

# Bobtail: A Proof-of-Work Target that Minimizes Blockchain Mining Variance

George Bissias    Brian N. Levine

College of Information and Computer Sciences, UMass Amherst

## ABSTRACT

Blockchain systems are designed to produce blocks at a constant average rate. The most popular systems currently employ a Proof of Work (PoW) algorithm as a means of creating these blocks. Bitcoin produces, on average, one block every 10 minutes. An unfortunate limitation of all deployed PoW blockchain systems is that the time between blocks has high variance. For example, 5% of the time, Bitcoin’s inter-block time is at least 40 minutes. This variance impedes the consistent flow of validated transactions through the system. We propose an alternative process for PoW-based block discovery that results in an inter-block time with significantly lower variance. Our algorithm, called Bobtail, generalizes the current algorithm by comparing the mean of the  $k$  lowest order statistics to a target. We show that the variance of inter-block times decreases as  $k$  increases. If our approach were applied to Bitcoin, about 80% of blocks would be found within 7 to 12 minutes, and nearly every block would be found within 5 to 18 minutes; the average inter-block time would remain at 10 minutes. The cost of our approach is a larger block header.

## 1 INTRODUCTION

Blockchain systems are designed to produce blocks at a constant average rate. The most popular systems currently employ a *Proof of Work (PoW)* algorithm as a means of creating these blocks. Bitcoin [25] (and Bitcoin Cash [1]) produce, on average, one block every 10 minutes, and will self-adjust the difficulty of producing a block every two weeks if too many or too few have been produced. Unfortunately, a limitation of all deployed PoW blockchain systems is that the time between blocks has high variance and the distribution of inter-block times has a very long tail. For example, 5% of the time, Bitcoin’s inter-block time is at least 40 minutes. This variance impedes the consistent flow of validated transactions through the system.

The high inter-block time variance is a direct consequence of the PoW algorithms that are at the core of blockchains, including Bitcoin [1, 25], Litecoin [2], and Ethereum [17]. In all these systems, generally, the miners repeatedly craft block headers by changing a nonce until the hash of the header is less than a target value  $t$ . In other words, the hash of each header is a sample taken randomly from a discrete uniform distribution that ranges between  $[0, S]$ , where  $S = 2^b - 1$

and typically  $b = 256$ . A block is discovered when the *first order statistic* (i.e., the minimum value) of all sampled values is less than target  $0 < t < S$ .

In this paper, we propose an alternative process for PoW-based block discovery that results in an inter-block time with significantly lower variance. Our algorithm generalizes the current algorithm by comparing the mean of the  $k$  lowest order statistics to a target. We show that the variance of inter-block times decreases as  $k$  increases. For example, if our approach were applied to Bitcoin, about 80% of blocks would be found within 7 to 12 minutes, and nearly every block would be found within 5 to 18 minutes; the average inter-block time would remain at 10 minutes. The cost of our approach is a larger block header. We call our approach *Bobtail*<sup>1</sup> mining.

**Problem Statement.** Consider a fixed interval of time during which the entire network produces  $\theta$  hashes generating a sequence of hash values  $\mathbf{Z} = Z_1, \dots, Z_\theta$ . Let  $Z$  be an arbitrary random variable from the sequence  $\mathbf{Z}$ ; note that  $Z \sim \text{Uniform}(0, S)$ . Define  $V_i$  to be the  $i$ th lowest order statistic of  $\mathbf{Z}$ , i.e.  $V_i = Z_{(i)}$  in standard notation. And let random variable  $W_k$  be the mean of the  $k$  lowest order statistics:

$$W_k = \frac{1}{k} \sum_{i=1}^k V_i. \quad (1)$$

$W_k$  constitutes the collective mining proof (*proof*, for short) for the entire network. Our Bobtail mining criterion says that a new block is discovered when a realized value of  $W_k$  meets the target  $t$ :

$$w_k \leq t. \quad (2)$$

Notably, this approach is a generalization of current systems, which are the special case of  $k = 1$ .

Our primary goals are therefore to show, given values of  $k > 1$ , that: (i) there is a significantly reduced inter-block time variance; and (ii) the costs are relatively small, which include an increase in block header size and a slight increase in network traffic.

## Contributions.

- We derive the statistical characteristics of our approach and validate each empirically. For example,

<sup>1</sup>A *bobtail* refers to an animal’s tail that is unusually short or is missing completely [9].

we derive expressions for the expectation and variance of the Bobtail mining proof and the number of hashes performed for any value of  $k$ . Using these expressions, we quantify the reduction in variance of inter-block time expected for values of  $k > 1$ .

- We show that the variance in block discovery time from using  $k$  order statistics reduces by  $O(1/k)$ , compared to using  $k = 1$ .
- We derive a formula for adjusting the target values of existing blockchains so that values of  $k > 1$  can be adopted without changing the mean inter-block time. This formula allows our scheme to be adopted as a patch, rather than deployed as a new system.
- We show that the rate at which Bobtail miners create forks in the blockchain is no higher than Bitcoin and Ethereum. Bobtail’s rate is larger when applied to Ethereum but only if miners are purposefully and dishonestly mining after receiving a new block. Furthermore, we prove that Bobtail miners receive significantly lower rewards when attempting to dishonestly fork, due to a split rewards formula we’ve designed.
- We quantify the byte overhead of our approach. It will increase the size of the blocks by roughly  $35k$  bytes plus the cost of a  $k$ -output coinbase transaction. For example, for  $k = 40$ , Bitcoin blocks would increase by about 3KB (including the larger coinbase transaction), which is still small compared to the 1–8MB necessary to detail the transactions themselves (depending on the version of Bitcoin used). We also show that the additional network traffic is very small by use of a simple filter we have designed.

## 2 RELATED WORK

Our approach is related to previous results in proof-of-work, cryptographic puzzles, and improvements to blockchain architectures.

**Proof of work.** A large number of papers have explored applications of proof-of-work. Dwork and Naor [16] first suggested proof-of-work in 1992, applying it as a method to thwart spam email. A number of subsequent works similarly applied PoW to thwarting denial of service attacks [5, 12, 14, 19, 20, 26]. Our approach can be adopted into many of these past works to improve computational variance. Jakobsson and Juels [21] and Jules and Brainerd [22] examine the security properties of PoW protocols, and base their theorems on the average work required; our approach would provide stronger guarantees under their theorems since the variance is lower. Laurie and Clayton [23] examine the practical limitations in deploying PoW solutions in DoS scenarios.

Douceur [15] noted in 2002 that proofs of work can mitigate Sybil attacks. Also in 2002, Back [6] applied PoW to

cryptocurrencies. Back noted the high variance of computational PoW and regarded it as an open problem. Nakamoto’s Bitcoin [25] built on these ideas.

In 2003, Abadi et al. [4] suggested memory-bound functions as a better foundation for avoiding the variance in CPU resources among users. Indeed, the ETHASH [3] PoW algorithm in Ethereum [17] adopted a PoW function that requires more memory than is economically profitable for custom ASICs. In contrast, our goal is to reduce the variance of the entire network’s time to solve a PoW problem, and it is not to increase egalitarianism or increase participation by eschewing specialized hardware. In any case, our approach is applicable to ETHASH.

Coelho [13] is the work is closest to ours in terms of goals. That work proposes a PoW puzzle based on Merkle trees that requires an exact number of steps and therefore has no computational variance. The paper is focused on mitigating denial of service attacks, and because it was not designed for use in blockchains it has a number of disadvantages in that context. Primarily, the amount of network traffic is on the same order as the amount of computation. As an example, for Bitcoin’s current difficulty of  $8e18$  hashes/second, the Coelho proof of work would add 4.6 Mbytes to the blockheader. The header size would increase further as Bitcoin’s difficulty increases. While our variance is not zero, our increase to the block header size is only a few kbytes and depends on only the decrease in variance desired.

**Bitcoin-NG [18]** was designed to allow Bitcoin’s rate of validated transactions per second to scale to a higher rate. To do so, the miner of the most recent Bitcoin-NG block acts as an elected leader until the next block is discovered. While the leader, a miner will issue validated transactions with sole authority in microblocks. The leader election time distribution has the same large variance as other PoW-based blockchain systems — though not the design goal, we note that Bitcoin-NG can issue validated transactions with low variance. Unfortunately, Bitcoin-NG has never been deployed, perhaps because of two of its limitations. First, Bitcoin-NG introduces the possibility that transactions will stop altogether if the elected leader is disconnected from the network (or elects to maliciously stop), which is not a problem shared by Bitcoin. Secondly, the position of authority allows the leader to double spend more easily; this is mitigated in Bitcoin-NG via a reporting mechanism that must be completed before the leader spends the ill-gotten funds. These two limitations are not present in our approach.

Bitcoin has not deployed Bitcoin-NG and instead has opted to keep blocks small and cap the rate of validated transactions per second; Bitcoin Cash has opted to deploy large 8MB blocks. We note that if Bitcoin-NG were adopted in the future, our results are complementary. The adoption of bobtail

mining into Bitcoin-NG would bound (with very high probability) the length of time that any miner is leader, therefore also bound the length of time that a faulty miner could block transaction validation.

**Blockchains without PoW.** Several newer blockchains are not based on computational proof of work, and our solution does not apply to them. These include proof-of-storage [24], proof-of-stake [7, 8], and blockless [10] schemes. However, almost all wealth stored in cryptocurrencies are in computational PoW blockchains that our approach does apply to, including Bitcoin, Bitcoin Cash, Litecoin, Ethereum, and Ethereum Classic.

### 3 LIMITING PROPERTIES OF ORDER STATISTICS

Our goal in this section is to derive the distribution of  $W_k$  (see Eq. 1); in Section 4, we use the distribution to derive the reduction in inter-block time variance that results from a given value of  $k$ .

$W_k$  is simply the sample mean over the lowest  $k$  order statistics  $V_1, \dots, V_k$ . But unfortunately, the analysis below is not straightforward because the  $V_i$  are neither independent nor identically distributed. We begin by deriving the individual and pair-wise joint distributions for the  $V_i$ , and ultimately use those to find  $W_k$ . In doing so, we establish that each order statistic is gamma distributed:  $V_i \sim \text{Gamma}(v; i, \beta)$ .

#### 3.1 Distribution of the $k$ th Order Statistic

Here we show that the  $k$ th order statistic is gamma distributed. To begin, note that parameter  $\theta$ , which is the number of hashes produced, can be expressed as a function of  $S$ :

$$\theta = \frac{S}{\beta}, \quad (3)$$

where  $\beta$  represents the expected minimum hash (i.e.,  $V_1$ ) that will arise from  $\theta$  hashes during interval  $I$ .

Next, we express a well-known result in our terms.

**LEMMA 1:** *The probability density function (pdf) of the  $i$ th order statistic,  $V_i$ , from  $\theta$  samples (i.e., hashes) is*

$$f_{V_i}(v; \theta) = \frac{\theta!}{(i-1)!(\theta-i)!} \frac{1}{S} \left(\frac{v}{S}\right)^{i-1} \left(1 - \frac{v}{S}\right)^{\theta-i}. \quad (4)$$

**PROOF:** The  $i$ th order statistic of a continuous random sample  $\mathbf{X} = X_1, \dots, X_n$  is given by<sup>2</sup>

$$f_{X_i}(x; n) = \frac{n!}{(i-1)!(n-i)!} f_X(x) [F_X(x)]^{i-1} [1 - F_X(x)]^{n-i}, \quad (5)$$

where  $F_X(x)$  is the CDF and  $f_X(x)$  is the PDF of the population from which samples are drawn. In our case, the number of samples is  $\theta$ , the CDF is  $\frac{v}{S}$ , and the PDF is  $\frac{1}{S}$ , and the result follows directly.  $\square$

Now consider how  $f_{V_i}(v | S)$  changes as  $S$  increases.

**THEOREM 1:** *The pdf of the  $i$ th order statistic,  $V_i$ , drawn from a population of samples parameterized by  $\beta$  is*

$$f_{V_i}(v; \beta) = g(v; i, \beta). \quad (6)$$

**PROOF:** Here we make use of Eq. 4 (Lemma 1) and Eq. 3:

$$\begin{aligned} f_{V_i}(v; \beta) &= \lim_{S \rightarrow \infty} f_{V_i}(v; S, \beta) \\ &= \lim_{S \rightarrow \infty} \frac{(S/\beta)!}{(i-1)!(\frac{S}{\beta}-i)!} \frac{1}{S} \left(\frac{v}{S}\right)^{i-1} \left(1 - \frac{v}{S}\right)^{\frac{S}{\beta}-i} \\ &= \lim_{S \rightarrow \infty} \frac{(S/\beta)!}{S^i (i-1)! (\frac{S}{\beta}-i)!} v^{i-1} \left(1 - \frac{v}{S}\right)^{\frac{S}{\beta}-i} \\ &= \frac{v^{i-1}}{(i-1)!} \left[ \lim_{S \rightarrow \infty} \frac{(\frac{S}{\beta})! \dots (\frac{S}{\beta}-i+1)!}{S^i} \right] \left[ \lim_{S \rightarrow \infty} \left(1 - \frac{v}{S}\right)^{\frac{S}{\beta}-i} \right] \\ &= \frac{v^{i-1}}{(i-1)! \beta^i} e^{-\frac{v}{\beta}} \\ &= g(v; i, \beta), \end{aligned} \quad (7)$$

where  $g(v; i, \beta)$  is the pdf of the  $\text{Gamma}(v; i, \beta)$  distribution. The second to last step follows from the fact that

$$\lim_{S \rightarrow \infty} \frac{(\frac{S}{\beta})! \dots (\frac{S}{\beta}-i+1)!}{S^i} = \lim_{S \rightarrow \infty} \frac{(\frac{S}{\beta})^i}{S^i} = \frac{1}{\beta^i}, \quad (8)$$

and the common limit

$$\lim_{S \rightarrow \infty} \left(1 - \frac{v}{S}\right)^{\frac{S}{\beta}} = e^{-\frac{v}{\beta}}, \quad (9)$$

which implies that

$$\lim_{S \rightarrow \infty} \left(1 - \frac{v}{S}\right)^{\frac{S}{\beta}-i} = \left[ \lim_{S \rightarrow \infty} \left(1 - \frac{v}{S}\right)^{-i} \right] \left[ \lim_{S \rightarrow \infty} \left(1 - \frac{v}{S}\right)^{\frac{S}{\beta}} \right] \quad (10)$$

$$= 1 \cdot e^{-\frac{v}{\beta}}. \quad (11)$$

$\square$

#### 3.2 Joint Distribution of Order Statistics

**THEOREM 2:** *The joint distribution of the  $i$ th and  $j$ th order statistic from a population parameterized by  $\beta$  is*

$$f_{V_i, V_j}(u, v; \beta) = g(u; i, \beta) g(v - u; j - i, \beta). \quad (12)$$

<sup>2</sup>See for example, Casella and Berger [11], Theorem 5.4.4.

**PROOF:** It is well known<sup>3</sup> that the joint distribution of the  $i$ th and  $j$ th order statistic is given by

$$f_{X_i, X_j}(x, y; n) = \frac{n!}{(i-1)!(j-1-i)!(n-j)!} f_X(x) f_X(y) [F_X(x)]^{i-1} \times [F_X(y) - F_X(x)]^{j-1-i} [1 - F_X(y)]^{n-j}. \quad (13)$$

Thus, we have

$$f_{V_i, V_j}(u, v; \theta) = \frac{\theta!}{(i-1)!(j-1-i)!(\theta-j)!} \frac{1}{S^2} \left(\frac{u}{S}\right)^{i-1} \times \left(\frac{v-u}{S}\right)^{j-1-i} \left(1 - \frac{v}{S}\right)^{\theta-j}, \quad (14)$$

and

$$\begin{aligned} f_{V_i, V_j}(u, v; S, \beta) &= \frac{u^{i-1}(v-u)^{j-1-i}}{S^j} \frac{\theta!}{(i-1)!(j-1-i)!(\theta-j)!} \left(1 - \frac{v}{S}\right)^{\theta-j} \\ &= \frac{\frac{S}{\beta} \dots \left(\frac{S}{\beta} - j + 1\right)}{S^j} \frac{u^{i-1}(v-u)^{j-1-i}}{(i-1)!(j-1-i)!} \left(1 - \frac{v}{S}\right)^{\frac{S}{\beta} - j}. \end{aligned} \quad (15)$$

Finally, assuming  $j > i$ , and reasoning in the same manner as in Section 3.1,

$$\begin{aligned} f_{V_i, V_j}(u, v; \beta) &= \lim_{S \rightarrow \infty} f_{V_i, V_j}(u, v; S, \beta) \\ &= \frac{1}{\beta^j} \frac{u^{i-1}(v-u)^{j-1-i}}{(i-1)!(j-1-i)!} e^{-\frac{v}{\beta}} \\ &= \frac{u^{i-1}}{\beta^i (i-1)!} e^{-\frac{u}{\beta}} \frac{(v-u)^{j-1-i}}{\beta^{j-i} (j-1-i)!} e^{-\frac{v-u}{\beta}} \\ &= g(u; i, \beta) g(v-u; j-i, \beta). \end{aligned} \quad (16)$$

□

## 4 PROPERTIES OF THE $K$ -OS CRITERION

We have established that  $V_i \sim \text{Gamma}(v; i, \beta)$ , and now wish to determine the mean and variance of  $W_k$ , the mining proof. Subsequently, we quantify how variance decreases as  $k$  increases.

### 4.1 Expectation and Variance of Bobtail Mining

**THEOREM 3:** The expectation of  $W_k$  is

$$E_\beta[W_k] = \frac{\beta(k+1)}{2}. \quad (17)$$

<sup>3</sup>See Casella and Berger [11], Theorem 5.4.6.

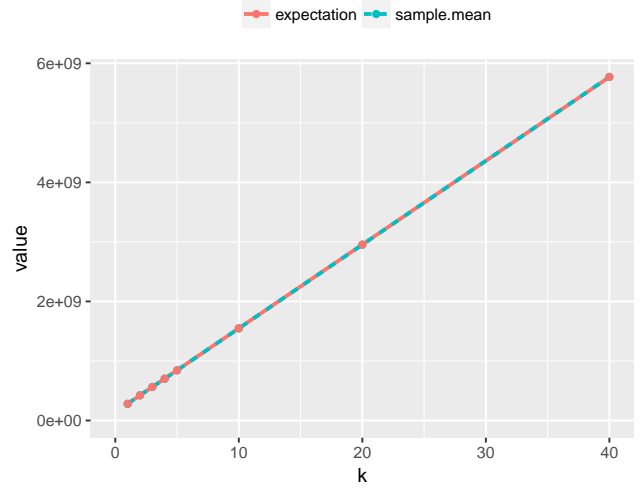


Figure 1: Eq. 17 versus simulation.

**PROOF:** Note first that since  $V_i \sim \text{Gamma}(v; i, \beta)$ , it follows that  $E_\beta[V_i] = i\beta$ . Taking the expectation of Eq. 1, we have

$$\begin{aligned} E_\beta[W_k] &= E_\beta \left[ \frac{1}{k} \sum_{i=1}^k V_i \right] \\ &= \frac{1}{k} \sum_{i=1}^k E_\beta[V_i] \\ &= \frac{1}{k} \sum_{i=1}^k i\beta \\ &= \frac{\beta(k+1)}{2}. \end{aligned} \quad (18)$$

□

**Empirical Validation of Theorem 3.** Figure 1 compares Eq. 17 versus a result obtained through a small Monte Carlo simulation of Bobtail mining run tens of thousands of times, with  $k$  as the independent variable. In all cases, the results match perfectly.

Now we prove our main result: an expression for the variance of the bobtail mining criterion, which decreases with  $k$ .

**THEOREM 4:** The variance of  $W_k$  is

$$\text{Var}_\beta[W_k] = \frac{(k+1)(2k+1)}{6k} \beta^2. \quad (19)$$

**PROOF:** Assuming that  $j > i$ , Equation 12 yields

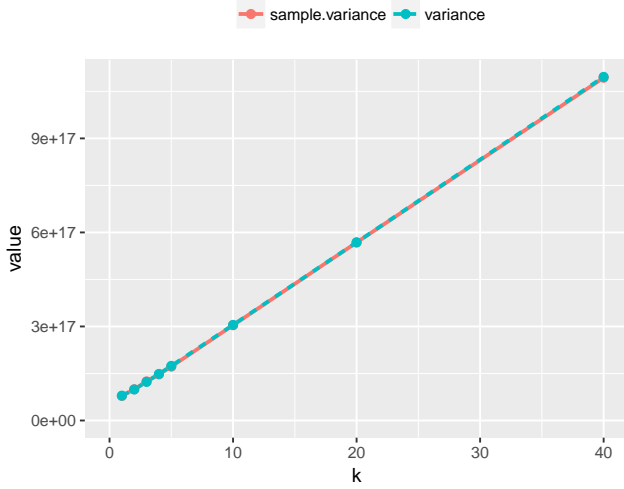


Figure 2: Eq. 19 versus simulation.

$$\begin{aligned}
E_\beta[V_i V_j] &= \int_0^\infty \int_u^\infty uv g(u; i, \beta) g(v - u; j - i, \beta) dv du \\
&= \int_0^\infty ug(u; i, \beta) \int_0^\infty (w + u) g(w; j - i, \beta) dw du \\
&= \int_0^\infty ug(u; i, \beta) [(j - i)\beta + u] du \\
&= \beta(j - i) \int_0^\infty ug(u; i, \beta) du + \int_0^\infty u^2 g(u; i, \beta) du \\
&= i\beta^2(j - i) + i\beta^2(1 + i) \\
&= i\beta^2(1 + j).
\end{aligned} \tag{20}$$

Before continuing, we note that since  $V_i \sim \text{Gamma}(v; i, \beta)$ , it follows that  $\text{Var}_\beta[V_i] = i\beta^2$ . Now, assuming that  $j > i$ , and using Eq. 20, it follows that

$$\begin{aligned}
\text{cov}_\beta[V_i, V_j] &= E_\beta[V_i V_j] - E_\beta[V_i] E_\beta[V_j] \\
&= i\beta^2(1 + j) - (i\beta)(j\beta) \\
&= i\beta^2 \\
&= \text{Var}_\beta[V_i].
\end{aligned} \tag{21}$$

Finally, we find the variance of  $W_k$  by substituting first Eq. 1 and then Eq. 21:

$$\begin{aligned}
\text{Var}_\beta[W_k] &= \text{Var}_\beta \left[ \frac{1}{k} \sum_{i=1}^k V_i \right] \\
&= \frac{1}{k^2} \text{Var}_\beta \left[ \sum_{i=1}^k V_i \right] \\
&= \frac{1}{k^2} \left( \sum_{i=1}^k \text{Var}_\beta[V_i] + 2 \sum_{j=1}^k \sum_{i=1}^{j-1} \text{cov}_\beta[V_i, V_j] \right) \\
&= \frac{1}{k^2} \left( \sum_{i=1}^k i\beta^2 + 2 \sum_{j=1}^k \sum_{i=1}^{j-1} i\beta^2 \right) \\
&= \frac{\beta^2}{k^2} \left( \frac{k(k+1)}{2} + \sum_{j=1}^k j(j-1) \right) \\
&= \frac{\beta^2}{k^2} \left( \frac{k(k+1)}{2} + \frac{k(k+1)(2k+1)}{6} - \frac{k(k+1)}{2} \right) \\
&= \frac{(k+1)(2k+1)}{6k} \beta^2.
\end{aligned} \tag{22}$$

□

**Empirical Validation of Theorem 4.** Figure 2 shows Eq. 19 versus our Monte Carlo simulation where  $k$  is the independent variable. The results show an exact match.

Figure 3 shows the distribution of  $W_k$ , the block discovery time. The top plot shows the probability distribution function (PDF) and the bottom plot shows the cumulative distribution function (CDF) based on the results of a Monte Carlo simulation. Each plot's x-axis is shown in terms of the minutes per block for Bitcoin. As the plots illustrate, the use of Bobtail mining results in a significant decrease in variance for discovering new blocks. Below, we characterize this reduction in variance due to the choice of  $k$  in a single equation.

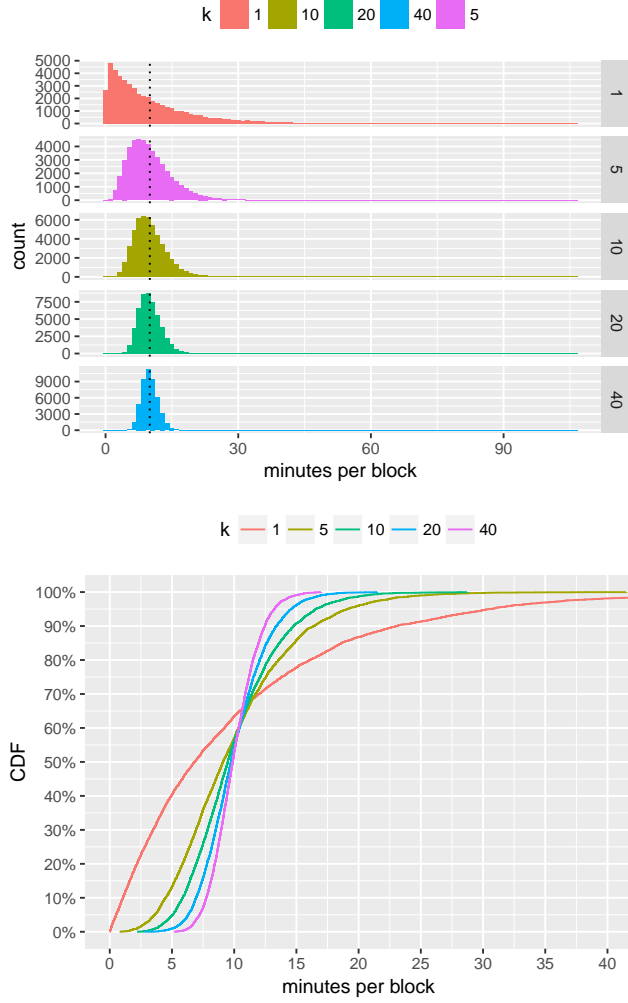
## 4.2 Reduction in Variance as $k$ increases

We can derive a formula for adjusting the target values of existing blockchains so that values of  $k > 1$  can be adopted without changing the mean inter-block time. This formula allows our scheme to be adopted as a patch, rather than deployed as a new system.

**Target adjustment.** Equation 3 relates the number of hashes performed,  $\theta$ , to the size of the hash space  $S$  and the minimum hash value achieved,  $\beta$ . Thus far we have interpreted  $\theta$  and  $\beta$  as fixed parameters, but it is equally valid to treat  $\theta$  as a random function of  $W_k$ , where  $\beta$  remains fixed, but is rewritten as  $\frac{k+1}{2E_\beta[W_k]}$  using Equation 17. Now noting that  $t_k = E_\beta[W_k]$  by definition, we can derive a functional expression for  $\theta_k$

$$\theta_k = \frac{S}{\beta} = \frac{S(k+1)}{2E_\beta[W_k]} = \frac{S(k+1)}{2t_k}. \tag{23}$$

Equation 23 relates the number of hashes performed,  $\theta_k$ , to the size of the hash space  $S$  and the target  $t_k$ . In this context,

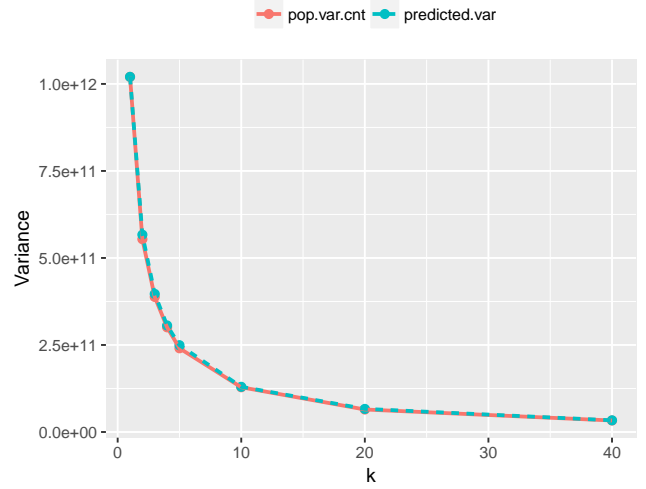


**Figure 3: The full PDF (top) of  $W_k$  for a different  $k$  in each facet, based on a Monte Carlo simulation. The same data is shown as a CDF (bottom). Each plot's x-axis is shown in terms of the minutes per block for Bitcoin.**

$\theta_k$  can be interpreted as the expected number of hashes performed network-wide under the  $k$ -OS criterion, given  $\beta$  and  $t_k$ . Thus in order to perform the same number of hashes in expectation under both the 1-OS and  $k$ -OS criteria, i.e.  $\theta_k = \theta_1$ , we should adjust the targets such that

$$t_k = \frac{t_1(k+1)}{2} \quad (24)$$

**Improvement in variance.** We next turn our attention to quantifying the improvement in mining time variance that is realized by using the  $k$ -OS criterion over the 1-OS criterion.



**Figure 4: Eq. 28 from Theorem 5 (in blue) versus simulation (in red).**

**THEOREM 5:** As  $k$  increases, the variance in block discovery time decreases by  $\frac{8k+4}{6(k^2+k)} = O(\frac{1}{k})$ .

**PROOF:** From the functional definition for the expected number of hashes provided in Equation 23, we can infer the following expression for  $\Theta_k$ , a random variable representing the number of hashes

$$\Theta_k = \frac{S(k+1)}{2W_k}, \quad (25)$$

where a fixed  $\beta$  is implicit. We seek to understand how  $\text{Var}_\beta[\Theta_k]$  changes with  $k$ ; in particular we would like to determine  $\text{Var}_\beta[\Theta_k]/\text{Var}_\beta[\Theta_1]$ , when  $E_\beta[\Theta_k] = E_\beta[\Theta_1]$  (i.e., expected hash rates are equal).

Because we have already developed a considerable set of tools for analyzing statistics of  $W_k$ , our goal will be to express  $\text{Var}_\beta[\Theta_k]$  in terms of  $W_k$ . For arbitrary random variable  $X$  having mean  $\gamma$ , and for any differentiable function  $g$ , it is well known<sup>4</sup> that  $\text{Var}_\gamma[g(X)] \approx [g'(\gamma)]^2 \text{Var}_\gamma[X]$ . In the context of our problem, this implies that

$$\text{Var}_\beta[\Theta_k] = \text{Var}_\beta \left[ \frac{S(k+1)}{2W_k} \right] \approx \left( \frac{S(k+1)}{2} \right)^2 \frac{\text{Var}_\beta[W_k]}{E_\beta[W_k]^4}. \quad (26)$$

<sup>4</sup>See Casella and Berger [11], Equation 5.5.9.

Now if  $E_\beta[\Theta_k] = E_\beta[\Theta_1]$ , then  $E_\beta[W_k] = \frac{k+1}{2}E_\beta[W_1]$  and Equation 26 can be used to show that

$$\begin{aligned} \frac{Var_\beta[\Theta_k]}{Var_\beta[\Theta_1]} &\approx \left(\frac{k+1}{2}\right)^2 \frac{Var_\beta[W_k]}{Var_\beta[W_1]} \frac{E_\beta[W_1]^4}{E_\beta[W_k]^4} \\ &= \left(\frac{2}{k+1}\right)^2 \frac{Var_\beta[W_k]}{Var_\beta[W_1]} \\ &= \left(\frac{2}{k+1}\right)^2 \frac{(k+1)(2k+1)}{6k} \\ &= \frac{4(2k+1)}{6k(k+1)}. \end{aligned} \quad (27)$$

Thus, when  $E_\beta[\Theta_k] = E_\beta[\Theta_1]$ ,

$$\frac{Var_\beta[\Theta_k]}{Var_\beta[\Theta_1]} \approx \frac{4(2k+1)}{6k(k+1)} = \frac{8k+4}{6(k^2+k)} = O\left(\frac{1}{k}\right). \quad (28)$$

□

Therefore, when the 1-OS and  $k$ -OS mining criteria are calibrated to the same expected block time ( $\theta_1 = \theta_k$ ), the variance in the block time for the  $k$ -OS criterion is less than that of the 1-OS criterion by a linearly decreasing factor.

**Empirical Validation of Theorem 5.** Figure 4 shows Eq. 28 versus our Monte Carlo simulation, showing an exact match.

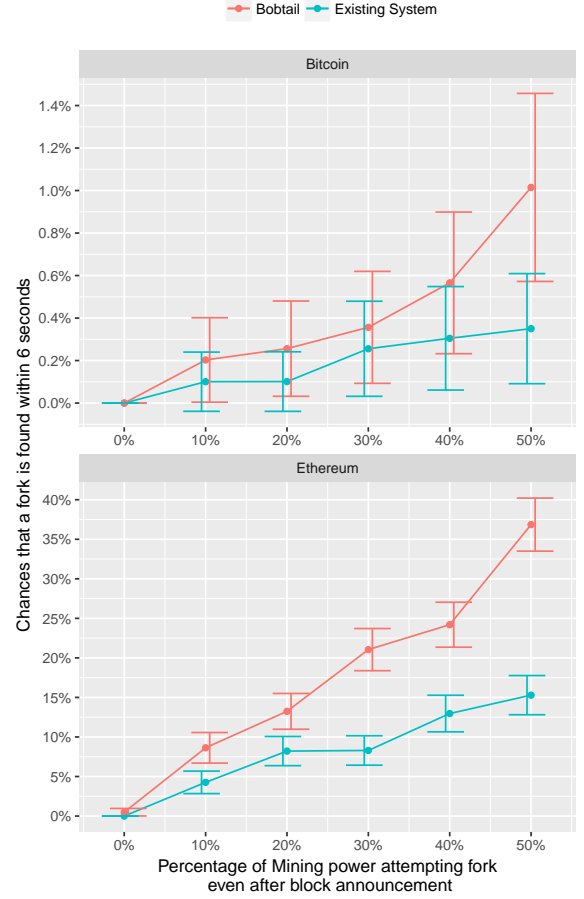
## 5 INCREASING CONSENSUS

Even when all miners operate honestly, all current blockchain systems suffer from forks during their operation that delay consensus. Even when miners are acting honestly, forks occur because the announcement of a new block take time to propagate to all other miners. A second miner may produce a valid block before the first miner's block reaches her. At that point, some fraction of the miners will attempt to build on the first block, and the complementary fraction will build on the second block. If the set of transactions in the two blocks is not the same, then consensus is delayed. While the chances of a fork in Bitcoin is relatively low, Ethereum's use of a 15 second average block discovery time increases the fork rate significantly.

In this section, we examine the rate at which forks occur in Bobtail mining compared to Bitcoin and Ethereum. We show that if all miners are acting honestly, forks are no more likely. However, if miners are purposefully trying to cause forks by continuing to mine after a block has been announced and received, Bobtail mining's forking rate is higher in the case of Ethereum's parameters, but about the same for Bitcoin's parameters.

### 5.1 Thwarting Forks

In this section, we introduce two restrictions on Bobtail miners that thwart forks. First, we require that all proofs are tethered to a *supporting proof*, which should be the smallest proof previously heard by the miner for the same prior block (which may or may not be smaller than the primary proof).



**Figure 5: The results from a discrete event simulation of existing systems and Bobtail mining. In the case of a Bitcoin-like configuration, the forking rate is not higher (statistically speaking) until 50% of the mining power is dishonest. In the case of an Ethereum-like configuration, the forking rate is higher only if dishonest miners are present; because Ethereum uses GHOST, the damage to consensus is limited. (Points are means, error bars are 95% c.i.s.)**

The use of supporting proofs thwarts proof reuse in new block because we require that the proofs included in a block must have a support proofs that are greater than or equal to the block's first order statistic.

As a second restriction, we require that miners delay their creation of a new block if they know of a first order statistic that they did not create. Proof announcements are small compared to a full block and will propagate quickly; they are placeholders for blocks that are to come and reduce forks. In practice, a miner can acquire a reputation for following up with a block after a proof is announced; any miner that has not earned a good reputation could be ignored.

## 5.2 Performance

These mechanisms can be ignored by dishonest miners: they might list the second order statistic as their supporting proof. Or they may elect to reuse proofs in new blocks; similarly, miners in existing blockchains may simply keep mining after a block announcement to try and fork the blockchain.

To quantify the affect that dishonest miners have the network, we ran a discrete event simulator. The simulation had 10 miners with equal mining power, with between 0 and 5 of them acting dishonestly. Miners were connected in a simple star topology. Proofs were received by other miners 1 second after they were mined, and blocks were received 3 seconds after they were mined. Honest miners kept mining until they mined a block or received a block from another miner. Dishonest miners kept mining for an additional 3 seconds after the block was received (i.e., 6 seconds total since the first block was found), unless they were in fact the miner of the first block. We ran one Bitcoin-like configuration where blocks appeared on average once every 10 minutes, and a separate set of simulations in an Ethereum-like network where blocks appeared every 15 seconds. (All times were simulated and mining occurred by sampling random numbers from a uniform distribution.) We compare

Figure 5 shows the results for both configurations. In the case of the Bitcoin, the simulation results show that while Bobtail mining has slightly larger rate of forks, it is not statistically significant until 50% of the miners are dishonest. For the Ethereum configuration, Bobtail mining is more prone to forking from dishonest miners. On the other hand, Ethereum embraces forks by implementing the GHOST [] algorithm, and unless they are coupled with some other attack, these forks cause no damage.

In both cases, when miners act honestly, the forking rate is no larger than the existing system.

## 6 HONEST BEHAVIOR IS MORE PROFITABLE

In contrast to the standard Bitcoin protocol, where only the miner who successfully mines a block is required to publish her work, the  $k$ -OS criterion requires  $k$  proofs from multiple miners in order to mine a block. Therefore, we must incentivize all miners producing these proofs to ensure that they participate. Moreover, given our consensus mechanism described in Section 5, we would also like to incentivize miners to include the lowest known LOS at the time that they produce their proof. Note that for any fixed incentive provided to any of the  $k$  lowest OSes, the natural tendency for miners would be to include the highest known proof as LOS so that they have the greatest possible chance of being include in some mining package. Thus it is clear from this discussion that each of the  $k$  proofs should receive *some* reward, but that

k	Proof	L.O.S.	Reward
1	358325	—	4.55439431
2	1217458	358325	0.23809524
3	1721868	358325	0.11309524
4	1777139	358325	0.05357143
5	1995396	358325	0.02380952
6	3621245	358325	0.01041667
7	4582015	358325	0.00372024
8	4781376	358325	0.00148810
9	7277279	358325	0.00046503
10	3761724	1826037	0.00055804
11	4420661	1826037	0.00025577
12	6302668	1826037	0.00008138
13	7514262	1826037	0.00002325
14	7601030	1826037	0.00000872
15	1826037	3521660	0.00001017
16	4707122	3521660	0.00000436
17	3521660	3927808	0.00000182
18	7881560	3927808	0.00000036
19	3927808	6374495	0.00000027
20	6374495	9175814	0.00000009

Figure 6: Rewards payout for an example block where  $k = 20$ .

reward should be structured so as to encourage the proofs to include the lowest possible LOS at the time of their creation.

Let  $B$  be the total block reward and consider the following incentive structure. Each block lists  $k$  proofs:  $P_1, \dots, P_k$ . We create an ordering of the proofs based on the LOS each references, and we break ties by using each proof's own values (i.e., the value of their order statistic  $V_i$ ). For proof  $P_i$ , function  $\pi(P_i)$  returns the position for  $P_i$  in the sort order. Note that, by construction, it is always the case that  $\pi(P_1) = 1$ .

As an example, suppose proofs  $P_{10}$ ,  $P_{11}$ , and  $P_{12}$  (corresponding to order statistics  $V_{10}$ ,  $V_{10}$ , and  $V_{12}$ ) have respectively chosen LOSes  $V_1$ ,  $V_2$ , and  $V_1$ . Then the ordering is  $\pi(V_{10}) < \pi(V_{12}) < \pi(V_{11})$ .

We provide the miner of a proof  $P_i$  (for  $i < 1$ ) according to this reward function:

$$R_i = 2B(1 - \epsilon)^i \frac{k + 1 - \max(i, \pi(P_i))}{k(k + 1)}. \quad (29)$$

where  $0 < \epsilon < 1$ .

Figure 6 shows an example block with all rewards computed.

We next argue that this reward structure provides the desired incentives outlined above.

To begin, note that the miner producing proof  $P_i$  is incentivized to include the lowest LOS possible in his proof because that will tend to improve his rank in the sort order, reducing  $\pi(P_i)$ , and increasing  $R_i$  overall. Note also that if all proofs point to the same LOS (that LOS being  $P_1$ ), then the reward reduces to

$$R_i = 2B(1 - \epsilon)^i \frac{k + 1 - i}{k(k + 1)}, \quad (30)$$



and,

$$\begin{aligned}\sum_{i=1}^k R_i &= \frac{2B}{k(k+1)} \sum_{i=1}^k (1-\epsilon)^i [k+1-i] \\ &\leq \frac{2B}{k(k+1)} \left( k(k+1) - \frac{k(k+1)}{2} \right) \\ &= B.\end{aligned}\quad (31)$$

Therefore, the maximum possible cumulative reward is bounded by  $B$ .

The last thing that we must check is that each miner is incentivized to hash as much as possible (i.e. always aims for achieving  $V_1$  instead of  $V_i$ ,  $i > 1$ ). We assume that the miners have achieved an equilibrium where the target  $t_k$  (and therefore  $\beta$ ) remains stable. Let  $\theta(P_i)$  be the expected number of hashes devoted to producing proof  $P_i$ . Using Theorem 1 and Equation 23, we can infer that

$$\theta(P_i) = \frac{S}{E_{i\beta}[W_1]} = \frac{S}{i\beta}. \quad (32)$$

Thus, the expected reward per hash for proof  $P_i$ ,  $E[R_i|\theta(P_i)]$ , is given by

$$E[R_i|\theta(P_i)] = \frac{R_i}{\theta(P_i)} \quad (33)$$

$$= 2B(1-\epsilon)^i \frac{k+1-i}{k(k+1)} \frac{i\beta}{S} \quad (34)$$

We next show that there exists a choice for  $\epsilon$  such that  $P_1$  offers a greater reward per unit hash than any other  $P_i$ . Let  $\epsilon = \frac{1}{2}$  and  $i > 2$ , then

$$\frac{E[R_1|\theta(P_1)]}{E[R_i|\theta(P_i)]} = \frac{k}{i(k+1-i)(1-\epsilon)^{i-1}} \quad (35)$$

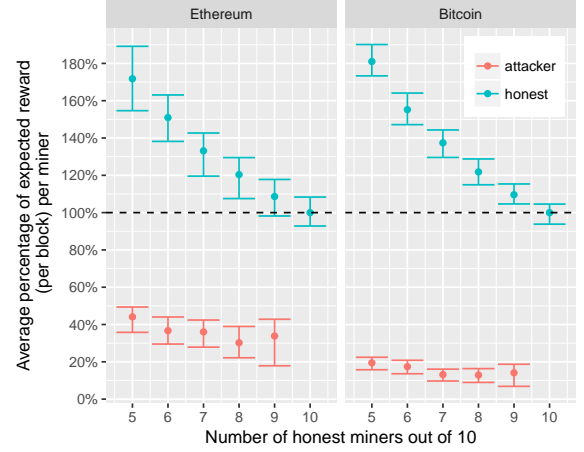
$$= \frac{k}{i(k+1-i) \left(\frac{1}{2}\right)^{i-1}} \quad (36)$$

$$> \frac{k}{ki \left(\frac{1}{2}\right)^{i-1}} \quad (37)$$

$$> 1. \quad (38)$$

The mean reward for honest nodes is higher than attackers and is statistically significant except when there is 1 attacker out of 10, in which case the attacker earns a reward with a lower average but is not statistically different. Notably, when all miners are attackers, there is no difference in reward compared to when all are honest.

Figure 7 shows the rewards given to miners following honest or dishonest strategies, using the same simulation as in Section 5. In all cases, dishonest miners suffer fewer rewards from not using the first order statistic so that they may mine their own block.



**Figure 7: The average reward for miners when following honest or attacker strategies. The dotted line is the expected reward given mining power. Error bars show 95% confidence intervals. In all cases, dishonest miners suffer fewer rewards from trying to mine their own block.**

## 7 HEADER FIELD LOGISTICS AND OVERHEAD

In this section, we detail changes that are required of blockchain systems to adopt our low variance block discovery algorithm. Primarily, changes are required for (i) block headers and (ii) block announcements.

### 7.1 Block Headers

In deployed blockchain systems, including Bitcoin and Ethereum, a large number of bytes are included as input to the PoW algorithm. For example, in Bitcoin the 80-byte header is hashed and compared against the target. Ethereum uses the ETHASH algorithm and is larger still. A naive strategy for deploying our system would be to concatenate  $k$  block headers in one package; we propose instead the following much more efficient approach.

Our deployment strategy works with any POW blockchain system. Let  $\{p_1, p_2, \dots\}$  be the set of values in a block header that act as input to a proof of work algorithm in the original blockchain algorithm. For example, in Bitcoin those values are a version number, the prior block's hash, the merkle root hash, the time, the target, and a nonce. In existing systems, a block is valid if  $H(p_1, p_2, \dots) \leq t$ .

We adjust the PoW algorithm as follows. For a miner  $i$ , let  $p_{i1}, p_{i2}, \dots$  be the set of values normally in a block header excluding the hash of the prior block, and let  $P_i = H(p_{i1}, p_{i2}, \dots)$ . Let  $A_i$  correspond to the coinbase address of the miner of  $P_i$ . Let  $Y$  be the hash of the prior block. Let  $L_i$  be the hash of current first order statistic announced to date (for the same prior block), a requirement explained in

Ver.	Time	Target	Prior	Merkle hash	Nonce
P2, L2		...			Pk, Lk
coinbase (with outputs A1, ..., Ak)					

**Figure 8: A header that includes values needed by Bobtail.**

Section 5. Let  $Q_i = H(P_i, A_i, Y, L_i)$ . A miner announces all four values together if  $Q_i$  is less than one of the  $k$  lowest  $Q$  values announced to date. (Below, in Theorem 6 we prove that only those values  $Q_i < kt$  need be announced.)

A new block is discovered if the following two requirements hold:

$$\frac{1}{k} \sum_{i=1}^k Q_i = \left( \frac{1}{k} \sum_{i=1}^k H(P_i, A_i, Y, L_i) \right) \leq t_k, \quad (39)$$

and

$$H(P_i, A_i, Y, 0) \leq L_i, \text{ for all } i, \quad (40)$$

where the comma represents concatenation. The requirement for all  $L_i$  is designed to keep the orphan rate low, as described in Section 5. Because a miner's address is included with the hashed value, the associated coinbase reward cannot be reassigned by a third party. Similarly, because the hash of the prior block is included, work cannot be reused in later blocks.

**Block header size.** Fig. 8 shows the new header if applied to Bitcoin as an example. Block headers in our system include each value  $P_2, \dots, P_k$ . The value of  $P_1$  can be determined from the other header fields. In this case,  $P_1 = H(\text{Time}, \text{Difficulty}, \text{Prior}, \text{Merkle Root}, \text{Nonce})$ . The coinbase transaction is a part of the header because the addresses  $A_1, \dots, A_k$  are required to validate the block. The merkle root hash does not include the coinbase transaction.

We shorted each  $L_i$  values to (i) a byte representing the number of leading zeros, followed by (ii) the next most significant 2 bytes of the hash. These three bytes are sufficient to validate the block. When a value  $P_i$  is announced and there is not yet a lower  $Q$  value, then all zeros are used for  $L_i$ .

Bitcoin's header is 80 bytes and has a 205 byte coinbase transaction, typically. Assuming SHA-256, our header will grow as follows. Each  $P_i$  value is 32 bytes. Each  $L_i$  value is 3 bytes. And the coinbase transaction will grow from a single output to  $k$  outputs. For example, in Bitcoin, a coinbase script with  $k$  outputs has very roughly  $170 + 35k$  bytes. Hence the header in our system will grow to  $80 + 35(k-1) + 170 + 35k = 215 + 70k$  bytes.

For example, when  $k = 40$ , block headers would contain 3,015 bytes of header and coinbase, compared to 285 bytes now. This is quite an increase but still small considering Bitcoin's 1MB blocks and Bitcoin Cash's 8MB blocks, or about 0.3% and 0.04%, respectively.

Ethereum's header is 508 bytes. Hence, bobtail mining would increase by  $35k$ , plus the cost of a transaction with  $k$  outputs. For  $k = 40$ , headers would grow to  $508 + 35(39) = 1873$  bytes plus the cost of a 40-output transaction. Blocks are about 22KB, so this increase is more than  $1873 / (22 \cdot 1024) = 8\%$  not including the transaction cost.

## 7.2 Network Overhead

When the mining criterion is  $k$ -OS, it is not efficient for each miner to send proof of work every time she finds a hash value lower than her previous best. A slight improvement to that scheme is for her to send proof of work only when her hash value is better than the lowest  $k$  hashes produced by all miners cumulatively. But even this approach will result in a large amount of network traffic early in the mining process because, initially, most hash values will be significantly higher than the target  $t_k$  (see Eq. 24).

To improve network efficiency significantly we require that miners do not send proof of work unless their achieved hash value falls below  $kt_k$  as the following simple theorem shows.

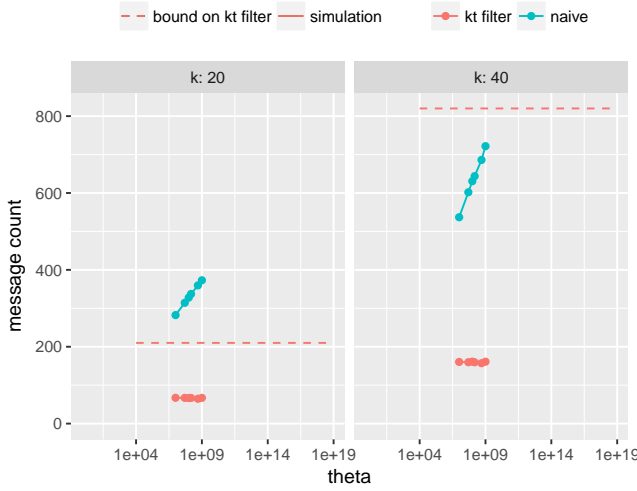
**THEOREM 6:** *The largest order statistic  $V_k$  of any valid block is bound by  $V_k \leq kt_k$ .*

**PROOF:** Even if the first  $k-1$  order statistics  $V_1, \dots, V_{k-1}$  are all zero, a valid proof against a target would require that  $V_k \leq kt_k$  to ensure that  $W_k \leq t_k$ . Note that this bound holds absolutely, regardless of the actual value of order statistics  $V_i$  for all  $i < k$ .  $\square$

**Bound on the number of messages.** Currently, bitcoin blocks are the result of about  $\theta = 8e18$  hashes, and so we do not possess the resources to simulate the process of real block creation. To evaluate our approach, we simulate block creation for significantly small theta, and we derive a bound on the number of hashes below Eq. 6 produced by miners; the number of network messages will be strictly smaller.

**THEOREM 7:** *The number of hashes per block produced by the network that fall below the threshold  $kt_k$  assuming a  $k$ -OS mining criterion and threshold  $t_k$  is*

$$E[M_{k,t_k}] = \frac{k(k+1)}{2}. \quad (41)$$



**Figure 9: The number of messages sent as a block is mined. The naive approach (in blue) sends every mining proof that is smaller than the lowest  $k$  sent. Our proposed approach has dramatically fewer messages (in red): mining proofs are sent only if they are lower than  $k * t$ . The dashed red line shows the upper bound for this method from Equation 41; notably it does not depend on  $\theta$  or difficulty. Bitcoin is currently parameterized to about  $\theta = 8e18$ .**

**PROOF:** We now seek to estimate  $M_{k,t_k}$ , the number of hashes per block produced by the network that fall below the threshold  $kt_k$  assuming a  $k$ -OS mining criterion and threshold  $t_k$ . For each block, miners generate a series of hashes  $Z_1, \dots, Z_\theta$ , with

$$\theta = \frac{S}{\beta} = \frac{S}{\frac{2t_k}{k+1}} = \frac{S(k+1)}{2t_k}. \quad (42)$$

Because the mining process generates a hash value uniformly at random from the interval  $[0, S]$ , we know that  $P[Z_i < kt_k] = \frac{kt_k}{S}$  for every  $Z_i$ . It follows that

$$E[M_{k,t_k}] = \sum_{i=1}^{\theta} P[Z_i < kt_k] = \frac{\theta kt_k}{S} = \frac{k(k+1)}{2}. \quad (43)$$

□

$E[M_{k,t_k}]$  is an upper bound on the number of hashes actually propagated through the network. This is because miners will also avoid sending a hash if it is not one of the  $k$  lowest order statistics observed thus far, even if it falls below the threshold  $kt_k$ .

**Empirical validation.** Fig. 9 shows the number of messages sent for various  $\theta$  from our Monte Carlo simulation. As the figure illustrates, all simulations were well below the bound.

## 8 CONCLUSION

We have designed and characterized a novel method of low-variance blockchain mining. We have derived expressions for the expectation and variance of the Bobtail mining proof and the number of hashes performed for any value of  $k$ . Using these expressions, we have shown that Bobtail reduces variance by a factor of  $O(1/k)$ , compared to using  $k = 1$ . We have shown that forks are created by Bobtail miners no more often than existing systems, and that dishonest miners receive significantly lower rewards due to a split rewards formula we’ve designed. Finally, we have quantified header overhead and network traffic and shown them as low-cost tradeoffs for reducing mining time variance.

## REFERENCES

- [1] Bitcoin cash. <https://www.bitcoincash.org/>.
- [2] Litecoin. <https://litecoin.org/>.
- [3] Ethash. <https://github.com/ethereum/wiki/wiki/Ethash>, Aug 3 2017.
- [4] Martin Abadi, Mike Burrows, Mark Manasse, and Ted Wobber. Moderately hard, memory-bound functions. *ACM Trans. Internet Technol.*, 5(2):299–327, May 2005.
- [5] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. Dos-resistant authentication with client puzzles. In *Revised Papers from the 8th International Workshop on Security Protocols*, pages 170–177, 2001.
- [6] Adam Back. Hashcash - Amortizable Publicly Auditable Cost-Functions, 2002.
- [7] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In *International Conference on Financial Cryptography and Data Security*, pages 142–157. Springer, 2016.
- [8] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of Activity: Extending Bitcoin’s Proof of Work via Proof of Stake [Extended Abstract] y. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.
- [9] Bobtails. [https://en.wikipedia.org/wiki/Natural\\_bobtail](https://en.wikipedia.org/wiki/Natural_bobtail).
- [10] Xavier Boyen, Christopher Carr, and Thomas Haines. Blockchain-Free Cryptocurrencies: A Framework for Truly Decentralised Fast Transactions. Cryptology ePrint Archive, Report 2016/871, Sept 2016. <http://eprint.iacr.org/2016/871>.
- [11] George Casella and Roger L. Berger. *Statistical inference*. Brooks Cole, Pacific Grove, CA, 2002.
- [12] Liqun Chen and Wenbo Mao. An auditable metering scheme for web advertisement applications. *Information Security*, pages 475–485, 2001.
- [13] F. Coelho. An (Almost) Constant-Effort Solution- Verification Proof-of-Work Protocol Based on Merkle Trees. In *Progress in Cryptology – AFRICACRYPT*, pages 80–93, June 2008.
- [14] Drew Dean and Adam Stubblefield. Using client puzzles to protect tls. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM’01, Berkeley, CA, USA, 2001. USENIX Association.
- [15] J. Douceur. The Sybil Attack. In *Proc. Intl Wkshp on Peer-to-Peer Systems (IPTPS)*, March 2002.
- [16] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *In 12th Annual International Cryptology Conference*, pages 139–147, 1992.
- [17] Ethereum Homestead Documentation. <http://ethdocs.org/en/latest/>.
- [18] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59, Santa Clara, CA, 2016. USENIX Association.

- [19] M. Franklin and D. Malkhi. Auditable metering with lighthweight security. In *Proc. Financial Cryptography*, pages 151–160, 1997.
- [20] Bogdan Groza and Bogdan Warinschi. Cryptographic puzzles and dos resilience, revisited. *Des. Codes Cryptography*, 73(1):177–207, October 2014.
- [21] Markus Jakobsson and Ari Juels. Proofs of Work and Bread Pudding Protocols. In *Proc. Conference on Secure Information Networks: Communications and Multimedia Security*, pages 258–272, 1999.
- [22] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proc. Networks and Distributed Security Systems*, pages 151–165, 1999.
- [23] Ben Laurie and Richard Clayton. “Proof-of-work” proves not to work; version 0.2. In *Proc. Workshop on Economics and Information Security*, 2004.
- [24] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. Permacoin: Repurposing bitcoin work for data preservation. In *Proc. IEEE Security and Privacy*, pages 475–490, 2014.
- [25] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, May 2009.
- [26] XiaoFeng Wang and Michael K. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP ’03, pages 78–, Washington, DC, USA, 2003. IEEE Computer Society.