56.953% | 42.417% | 96.484%         | 49.668% | 26.370% | 18.690% | 9.512%  |
|   | 10 | 99.902%                           | 89.263% | 73.692% | 65.132% | 49.839% | 98.926%         | 62.419% | 36.110% | 26.390% | 14.009% |
|   | 12 | 99.976%                           | 93.128% | 79.858% | 71.757% | 56.304% | 99.693%         | 72.512% | 45.330% | 34.100% | 18.850% |
|   | 15 | 99.997%                           | 96.482% | 86.510% | 79.411% | 64.474% | 99.951%         | 83.287% | 57.592% | 45.096% | 26.410% |
|   | 20 | 99.999%                           | 98.847% | 93.079% | 87.842% | 74.839% | 99.998%         | 93.082% | 73.305% | 60.825% | 38.894% |
|   | 25 | 99.999%                           | 99.622% | 96.450% | 92.821% | 82.180% | 99.999%         | 97.261% | 83.772% | 72.879% | 50.357% |

As can be seen in the above table, the same *false\_EAD\_hashpower\_requirement* values can be achieved with slightly higher probability (and thus, efficiency) with lower *topb* values. However, many *false\_EAD\_hashpower\_requirement* options are not available with *topb=1* (particularly in the case of low *topt* values), so selecting *topb* > 1 may prove useful.

For a concrete example, a blockchain with properties: {*totb=20, topb=1, mindm=8*} will have a *false\_EAD\_hashpower\_requirement* of 40% (so an attacker wishing to falsely trigger EAD would need 40% of the hashrate the legitimate chain is built with). Additionally, it will have a 93.079% chance of each 20-block period producing a valid PoW Proof (which requires publication of *topb=1* additional headers), meaning 6.921% of publications will not be able to generate a valid proof (and require alternate proof, see section 10.4).

#### 10.4 Alternate Proofs When *mindm* Isn't Met by *topb* Blocks

Not every period of *totb* blocks will generate *topb* blocks which meet or exceed the prescribed *mindm* value (except in the trivial case where *mindm* = 1.0), so a backup mechanism needs to exist.

The simplest backup mechanism is to require the PoP miner to publish all of the headers in the *topb* period (note that only requiring publication of *topb* \* *mindm* blocks is insufficient as it lowers the challenge to producing either *topb* blocks at *mindm* OR *topb* \* *mindm* blocks at *dm*=1). However, this does produce large proofs; requiring *topb* headers to be published.

Another potential backup mechanism is to use zero-knowledge proofs, covered in section 10.6.

#### 10.5 [Experimental] Dynamic Keystoning

Altchains can also implement a dynamic form of keystoning, where rather than keystone blocks occurring at a scheduled interval, keystone blocks are instead those which happen to have a high difficulty multiple. Because of the ability of an attacker with a significantly higher hashrate than the entire network to produce keystones at a more frequent interval if keystoning is based off of the *dm* value of a particular block, rules regarding the minimum permitted number of blocks between acknowledged keystones should be introduced, such that the worst-case keystone production speed is unable to cause the finality issues discussed in sections 6.4, 7.5, and 9.6.

Although the keystone block intervals will be unpredictable, the average frequency of keystones given *mindm* is:

$$p = \frac{1}{mindm}$$

And the probability of an interval taking more than *interval\_length* blocks is:

$$p = \left(1 - \frac{1}{mindm}\right)^{interval\_length}$$

Blockchains also must enforce a *maximum interval length* (*max\_interval\_length*) which prevents an attacker from forging "long" chains with few publications. The chain is protected against an EAD-triggering attacker having less than *mindm*/*max\_interval\_length* hashing power relative to the main network.

For example, a blockchain could define a keystone as any block with *dm* > 10, and enforce a *max\_interval\_length* of 50 blocks. The probability that a period of 50 blocks would pass without having one block with *dm* >= 10 is approximately 0.515%, and would cause the

network to ‘freeze’ while trying to mine the 50<sup>th</sup> block at  $dm \geq 10$  for, on average, 10 additional SI blocktimes. To falsely trigger EAD, an attacker would need to control  $10/50 = 20\%$  of the network’s hashrate.

## 10.6 [Experimental] Zero-Knowledge Proofs

Altchains with Proof-of-Work algorithms simple enough to be implemented (practically) in a PoW verification circuit which zero-knowledge proofs can demonstrate a prover has appropriate headers such that, when provided to the function, are all contiguous and all satisfy the relevant PoW difficulty proofs may be able to utilize zero-knowledge proofs to attest to the existence of a number of valid Proof-of-Work solutions without requiring the publication of all of them.

Zero-Knowledge Proofs provide better anti-false-EAD triggering characteristics than publishing a small number of high- $dm$  block headers as probabilistic proof, particularly considering the concessions required to make high- $dm$  constructions work without requiring alternate proofs often. However, the protection against anti-false-EAD triggering is strictly worse than that provided by requiring the publication of all block headers in the relevant period, because of the additional hardness assumptions which protect against false proofs.

The zero-knowledge circuit would, given an input of a number of headers, confirm that each provided header hashes to a result which is below a particular specified target, that each header builds on the previous header’s hash, and that the final header is the one previous to the plaintext header that the PoP miner is endorsing.

In systems where the security profile of zk-SNARKs are tolerable (trusted setup<sup>[7]</sup> and the KEA1<sup>[8]</sup> assumption), zk-SNARKs should provide small-footprint proofs that sufficient Proof-of-Work power was used to build a chain without requiring publication of all of the headers.

Bulletproofs<sup>[9]</sup> may prove a useful alternative to zk-SNARKs where the trusted setup of zk-SNARKs is unacceptable, and only the hardness of the discrete log problem is required. Bulletproofs will be larger than zk-SNARKs, but likely smaller than whole-header publication.

The large proof sizes of zk-STARKs<sup>[10]</sup> likely renders them uninteresting for PoW Proofs; it will generally be cheaper to publish all of the headers (particularly because the “zero knowledge” part is unimportant for our uses; we only care about the succinctness of proofs being more space-efficient than publishing all of the headers).

It should be noted that the security failure of the zero-knowledge proof used for PoW Proofs will only reduce the blockchain’s security against false EAD triggering; it has no bearing on the ability to actually perform a reorganization (because all of the actual blocks would need to be broadcast to the network), nor any bearing on the soundness of the altcoin’s money supply.

## Appendix A: Summary of VeriBlock Blockchain Parameters

| VeriBlock Blockchain Parameters |                   |   |
|---------------------------------|-------------------|---|
| Parameter                       | Setting           | Motivation  |
| Block Time                      | 30 Seconds        | Balance between propagation time and Bitcoin finality   |
| Bitcoin Finality Delay          | 11 Bitcoin Blocks | Probability of 20 VBK blocks occurring before finality delay closure is extremely small, and a 30% Bitcoin hashrate controller would have to try tens of thousands of attempts at censoring VBK PoP transactions to succeed for the entire finality delay period. |
| Minimum Difficulty              | 900,000,000,000   | This difficulty ensures that the shorter hashes encoded in the VeriBlock header have a higher degree of protection against collision attacks. In practice, this variable is unlikely to be relevant because it represents a hashrate of ~30 GH/s.                 |

## Appendix B: VeriBlock Integration Point

Altchains adopting Bitcoin security through VeriBlock will use the VeriBlock Integration Point to record and maintain SPV-level knowledge of VeriBlock (and by extension, Bitcoin). *VTB* payloads contained in PoP transactions on the altchain (discussed in section 9.3) are provided to the VeriBlock integration point, from which it constructs the SPV view. During fork resolution over a keystone boundary, the altchain will consult the integration point to determine the timeliness of publications of its keystone periods to the VeriBlock blockchain.

During normal operation, the integration point maintains a deterministic SPV-level view of VeriBlock based entirely on the *VTB* publications it has been provided. It maintains an association of these *VTB* payloads with their enclosing altchain block (by block number), so that altchain reorganizations can undo the state changes the unapplied blocks made to the integration point's view of VeriBlock.

During fork resolution, the integration point is temporarily updated with *VTB* information contained within the challenging fork to ensure the integration point's view includes information stored in both forks. After fork resolution, the altchain sends a command to the integration point to clear the temporarily-added *VTB* payloads, which returns the integration point's view to one which only reflects *VTB* payloads in the main chain. In the event that the challenging fork is selected, the altchain still sends the clear command, and then un-applies *VTB* payloads in the previously-main chain back to the forking point, and applies the new *VTB* payloads from the new fork.

The integration point supports the following commands:

### checkATVInternally

**Description:** Runs internal validity checks on altchain-to-VeriBlock publication.

**Expected Use:** Run on ATV payloads by the altchain for PoP transaction validity

**Notes:** Internal validation checks performed:

- The provided VeriBlock transaction is formatted correctly
- The VeriBlock transaction is signed correctly
- The Merkle root proves the transaction exists in the claimed VeriBlock block header
- The claimed VeriBlock block header hashes correctly according to its encoded difficulty, as do any VeriBlock context headers (if provided)
- Context VeriBlock headers (if provided) are all contiguous

| Argument       | Required | Description                               |
|----------------|----------|---|
| ATV ATVToCheck | TRUE     | The ATV to perform internal validation on |

### checkVTBInternally

**Description:** *Runs internal validity checks on a VeriBlock-to-Bitcoin publication.*

**Expected Use:** *Run on VTB payloads by the altchain for PoP transaction validity*

**Notes:** *Internal validation checks performed:*

- *The provided VeriBlock PoP transaction is formatted correctly*
- *The VeriBlock PoP transaction is signed correctly*
- *The embedded Bitcoin transaction in the PoP transaction is formatted correctly*
- *The Bitcoin Merkle root proves the Bitcoin transaction is in the claimed Bitcoin block header*
- *All included Bitcoin block headers hash correctly according to their encoded difficulty*
- *All included Bitcoin block headers are contiguous*
- *The VeriBlock Merkle root proves the PoP transaction exists in the claimed VeriBlock block header*
- *The claimed VeriBlock block header hashes correctly according to its encoded difficulty, as do any VeriBlock context headers (if provided)*
- *Context VeriBlock headers (if provided) are all contiguous*

| Argument       | Required | Description                               |
|----------------|----------|---|
| VTB VTBToCheck | TRUE     | The ATV to perform internal validation on |

### addPayloads

**Description:** Adds one or more VTBs and/or ATVs to the integration point's SPV-level view of the VeriBlock (and by extension, Bitcoin) blockchain.

**Expected Use:** When an altchain is applying a block to the main chain (either normally, or during fork resolution after now-forked blocks have been unapplied), this method would be called with all VTBs and ATVs (if ATVs contain VeriBlock context headers) contained in altchain PoP transactions in the given block.

**Notes:** This command throws an error if the associatedAltchainHeight is less than the highest altchain height the integration point is aware of. If an altchain block 'n' does not contain any VTBs (or ATVs if relevant), this command does not need to be run. An error is also thrown if any of the provided VTBs or ATVs do not connect to the known VeriBlock and Bitcoin SPV views at the time of addition. Payloads are applied in the order provided, and all VTBs are processed before ATV processing begins. A payload can depend on knowledge provided by previous payloads in the same provided array, but not on later payloads.

| Argument                       | Required | Description  |
|--------------------------------|----------|--|
| int64 associatedAltchainHeight | TRUE     | The height of the SI block which contains the particular VTBs and/or ATVs provided |
| byte[] associatedAltchainHash  | TRUE     | The hash of the SI block which contains the particular VTBs and/or ATVs provided   |
| VTB[] VTBsToAdd                | FALSE    | Array of the VTBs to add   |
| ATV[] ATVsToAdd                | FALSE    | Array of the ATVs to add the VBK context from                                      |

## removePayloads

**Description:** Removes VTBs and ATVs (if applicable) which were associated with a particular altchain height.

**Expected Use:** When an altchain is unapplying a block which contained one or more VTBs or ATVs which were previously provided to the integration point using **addPayloads**.

**Notes:** This command throws an error if the altchain height to remove is greater than the highest altchain height the integration point knows about. If the most recent batch of VTBs (+ ATVs when applicable) was provided to the integration point with `associatedAltchainHeight=n`, then `removePayloads(n+1)` will fail. Additionally if `addPayloads` was called for `associatedAltchainHeight=m` and `removePayloads(m)` has not been called, then `removePayloads(m-1)` will fail. Also if a VTB or ATV was previously provided with multiple altchain indexes associated, the VTB's or ATV's effect on the integration point's view of VeriBlock won't be undone until all altchain indexes they were related to are undone (if altchain blocks 'n' and 'n+1' are both associated with a particular VTB, then the VTB's effect on the VBK view will only disappear if `removePayloads(n)` occurs).

| Argument                                    | Required | Description   |
|---|----------|---|
| int64 <code>associatedAltchainHeight</code> | TRUE     | The height of the SI block which contains the particular VTBs and/or ATVs to be removed |
| byte[] <code>associatedAltchainHash</code>  | TRUE     | The hash of the SI block which contains the particular VTBs and/or ATVs to be removed   |

### addTemporaryPayloads

**Description:** Temporarily adds one or more VTBs and/or the VBK header knowledge from one or more ATVs to the integration point's SPV-level view of the VeriBlock (and by extension, Bitcoin) blockchain.

**Expected Use:** When an altchain is performing fork resolution over a keystone boundary, it will add payloads temporarily which are present in the non-incumbent fork to ensure that the integration point's view of VeriBlock reflects all information available in both forks.

**Notes:** This command throws an error when the provided VTBs and/or ATVs do not connect to the current SPV view of VeriBlock and Bitcoin.

| Argument        | Required | Description                                   |
|-----------------|----------|---|
| VTB[] VTBsToAdd | FALSE    | Array of the VTBs to add                      |
| ATV[] ATVsToAdd | FALSE    | Array of the ATVs to add the VBK context from |

### clearTemporaryPayloads

**Description:** Clears the effects of all VTB and/or ATV payloads which were added using **addTemporaryPayloads**. This returns the SPV views of VeriBlock and Bitcoin back to the deterministic view based only on VTB (and ATV, when relevant) payloads in the current main altchain fork.

**Expected Use:** After the correct fork is determined by the altchain's fork resolution protocol when resolving forks which cross a keystone boundary, the temporarily-added payloads will be removed.

**Notes:** In the event that a reorganization does occur, clearing the temporary payloads will occur, followed by **removePayloads** for all altchain blocks after the forking point. The application of blocks from the forking point forward will cause calls to **addPayloads** for all of the VTBs (and ATVs, where relevant) in the new fork.

**No arguments**

### simplifyVTBs

**Description:** Given a list of VTBs, returns a reduced list of VTBs which communicate the “same” information regarding the state of the SPV-level view of VeriBlock and Bitcoin that the integration point provides. This allows altchains to de-duplicate VTBs which provide redundant information, keeping only the ones which provide the best VeriBlock-to-Bitcoin publication for each keystone period covered by the list of VTBs provided.

**Expected Use:** Altchains can de-duplicate VTBs that they store in their blockchain for space savings because some VTBs carry equivalent (or inferior to another available VTB) information.

**Notes:** VTBs are considered informationally equivalent if they contain the proof of the same VeriBlock block header to the same Bitcoin block. One VTB is considered strictly inferior to another if it either:

1. Expresses a VTB publication which published a VeriBlock block header to Bitcoin in a later Bitcoin block than another available VTB expresses a VTB publication of a VeriBlock block header in the same VeriBlock keystone period to
2. Expresses a VTB publication which published a VeriBlock header to the same (or a later) Bitcoin block which is after another VeriBlock block header in the same VeriBlock keystone period which another available VTB expresses publication to the same (or an earlier)

Additionally, VTBs are considered equivalent in the information they provide when they contain proof of the same VeriBlock header in a Bitcoin transaction to the same Bitcoin block, and a deterministic tie-breaker algorithm.

VTBs which publish VeriBlock headers from the same keystone period to different Bitcoin blocks in competing Bitcoin forks are retained.

| Argument                | Required | Description                      |
|-------------------------|----------|----------------------------------|
| VTB[] VTBSToDeduplicate | TRUE     | Array of the VTBS to deduplicate |

### checkATVAgainstView

**Description:** Checks that an ATV connects to the known VeriBlock view.

**Expected Use:** Run on ATV payloads by the altchain during altchain PoP transaction validity checks and during fork resolution

**Notes:** This command calls **checkATVInternally** on the provided ATV.

| Argument       | Required | Description   |
|----------------|----------|---|
| ATV ATVToCheck | TRUE     | The ATV to check for connection to the known VeriBlock view |

## References

- [1] Bitcoin Block Propagation Statistics [<https://bitnodes.earn.com/dashboard/?days=730>]
- [2] Bitcoin Difficulty and Target Encoding [<https://en.bitcoin.it/wiki/Difficulty>]
- [3] LWMA Difficulty Algorithm History [<https://github.com/zawy12/difficulty-algorithms/issues/24>]
- [4] Ethereum Merkle Patricia Trees [<https://github.com/ethereum/wiki/wiki/Patricia-Tree>]
- [5] Enabling Blockchain Innovations with Pegged Sidechains  
[<https://www.blockstream.com/sidechains.pdf>]
- [6] Non-Interactive Proofs of Proof-of-Work Paper [<https://eprint.iacr.org/2017/963.pdf>]
- [7] Zcash Trusted Setup [<https://z.cash/technology/paramgen/>]
- [8] The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols  
[<https://eprint.iacr.org/2004/008.pdf>]
- [9] Bulletproofs: Short Proofs for Confidential Transactions and More  
[<https://eprint.iacr.org/2017/1066.pdf>]
- [10] Aggregated “awesome-zero-knowledge-proofs” Repo [<https://github.com/gluk64/awesome-zero-knowledge-proofs>]