Bobtail: Improved Blockchain Security with Low-Variance Mining

George Bissias
University of Massachusetts Amherst
gbiss@cs.umass.edu

Brian N. Levine
University of Massachusetts Amherst
brian@cs.umass.edu

***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. An unfortunate limitation of all deployed PoW blockchain systems is that the time between blocks has high variance. For example, Bitcoin produces, on average, one block every 10 minutes. However, 5% of the time, Bitcoin's inter-block time is at least 30 minutes.**

In this paper, we show that high variance is at the root of fundamental attacks on PoW blockchains. 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, which uses a single PoW sample, to one that incorporates k samples. We show that the variance of inter-block times decreases as k increases. Bobtail significantly thwarts doublespend and selfish mining attacks. For example, for Bitcoin and Ethereum, a doublespending attacker with 35% of the mining power will succeed with 44% probability when the merchant sets up an embargo of 1 block; however, when $k \geq 40$, the probability of success for the same attacker falls to less than 1%. Similarly, for Bitcoin and Ethereum currently, a selfish miner with 45% of the mining power will claim about 71% of blocks; however, when $k \geq 20$, the same miner will find that selfish mining is less successful than honest mining. We also investigate attacks newly made possible by Bobtail and show how they can be defeated. The primary costs of our approach are larger blocks and increased network traffic.

I. Introduction

Blockchain systems are designed to produce blocks of validated transactions at a constant average rate. The most popular systems employ a *Proof of Work (PoW)* algorithm as a means of creating these blocks [35], including Bitcoin [8], Bitcoin Cash [9], Ethereum [21], and Litecoin [33]. Whether a pure PoW or hybrid approach [11, 16, 31], the two most fundamental attacks on PoW blockchains are *selfish mining* [23] and the *doublespend* [35].

A direct consequence of using a PoW algorithm is that the time between blocks has high variance and the distribution of inter-block times has a very long tail. As we show in this paper, high variance is responsible for enabling doublespend and selfish mining attacks. Generally, miners in all deployed systems craft blocks by repeatedly changing a nonce in the block header until the cryptographic hash of that header is less than a target value t, $0 < t < S$, where S is the largest hash value (typically $S = 2^{256} - 1$). In other words, the hash of each header is a sample from $[0, S]$ taken randomly from a discrete uniform distribution. A block is discovered when the *first order statistic* (i.e., the minimum value) of all sampled values is less than target t. The resulting inter-block times are exponentially distributed (a result we show formally), and have a *variance equal to the square of the mean*.

Intuitively, variance explains why a miner with a small fraction of the system's total hash rate can successfully doublespend. Whenever the honest miners are unlucky and encounter high inter-block times, it's an opportunity for the attacker to conversely be lucky, mine with relatively smaller inter-block times, and produce a longer chain. In sum, if PoW mining could be achieved with a *lower inter-block time variance*, then doublespend and selfish mining attacks would fail more often.

In this paper, we propose an improved process for PoW-based block discovery that results in an inter-block time with significantly lower variance. As a result, our approach is significantly more secure against doublespend and selfish mining attacks. Our algorithm generalizes the current algorithm, which uses a single PoW sample, to one that incorporates k samples. We show that the variance of inter-block times decreases as k increases. For example, if our approach were applied to Bitcoin, nearly every block would be found within 5 to 18 minutes; and the average inter-block time would remain at 10 minutes. In comparison, currently 5% of Bitcoin's blocks have inter-block times of at least 30 minutes. We call our approach *Bobtail* mining.

As a result, in Bobtail, doublespend and selfish mining attack efficacies are drastically reduced. For example, in either Bitcoin or Ethereum, a doublespending attacker with 35% of the mining power will succeed with 44% probability when the merchant sets up an embargo of 1 block; however, for Bobtail, the probability of success for the same attacker falls to less than 1% (when $k \geq 40$). Similarly, for Bitcoin and Ethereum currently, a selfish miner with 45% of the mining power will claim about 71% of blocks; however, when $k \geq 20$, the same miner will find that selfish mining is defeated, resulting in a smaller fraction of blocks than honest mining.

One disadvantage of Bobtail is that new *withholding* and *denial-of-service* attacks are possible. However, we show how these attacks can be thwarted with careful protocol design.

Network and Distributed Systems Security (NDSS) Symposium 2020
23–26 February 2020, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2020.23095
www.ndss-symposium.org

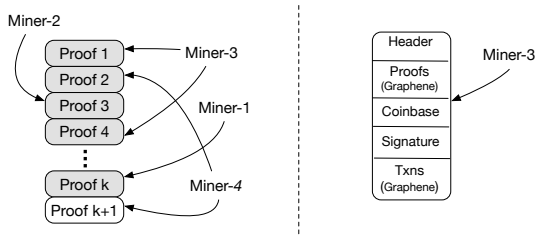

Fig. 1: (Left) In Bobtail, miners each broadcast *proof sets* whose hash values are called *proofs*. When the average of k proofs is at or below the target, a new block is found. (Right) The miner of the lowest proof selects the transactions that are included in the new block and announces that block to the network. Coinbase is rewarded to miners proportional to the number of proofs they contributed. Proofs (and transactions) in the block can be efficiently announced using Graphene [38].

Contributions. Our contributions are summarized as follows.

- We derive the statistical characteristics of our approach and validate each empirically. For example, we derive expressions for the expectation and variance of Bobtail block times and the amount of work performed for any value of k. Using these expressions, we quantify the reduction in variance of inter-block time for values of $k > 1$. We show that variance in block discovery time is reduced by $O(1/k)$ relative to the status quo.
- We demonstrate that low-variance mining significantly mitigates the threats to security posed by doublespend and selfish mining attacks, as stated above.
- We demonstrate that, due to our protocol mechanisms, Bobtail mining has the same or lower *orphaned block* rate as existing schemes.
- We quantify the increased network overhead of Bobtail, and demonstrate how these costs can be kept minimal by leveraging the statistical characteristics that we derive.
- We show that new intra-block DoS and withholding attacks are possible for Bobtail. However, through careful protocol design, these attacks are mitigated.

Bobtail is also a convenience for consumers, who would benefit from reduced block time variance and a more consistent flow of validated transactions through the system.

We close by presenting related work and offering concluding remarks. We begin by defining our problem statement and the Bobtail mining process and network protocol.

II. An Explanation of the Bobtail Protocol

In this section, we provide a high-level overview of the Bobtail protocol and its features.

Summary. In current PoW systems, miners iterate on a nonce within a block header until the hash of the header is equal to or lower than a known target t. The new block is announced to all other miners, who reference this hash value within the header of the next block that they will mine. This process creates a *blockchain* in that each subsequent block points to the previous. Similarly, in Bobtail, miners iterate a nonce in a block header seeking a value near t_k, a threshold that varies with security parameter k. As illustrated in Fig. 1(Left), once a suitable header is found, a miner announces it to all other miners; for clarity we call the announcement a *proof set* and its hash value simply a *proof*. If the mean of the k-lowest proofs is equal to or lower than the target t_k, then a new block is found. The miner with the lowest of all proofs is the only one that can announce the new block and is the one that controls the list of transactions confirmed by the block, as shown in Fig. 1(Right). To be clear, Bobtail blocks are not the result of k proofs from one miner, but the k-lowest from all miners. The coinbase reward is split evenly among all contributors (proportional to number proofs contributed), which leads to a smoother short-term distribution of rewards among lower hash rate miners than does the status quo. We now define this process more formally.

A. Problem Statement

Imagine that the mining process is carried out for exactly h hashes during time interval I, generating hash values $H_1, \ldots, H_h$ from $[0, S]$ uniformly at random. Now define V_i to be the value of the ith order statistic at the end of I, i.e. $V_i = H_{(i)}$ in standard notation. Let W_k be a random variable representing the average value over the k-lowest order statistics after h hashes.

$$W_k = \frac{1}{k} \sum_{i=1}^{k} V_i. \tag{1}$$

W_k constitutes the collective mining proof for the entire network. Our Bobtail mining criterion says that a new block is discovered when a realized value of W_k falls at or below the target t_k:

$$w_k \leq t_k. \tag{2}$$

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

The primary goals of our paper are therefore to show the following, given values of $k > 1$.

- Inter-block time variance is significantly reduced (Section IV).
- Selfish mining and doublespend attacks are significantly more difficult to carry out as k increases (Section V).
- Costs are relatively small. Orphan rates are no worse than for $k = 1$ (Section VI), and the increases in block size and network traffic are small and manageable (Section VII).
- New attacks made possible by setting $k > 1$ are easily mitigated (Section VIII).

B. Protocol Details

So that our results are presented in a concrete context, we begin by stating the details of the Bobtail network protocol.

Blocks. Bobtail blocks, illustrated in Fig. 2, consist of several components.

- A *block header* $\mathcal{H}$ contains the same fields as a conventional block header (e.g., in Bitcoin or Ethereum) and one additional field, described below.
- Proof package $\mathcal{K}$ is a collection of k *proof sets*. Each proof set $\mathcal{P}_i$ contains a payout address for the miner,

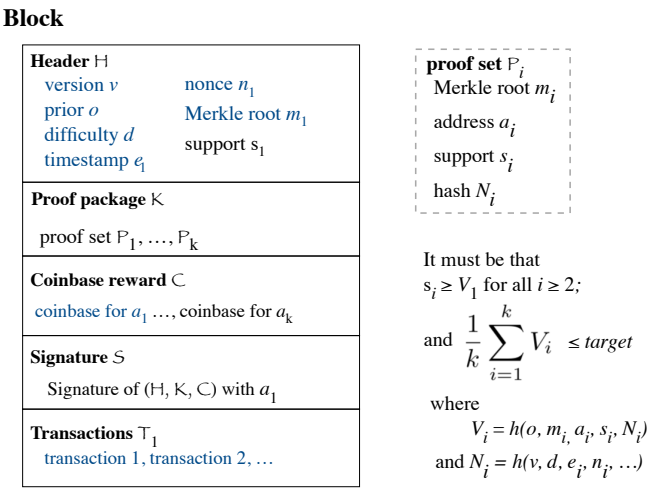

Fig. 2: Bobtail blocks are a superset of existing PoW schemes; items already present in a typical PoW blockchain appear in blue. Bobtail blocks are additionally signed by the miner of V_1 and add *proof sets* contributed by other miners.

values necessary for creating valid proof of work, and other values used for thwarting attacks. Proof sets are hashed to create PoW, i.e. proof $V_i = h(\mathcal{P}_i)$. The sets are ranked so that V_1 is defined as the smallest value or *first order statistic*.

- A set $\mathcal{C}$ of *coinbase transactions* awarded to the miners of the k proof sets.
- Cryptographic signature $\mathcal{S}$ of the set $(\mathcal{H}, \mathcal{K}, \mathcal{C})$, which must be generated with the private key that matches the payout address in $\mathcal{P}_1$. This thwarts an attack detailed in Section VIII.
- The body of the block is a set $\mathcal{T}_1$ of valid previously unconfirmed *transactions*. The subscript denotes that the miner of V_1 selects the transactions.

Mining. Bobtail mining is a generalization of the procedure implemented in conventional PoW blockchains. Each miner seeks to receive coinbase reward by generating one of the k proof sets $\mathcal{P}_1, \ldots, \mathcal{P}_k$ included in $\mathcal{K}$. Each proof set $\mathcal{P}_i$ contains the following fields.

- o is the hash of the signature $\mathcal{S}$ of the prior block.
- m_i is the root of a Merkle tree containing transactions $\mathcal{T}_i$.
- a_i is the miner's coinbase payout address.
- s_i is a *supporting proof* or *support*, which we define as the smallest proof value among all proof sets (pointing to the same prior block) that the miner has seen to date. This value helps prevent orphan blocks and withholding attacks.
- N_i is the hash of set $\mathcal{N}_i = (v, d, e_i, n_i, \ldots)$ containing the protocol version v, current difficulty d, timestamp e_i, and nonce n_i. Note that v and d are taken from $\mathcal{H}$. We allow zero or more optional arguments in $\mathcal{N}_i$ that may be required by the blockchain Bobtail is applied to.

To be eligible for inclusion in the same block, proof sets must use the same prior block o, version v, and difficulty d.

Whether by virtue of network delay or by intentional deviation from the protocol, the value of V_1 in the block might be greater than the lowest proof value overall. In such cases, we emphasize this difference by referring to the actual lowest proof value as the *1OS* (first order statistic).

Like other PoW blockchains, miners select new nonces and generate proofs continuously. In Section VII, we show how a miner can precisely determine the probability that any given proof will eventually be included in the mining package. Once a proof is discovered having sufficient probability of inclusion, the values in $\mathcal{P}_i$ are propagated. Proof sets corresponding to proofs with sufficiently low value (according to a threshold to be defined later) are propagated throughout the network.

We say that a block can be *assembled* when *(i)* the mean of the k proofs, $V_1, \ldots V_k$, is less than or equal to target t_k; *(ii)* the package is signed by the miner who generated V_1 using address a_1; *(iii)* supports $s_2, \ldots s_k$ are greater than or equal to V_1.

Current blockchains place the coinbase reward within the Merkle root. In Bobtail, these rewards are unknown at the time of mining. Therefore, the set $\mathcal{C}$ of k coinbase transactions is listed separately. The coinbase payout of the block is distributed according to the scheme described in Section VIII. Because signature $\mathcal{S}$ covers set $\mathcal{C}$, and the signature is tied to the lowest proof value, the coinbase values are determined entirely by the block miner. However, any block whose set $\mathcal{C}$ fails to follow protocol conventions is considered invalid, and will be ignored by other miners.

The assembled block, comprised of $\mathcal{H}, \mathcal{K}, \mathcal{C}, \mathcal{S}$, and $\mathcal{T}_1$, is propagated throughout the network. $\mathcal{T}_1$ and the proof package $\mathcal{K}$ can both be expressed very efficiently using Graphene [36, 38] or a similar compression algorithm. Receivers validate the three assembling rules stated above, that the signature uses a_1 from $\mathcal{P}_1$, and that the coinbase is allocated fairly. If validation succeeds, then the block is added to the blockchain and propagated to peers.

Difficulty adjustment. Bobtail is compatible with any deployed difficulty adjustment algorithm (DAA). For example, Bitcoin adjusts roughly once every two weeks[1]. It uses the mean block time for the last 2016 blocks to estimate the actual difficulty at which the miners were operating; then the difficulty parameter d is adjusted up or down in order to ensure that the expected block time is 10 minutes if miners continue mining at the same rate. At a given difficulty d, the target t_k can be derived from d in the same manner that it is for Bitcoin[2], which involves translating integer d into a threshold 256-bit arithmetic value (i.e., one that supports arithmetic operations). The DAAs used in Bitcoin Cash [9] and Ethereum [21] are similarly fully compatible with Bobtail.

Fork-choice rule. If multiple miners generate proofs with value low enough to mine a block with the same parent, then there can arise ambiguity over which extends the *main chain*, i.e. the chain that honest miners will continue to extend. To avoid this ambiguity, we define the main chain to be the one comprising the most *aggregate work*, from the genesis block up to the

[1] https://github.com/bitcoin/bitcoin/blob/78dae8cacc/src/pow.cpp#L49
[2] https://github.com/bitcoin/bitcoin/blob/78dae8cacc/src/pow.cpp/#L80

tip; all competing chains are *orphaned*, i.e. ignored by honest miners. Aggregate work is calculated as the sum of inferred hashes, S/w_k, over each block, where S is the size of the hash space and w_k is the value of the mining statistic. In general, this fork-choice rule ensures that blocks with lower average proof values will be favored over those with higher values. Note that because proof sets reference a specific parent block, they can only be shared between child blocks having the same parent. And according to our fork-choice rule, ultimately the child block with the lowest value w_k will extend the main chain.

Additional rules. In order to reduce the number of orphaned blocks (discussed in Section VI-A) and thwart various attacks, (see Section III), miner M will adhere to the following rules. *(i)* M rejects proof package $\mathcal{K}$ if V_1 is higher than the lowest proof she has seen announced on the network (the 1OS). However, she continues to mine on the same prior block until a block containing the 1OS is actually propagated. M does this in order to mitigate a possible DoS attack by the miner of the 1OS and also in hopes of generating more of her own proofs to be included in the block. *(ii)* When assembling a proof package as the miner of V_1, M will include her own proofs first, and then proofs from other miners in the order she received them from the network. Specifically, M begins by identifying all sets of k proofs $S_1, \ldots$, each with mean value below t_k. Let r be the maximum reward value, across all S_i, that would be returned to M if the given set was assembled into a block. She discards any sets that do not return reward value r, and then assembles the proof package from the remaining set with the earliest average proof receipt time so that the mean remains below t_k.

III. Assumptions and Methodology

A. Threat Model

In this section, we list the set of known attacks that can be carried out on Bobtail. Doublespend, selfish mining, and eclipse attacks exist on current blockchains, but the other two are unique to Bobtail. Solutions to each attack, except eclipse, are presented in subsequent sections.

Attacker Model. We assume a straightforward attacker model: attackers are assumed to have some significant proportion of the network's total mining power, but less than 50%. We are interested in attacks on Bobtail's design only, and thus we assume the attacker ignores aspects of the blockchain that are orthogonal, such as hacking into the systems of other miners.

Doublespend. In a doublespend attack [35, 37], the attacker purchases off-chain goods or services from a merchant using an on-chain transaction T. Assume the main chain ends with block B_0. Honest miners will add a sequence of blocks $\mathbf{H} = H_1, H_2, \ldots$ after B_0, with transaction T appearing in block H_1. In the meantime, the attacker mines an alternate fork, also beginning from B_0, with a fraction q of the mining power, producing blocks $\mathbf{A} = A_1, A_2, \ldots$. Block A_1 contains a transaction T' that conflicts with T, allowing the attacker to avoid paying the merchant. To thwart the attack, the merchant selects a value $z \geq 0$ and does not release the goods or services until the length of $\mathbf{H}$ is at least z. Larger values of z decrease the probability of success for the attacker. When $z = 0$, the item is released immediately. The attacker withholds $\mathbf{A}$ until its length is at least $z + 1$ and is also longer than the honest miners' fork. Upon announcement of $\mathbf{A}$, the honest miners will adopt it along with the conflicting transaction T' instead of T.

Selfish mining. A selfish mining attack [23, 26, 41] is another strategy where an attacker attempts to increase her proportion of blocks (and rewards) to an amount above the proportion she has of the network's mining power by causing honest miners to waste some of their work. We follow the attack as described by Eyal and Sirer [23], which unfolds as follows. The attacker withholds a secret chain $\mathbf{A} = A_1, A_2, \ldots$ forked from B_0, releasing one block at a time only when she is ahead by two, and continuing until honest miners are able to produce a competing chain $\mathbf{H} = H_1, H_2, \ldots$ of equal or greater length. At this point, the attacker releases the remaining private portion of $\mathbf{A}$. If $\mathbf{A}$ is the same length as $\mathbf{H}$ at the time of release, then some (non-empty) set of all miners (including the attacker) will adopt $\mathbf{A}$. Thus, $\mathbf{A}$ will be extended with some positive probability, nullifying the work performed on chain $\mathbf{H}$.

Doublespend and selfing mining attacks are the two most fundamental attacks on blockchains. In both cases, because the attacker is assumed to have a minority of the mining power, in expectation, he cannot create a longer fork than that of the honest miners. However, just like a person visiting a casino, the attacker is seeking a short-term win. He is attempting to get lucky and find a series of blocks quickly while the honest miners are relatively unlucky and discover blocks slowly, despite their larger amount of mining power. Intuitively, the success of the attacks lies in leveraging the inherent variance of mining. We show how Bobtail's low variance inter-block time defeats these attacks in Section V.

Eclipse. Most public blockchains arrange fully validating nodes (including miners) in a p2p network [17]. An attacker *eclipses* [28] a peer in the network by conspiring to take over all of its incoming and outgoing connections. By doing so, the attacker can censor both the set of transactions and blocks sent and received by the peer. As a result, the attacker can effectively eliminate the hash rate of the targeted peer by refusing to forward the peer's blocks. For example, in Bitcoin, if the peer has a significant fraction of the hash rate, then it becomes easier for the attacker to doublespend on the main chain. Bobtail experiences the same susceptibility to eclipse attacks as Bitcoin because the attacker can censor proofs the same way he would censor blocks. In particular, an attacker with fraction x of the hash rate who can eclipse fraction y of the honest hash rate, will increase his effective hash rate from x to $x/(1-y)$. We do not focus further on eclipse attacks in this work because they amount to attacks on network topology and management mechanisms, which are largely orthogonal to the consensus mechanism developed by Bobtail. Moreover, mitigations have been introduced [28] for the eclipse attack in existing blockchains.

Proof withholding. This attack involves miner A declining to publish some subset of her Bobtail proofs immediately after they are generated. Instead, A withholds the proofs in order to gain an informational advantage over the remaining miners, H. While miner A sees all proofs, an honest miner in H sees only proofs generated by members of H. A hopes that this advantage will allow her to assemble some proof packages

with more than her fair share of proofs and ultimately lead to an increase in her total reward. Section VIII-B describes how honest Bobtail miners defeat this attack by using supporting proofs and assembling blocks using the earliest arriving proofs.

Denial-of-Service. We distinguish denial-of-service (DoS) attacks from proof withholding by the property that, for the former, a miner elects not to release a *complete* block when he is capable of doing so. Such an attack is clearly disincentivized in current PoW protocols like Bitcoin because of the opportunity cost associated with losing the mining reward. However, opportunity cost is less obvious in Bobtail because the proofs generated by a miner are eligible for inclusion in multiple potential blocks.

B. Methodology

In this paper, our primary conclusions are based on theoretical analysis, which are subsequently validated using a detailed Monte Carlo simulation[3]. In the simulation, the mining process is modeled as sampling from a uniform distribution and comparing those samples against a target. For each trial, we set a target and count the number of samples taken until the target is reached. This procedure yields a valid approximation of mining time assuming a constant hash rate, in which case block generation times can be inferred from a count of hashes performed. For each plot in the paper, we computed many thousands of trials such that the 95% confidence intervals are sometimes too small to be shown or are represented as (very small) intervals. Every point on each plot is a separate set of trials to ensure independence. In some simulations we measured block times only; in other simulations, sharing the same code, we awarded coinbase and included network delays required to examine the operation of the full Bobtail protocol and operation. In our simulations, we do not model the network topology, which is to say that all miners are nominally connected in a clique. Nevertheless, we feel that the latencies we introduce, where appropriate, are realistic for the networks they model [24].

IV. Low Variance and other properties of the k-OS Criterion

In this section, we derive the statistical characteristics of Bobtail. Our primary goal is to prove that the inter-block time variance decreases as $O(1/k)$.

Roadmap. To derive the reduction in Bobtail's inter-block time variance, we proceed in four steps. We begin by laying a statistical foundation upon which our results are built. In Section IV-A, we derive an equation for Bobtail's PoW target, which is a function of k. In Section IV-B, we use the target t_k to derive an expression for the expected amount of time it takes to mine a block, given k. In Section IV-C, we use this expectation to calculate the inter-block time variance, and finally, in Section IV-D, we compare as a ratio the variance of mining with $k > 1$ to current systems (i.e., $k = 1$). The primary challenge in this derivation is in linking the statistics of PoW sample values with mining time. The former is dictated directly by the protocol (through mining criterion $w_k < t_k$), while the latter is emergent.

[3]https://github.com/umass-forensics/bobtail-simulations

Foundation. To begin, we state a fundamental relationship between the size of the hash space S, the total number of hashes performed h, and the expected value of the minimum hash v found during an *interval* of time I when miners are hashing. (I is any duration, but one could think of it as, for example, the desired inter-block time.)

Recall from Section II-A that V_i represents the ith lowest hash value achieved after h hashes are performed during interval I. Next, define X_i to be the *number of intervals* I required for V_i to fall below v when h hashes are performed per interval. In Appendix A, we show that both V_i and X_i are gamma distributed with shape parameter i, only differing in scale parameter. Specifically, Theorem 6 proves that

$$V_i \sim \texttt{Gamma}(i, v), \tag{3}$$

and Theorem 7 proves that

$$X_i \sim \texttt{Gamma}(i, 1/r), \tag{4}$$

where v is the expected value of the minimum hash (V_1) during interval I and $1/r$ is the expected number of intervals required for V_1 to fall below v.

We can see that the distributions for V_i and X_i are related through the change of variables $v = 1/r$, and all four parameters v, r, h, and S are related by

$$v = r\frac{S}{h}. \tag{5}$$

In words, Eq. 5 states: the expected minimum hash v during interval I is related to the expected rate at which the first order statistic falls below v by the ratio $\frac{S}{h}$.

A. Adjustment of the Target Given k

Intuitively, if we increased $k > 1$ without adjusting the target for $k = 1$, then the expected block time would increase: the average of the k lowest order statistics has higher expected value than the first order statistic alone, so we expect that more hashes are required for the average to fall below a given value. Therefore, to keep the number of hashes expected to find a block constant as we increase k, the target t_k appropriate for each k should also be increased. In this subsection, we determine the relationship between t_i and t_j, $i \neq j$, such that the expected time to mine a block using criterion $w_i < t_i$ is the same as when using the criterion $w_j < t_j$.

Define $X'_i = X_i/i$ and $V'_i = V_i/i$ to be the *normalized* value and interval count, respectively, for order statistic i. Furthermore, define

$$W'_k = \frac{1}{k}\sum_{i=1}^{k} V'_i$$

to be the *normalized mining statistic*. We have the following.

LEMMA 1: *In expectation,* $1/r$ *intervals are required to ensure that* $V'_i = V_i/i < v$ *for all* i*.*

PROOF: As stated above, X_i is gamma distributed so that $E[X_i] = \frac{i}{r}$. Each X'_i is interpreted as the number of intervals required for V_i to fall below iv, or analogously, the number of intervals required for V'_i to fall below v. Since $E[X'_i] = E[iX_i] = \frac{1}{r}$ for all $i \geq 1$, it follows that we expect $V'_i < v$ after $1/r$ intervals, as required.

□

The next result shows how mining statistic W_k is related to the expected value of the minimum hash.

LEMMA 2: *The expected value of W_k is $E[W_k] = \frac{k+1}{2}v$.*

PROOF: Starting from the expectation of Eq. 1:

$$\begin{aligned} E[W_k] = E\left[\frac{1}{k}\sum_{i=1}^{k} V_i\right] &= \frac{1}{k}\sum_{i=1}^{k} E\left[V_i\right] \\ &= \frac{1}{k}\sum_{i=1}^{k} iv \\ &= \frac{k+1}{2}v. \end{aligned} \tag{6}$$

□

Now we are in a position to prove our desired lemma.

LEMMA 3: *In expectation, $1/r$ intervals are required to ensure $W_k < t_k$, for all $k > 0$, provided that t_k is chosen such that*

$$t_k = \frac{k+1}{2}v. \tag{7}$$

PROOF: Define $T(W_k, t_k)$ to be the expected number of intervals required for W_k to fall below t_k. It will suffice to show that $T(W_k, (k+1)v/2) = \frac{1}{r}$ for all $k > 0$. Unfortunately, it is difficult to reason directly about $T(W_k, t_k)$, but it is straightforward to reason about $T(W'_k, v)$. From Lemma 1, we expect each V'_i to fall below v after exactly $1/r$ intervals. So it must also be the case that $T(W'_k, v) = \frac{1}{r}$. Using reasoning similar to that in Lemma 2, it can be shown that $E[W'_k] = v$. Therefore, $\frac{k+1}{2}W'_k$ is an unbiased estimator of W_k, and we expect that $T(W_k, t_k) = T(\frac{k+1}{2}W'_k, t_k)$. Finally, we have the following.

$$\begin{aligned} T\left(W_k, \tfrac{(k+1)v}{2}\right) &= T\left(\tfrac{2}{k+1}W_k, v\right) \\ &= T(W'_k, v) \\ &= \tfrac{1}{r}. \end{aligned} \tag{8}$$

□

B. Estimating Mining Time

In this section, we derive a consistent estimator of block mining time for mining statistic W_k assuming that $t_k = \frac{k+1}{2}v$ according to Lemma 3.

Consider the following choice of estimator for the overall number of intervals required to ensure $W_k < t_k$:

$$Y_k = \frac{2}{k+1}\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right). \tag{9}$$

We next derive the expected value and establish the consistency of Y_k as an estimator of mining time.

LEMMA 4: *Assuming that $t_k = \frac{v(k+1)}{2}$, Y_k is a consistent estimator of the expected number of intervals*

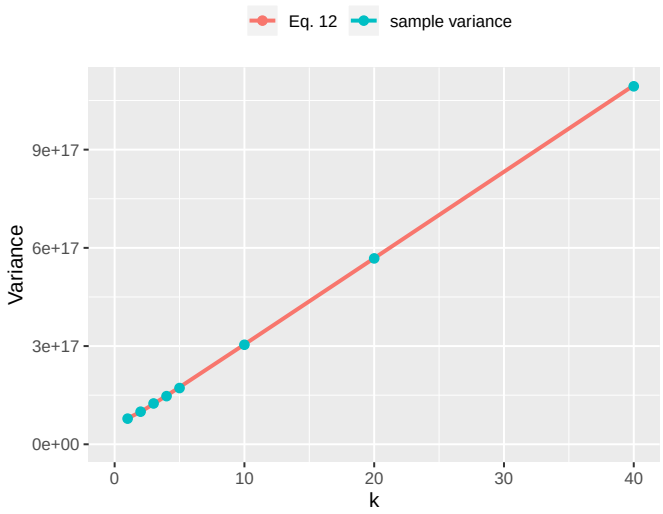

Fig. 3: We measured the variance of Bobtail's average proof of work statistic using a Monte Carlo simulation and it has the same values as predicted by Eq. 12 from Theorem 1. 95% c.i.'s are too small to show.

required for W_k to fall below t_k with

$$E[Y_k] = \frac{1}{r}, \tag{10}$$

where $r = vh/S$.

PROOF: Beginning with Eq. 9 we have,

$$\begin{aligned} E[Y_k] &= E\left[\tfrac{2}{k+1}\left(\tfrac{1}{k}\sum_{i=1}^{k} X_i\right)\right] \\ &= E\left[\tfrac{1}{k}\sum_{i=1}^{k} X'_i\right] \\ &= \tfrac{1}{k}\sum_{i=1}^{k} E[X'_i] \\ &= \tfrac{1}{k}\sum_{i=1}^{k} \tfrac{1}{r} \\ &= \tfrac{1}{r}. \end{aligned} \tag{11}$$

Now Lemma 1 establishes that $E[V_i] = iv$ after mining for $1/r$ intervals. And because we assume $t_k = \frac{k+1}{2}v$, the target can be decomposed as

$$t_k = \frac{k+1}{2}v = \frac{1}{k}\sum_{i=1}^{k} iv.$$

That is to say, t_k is simply the average of the expected values for each V_i when mining for $1/r$ intervals. Of course W_k is the sample average of the V_i. Therefore, by the law of large numbers, in the limit that k approaches infinity, W_k will identically equal its expected value $\frac{k+1}{2}v$ and will thus fall below t_k after exactly $E[Y_k] = 1/r$ intervals.

□

C. Variance of W_k

Our third step is to derive and validate an expression for $Var[W_k]$. W_k is simply the sample mean over the lowest k order statistics $V_1, \ldots, V_k$. But, unfortunately, the analysis below

is not straightforward because the V_i are neither independent nor identically distributed.

THEOREM 1: *The variance of W_k is*

$$Var[W_k] = \frac{(k+1)(2k+1)}{6k} v^2. \tag{12}$$

PROOF: Assuming that $j > i$, Lemma 8 in the Appendix yields

$$\begin{aligned} E[V_i V_j] &= \int_0^\infty \int_{t_i}^\infty t_i t_j g(t_i; i, v) g(t_j - t_i; j - i, v) dt_j dt_i \\ &= \int_0^\infty t_i g(t_i; i, v)[(j-i)v + t_i] dt_i \\ &= iv^2(1+j). \end{aligned} \tag{13}$$

Before continuing, we note that since $V_i \sim \texttt{Gamma}(i, v)$, it follows that $Var[V_i] = iv^2$. Now, assuming that $j > i$, and using Eq. 13, we have

$$\begin{aligned} \texttt{cov}[V_i, V_j] &= E[V_i V_j] - E[V_i]E[V_j] \\ &= iv^2(1+j) - (iv)(jv) \\ &= iv^2 \\ &= Var[V_i]. \end{aligned} \tag{14}$$

Finally, we find the variance of W_k by substituting first Eq. 1 and then Eq. 14:

$$\begin{aligned} Var[W_k] &= Var\left[\frac{1}{k}\sum_{i=1}^{k} V_i\right] \\ &= \frac{1}{k^2}\left(\sum_{i=1}^{k} Var[V_i] + 2\sum_{j=1}^{k}\sum_{i=1}^{j-1} \texttt{cov}[V_i, V_j]\right) \\ &= \frac{1}{k^2}\left(\sum_{i=1}^{k} iv^2 + 2\sum_{j=1}^{k}\sum_{i=1}^{j-1} iv^2\right) \\ &= \frac{(k+1)(2k+1)}{6k} v^2. \end{aligned} \tag{15}$$

□

Empirical Validation of Theorem 1. Fig. 3 shows Eq. 12 versus our Monte Carlo simulation where k is the independent variable. The results show an exact match.

D. Improvement in Variance

Finally, we turn our attention to quantifying the reduction in mining time variance that is realized by using Bobtail with $k > 1$ versus the status quo in PoW mining, $k = 1$. In Section IV-B we established that statistic Y_k is a consistent estimator of the number of intervals required for mining statistic W_k to fall below target $t_k = \frac{k+1}{2}v$. Here we measure the change in variance of Y_k as k increases, while holding its expected value constant.

First note that because X_i shares the same distribution as V_i, up to the change of variables $v = 1/r$, the following result follows trivially from Theorem 1.

$$Var\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right] = \frac{(k+1)(2k+1)}{6k}\left(\frac{1}{r}\right)^2. \tag{16}$$

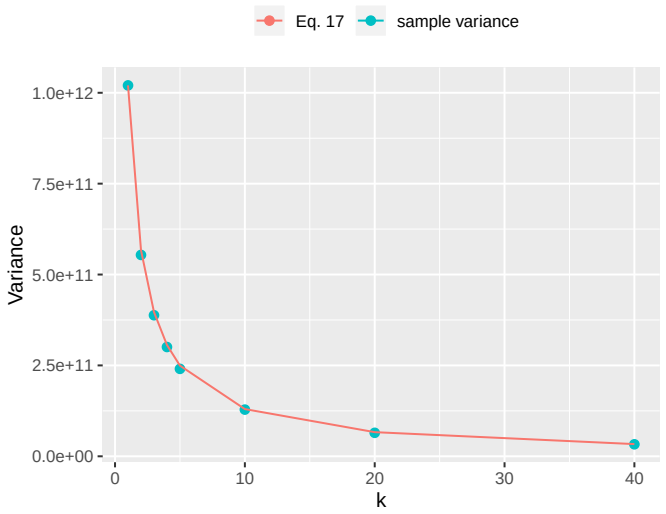

Fig. 4: We measured the ratio of Bobtail's block discovery-time variance to current systems using a Monte Carlo simulation and found that the results exactly match $\frac{Var[Y_k]}{Var[Y_1]} = \frac{8k+4}{6(k^2+k)}$ (Eq. 17) from Theorem 2. 95% c.i.'s are too small to show.

THEOREM 2: *For fixed expected block discovery time, variance decreases by fraction $\frac{8k+4}{6(k^2+k)} = O\left(\frac{1}{k}\right)$ when using mining statistic W_k instead of W_1.*

PROOF: Lemma 4 establishes that block discovery time Y_k is the same in expectation for all mining statistics W_k provided that $t_k = \frac{v(k+1)}{2}$. Therefore, the ratio of variance in Y_k to the variance in Y_1 estimates the reduction in block time variance due to Bobtail.

$$\begin{aligned} \frac{Var[Y_k]}{Var[Y_1]} &= \frac{Var\left[\frac{2}{k+1}\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right)\right]}{Var[X_1]} \\ &= \frac{4}{(k+1)^2}\frac{\frac{(k+1)(2k+1)(1/r)^2}{6k}}{(1/r)^2} \\ &= \frac{4}{(k+1)^2}\frac{(k+1)(2k+1)}{6k} \\ &= \frac{8k+4}{6(k^2+k)} \\ &= O\left(\frac{1}{k}\right), \end{aligned} \tag{17}$$

where we use the expression for $Var\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right]$ stated in Eq 16.

□

Empirical Validation of Theorem 2. In Fig. 4, we show the results from a Monte Carlo simulation that compares the variance of Y_k, the block discovery time under mining statistic W_k, to the variance of Y_1, the mining statistic used by current PoW algorithms. The results exactly match Eq. 17 from Theorem 2.

Fig. 5 shows the distribution of Y_k when $t_k = v(k+1)/2$ so that $E[Y_k] = E[Y_1]$. The plot shows the cumulative distribution function (CDF) based on the results of a Monte Carlo simulation. As the plot illustrates, the use of Bobtail mining results in a significant decrease in variance in block discovery time.

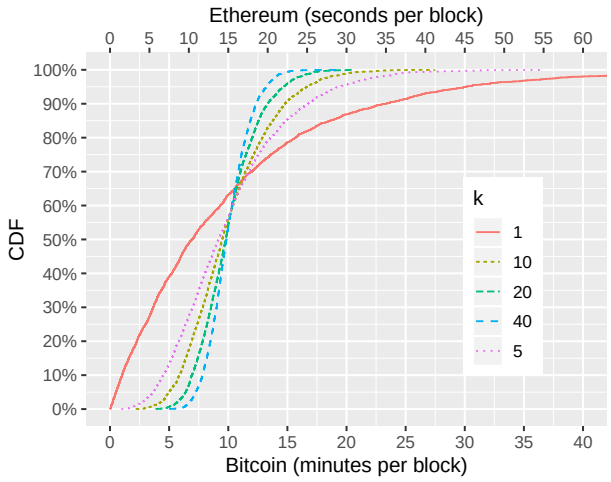

Fig. 5: Results of a Monte Carlo simulation showing the CDF of Y_k, the block discovery time under mining statistic W_k, where k varies for each curve and target t_k is chosen so that $E[Y_k] = E[Y_1]$. Each plot's independent axis is shown in terms of the minutes per block for Bitcoin (bottom axis) and seconds per block for Ethereum (top axis).

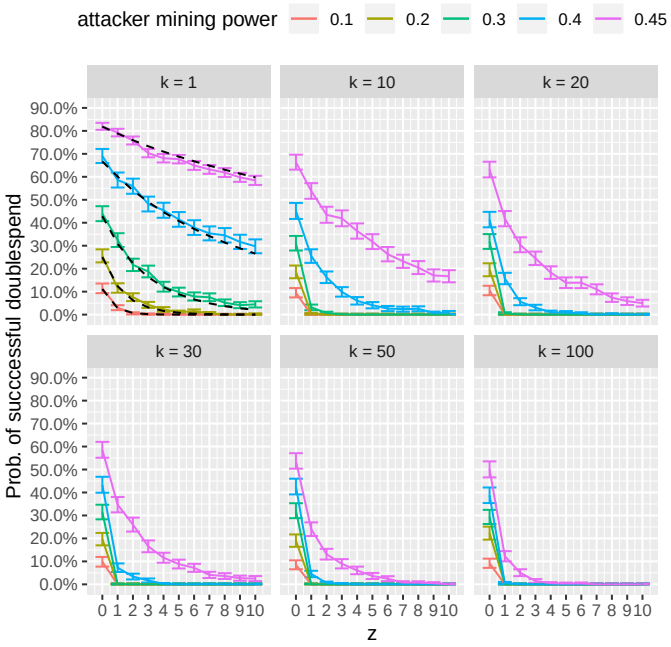

Fig. 6: Monte Carlo Simulation. Doublespend attack success given k for various values of attacker mining power (each curve) and merchant's embargo period z (on the independent axis). Error bars show 95% c.i.'s. The dashed lines show the probability of success predicted by Nakamoto's model for $k = 1$ [35, 37].

V. Bobtail thwarts selfish mining and doublespend attacks

We next demonstrate quantitatively that its reduced inter-block-time variance allows Bobtail to thwart both *double-spend* [35] and *selfish mining* [23] attacks. Note that, in our experiments below, only the variance of the inter-block time changes as k increases, leaving the expected time unchanged, and hence we can ascribe the increased resilience against these attacks to Bobtail.

Recall the definitions of doublespend and selfish mining from Section III; in particular the honest and attacker forks are composed of blocks $\mathbf{H} = H_1, H_2, \ldots$ and $\mathbf{A} = A_1, A_2, \ldots$, respectively. And, in the context of the Bobtail protocol, each block is itself comprised of k proofs, which are generally contributed by both honest and attacker miners. For both attacks, Bobtail allows the attacker two additional strategies when $k > 1$. First, because they point to the same prior block, proofs can be reused between A_1 and H_1. As a result, while the lowest proof in A_1 must be her own, the attacker can include any proofs from the honest miners that help her reach the target. And second, while mining A_1, the attacker need only withhold proofs that are lower than the honest miners' lowest proof in order to delay the creation of a block. Thus, her larger proofs may still appear in H_1, which reduces her cost if the attack fails. We include these strategies in our evaluations of Bobtail.

Fig. 6 shows a Monte Carlo simulation of the doublespend attack. To ensure the attack has a finite duration, the attacker gives up when the honest branch is $3z + 1500$ blocks ahead. Each facet of the plot represents a value of k. The results show that as k increases and variance decreases, the probability of attacker success significantly decreases. For example, in today's implementations of both Bitcoin and Ethereum ($k = 1$), an attacker with 40% of the mining power will succeed with 37% probability on average when $z = 6$; however, using Bobtail with $k \geq 20$, the probability of success falls to less than 0.5% on average for the same scenario.

Fig. 7 shows the efficacy of selfish mining attacks via a Monte Carlo simulation. In the simulation, during a block propagation race, with probability γ the attacker's block propagates to other miners before any block belonging to an honest miner can. Fig. 7(Top) shows the proportion of *blocks* on the main chain won by attackers for $\gamma = 0$ and $\gamma = 1$; and Fig. 7(Bottom) shows the proportion of *rewards*. The dashed identity line ($y = x$) represents the proportion that are won mining honestly. Any selfish mining result that is below the identity line is worse for the attacker than honest mining.

As the results demonstrate, Bobtail is a significant defense against selfish mining. For example, a selfish miner with 45% of the mining power will claim about 71% of blocks with Bitcoin and Ethereum currently ($k = 1$ and $\gamma = 1$); however, using Bobtail with $k \geq 20$ and $\gamma = 1$, the same miner will find that selfish mining is less successful than honest mining whether in terms of the fraction of blocks or rewards.

Finally, we note that while Bobtail is not more robust against eclipse attacks [28], its low variance block times make such attacks easier to detect.

As stated in Section III, other attacks are possible for Bobtail. We return to them in Section VIII.

VI. Bobtail's orphan rates are the same or lower

Even when all miners operate honestly, current blockchain systems frequently suffer from orphaned blocks during their operation that diminish security and delay consensus. Orphans

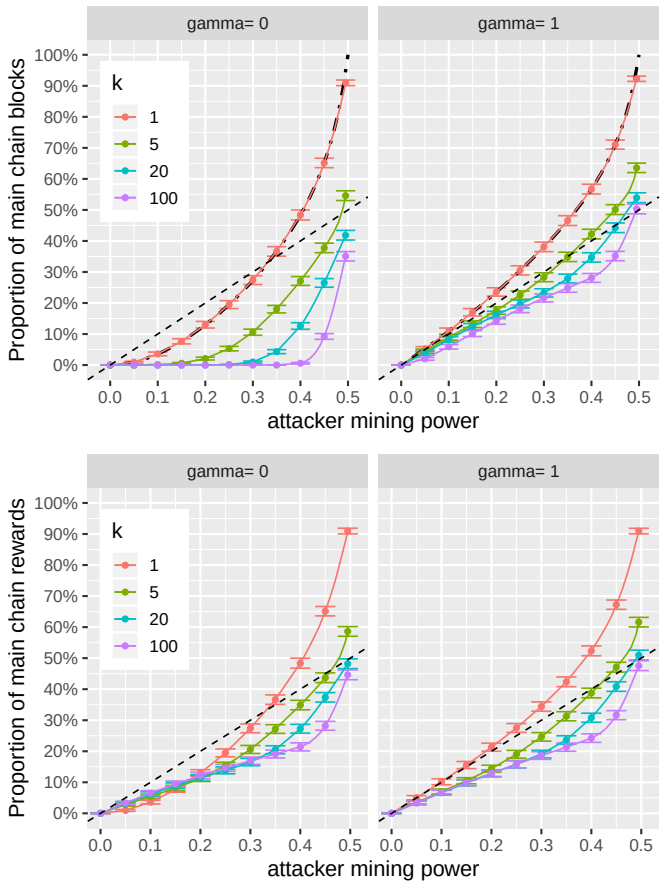

Fig. 7: Monte Carlo Simulation. Selfish mining attack success in terms blocks mined (top) and rewards captured (bottom) given k (different curves) for various values of attacker mining power q (on the independent axis); $q \leq 0.495$. The straight dashed identity line shows the results of honest mining; the curved dash-dotted line shows analytical results from Eyal and Sirer [23, Eq. 8]. Error bars show 95% c.i.'s.

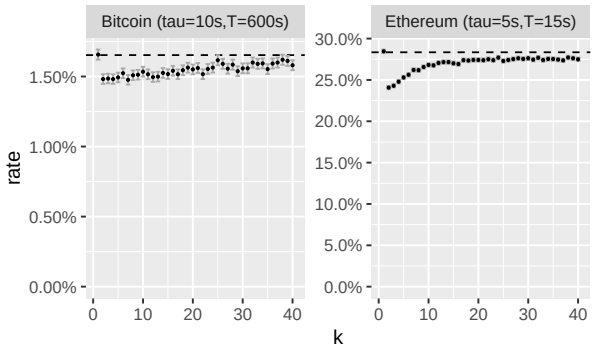

Fig. 8: A simulation of Bobtail's orphan rate when proofs and blocks propagate with constant delay of $\tau = 10$ seconds and the inter-block time is $T = 600$ seconds. The orphan rate of Bobtail is at or below the expected orphan rate, $1 - 1/e^{\tau/T}$ for $k = 1$ (shown as a dashed line). Similar results hold for Ethereum where $\tau = 5$ and $T = 15$ seconds. Error bars represent a 95% c.i.

are generated when the announcement of a new block by one miner takes time to propagate to all other miners. In the interim, a second miner may produce a valid block. At that point, the subset of miners who received the first block first will attempt to build upon it, and the remaining miners will build upon the second. Eventually the blockchain will fork on just one of those blocks, orphaning the other. If the set of transactions in the two blocks is not the same, then consensus is delayed. While the occurrence of orphans in Bitcoin is relatively low, Ethereum's use of a 15-second average block discovery time increases its orphan rate significantly.

In Appendix A, we show that X_i, the number of block intervals required to mine the ith order statistic, has distribution $\texttt{Gamma}(i, 1/r)$, where $1/r$ is the expected number of intervals I required for V_i to fall below v. Interpreting v as the target and letting I equal one second, it follows that X_1 represents the block inter-arrival time, and it has distribution $\texttt{Exponential}(1/r) = \texttt{Exponential}(T)$, where T is the expected block time in seconds. Therefore, in existing PoW blockchains, the probability that one or more other blocks will be discovered during propagation time τ is bounded by $1 - \frac{1}{e^{\tau/T}}$ (see also Rizun [40]). Note that this bound is pessimistic in that it assumes the worst case scenario where the author of the first block has a negligible percentage of the total network hash rate.

In this section, we examine the orphan rate associated with Bobtail mining compared to Bitcoin and Ethereum. We show that when miners follow the Bobtail protocol, orphans are no more likely.

A. *Orphan Prevention Measures*

The principal cause of orphans in Bobtail is the fact that, once more than k proofs have been disseminated, there exist a combinatorial number of k-element subsets of those proofs. Thus at the time when there exists some subset of k proofs whose mean falls below target t_k, there is a reasonable chance that *some other* subset also exists (or will exist relatively soon). Fortunately, the proof package rules introduced in Section II greatly reduce the number of valid subsets. First, all proofs must be tethered to a *supporting proof*, the latter of which should be the smallest proof previously received by the miner. Second, no support in the proof package can have value less than V_1 (the lowest proof in the package). And third, the block must be signed by the private key used to generate V_1.

Together, these conditions ensure that if at least one of the k proof sets in the proof package points to V_1 as support (excluding the support for V_1 itself), then the creator of V_1 is the *only* miner capable of assembling that proof package. Conversely, although another miner, say the one who generated V_2, might be capable of collecting a set of proofs that exclude V_1 but still have mean below target t_k, that miner cannot assemble a proof package if even a single proof set includes V_1 as support.

B. *Performance*

We ran a discrete event simulator to determine the efficacy of the orphan prevention measures described in Section VI-A. The simulation includes only honest miners: once he has received a valid proof package, an honest miner does not release a

competing proof package of his own. We evaluate attacks on Bobtail subsequently.

The simulation generates blocks by repeatedly selecting values uniformly at random between 0 and 2^{32}. The smallest k values are used to assemble a candidate block given a pre-set target value. The propagation delay of new proofs and blocks is τ seconds. Once a block is found, we assume that the authoring miner drops out and her mining power is replaced by a new honest miner; i.e., the hash rate does not change. For τ seconds, the miners continue seeking a new block following the rules in Section II. For example, if they find a block is possible with a higher V_1, they will not release the new block.

Fig. 8 (left) shows the results for a Bitcoin-like scenario where the inter-block time is targeted at $T = 600$ seconds and the propagation delay is $\tau = 10$ seconds. The orphan rate for $k = 1$ follows the expected exponential distribution, shown as a dashed line. The experiment shows that, across multiple values of k, the 95% confidence interval for orphan rate in Bobtail consistently falls at or below the $k = 1$ rate. Fig. 8 (right) shows the same result for a simulation of Ethereum where $\tau = 5$ and $T = 15$ seconds, respectively.

VII. Lowering Network Overhead

When the mining statistic is W_k, $k \gg 1$, it is not efficient for each miner to send proof of work every time she finds a hash value lower than her previous k best. A slight improvement to that scheme is for her to send proof of work only when her hash value is lower 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 hash values are generated uniformly at random throughout the mining interval (see Lemma 5); the k lowest overall are unlikely to be generated early in the mining process.

To improve network efficiency significantly we instruct miners to propagate proof sets only if the associated proof is at least minimally likely to be among the k lowest. To that end, we seek a proof value x such that the k lowest proofs will fall below x with probability p, $p \approx 1$.

We know from Theorem 6 in Appendix A that the kth order statistic V_k has distribution $\texttt{Gamma}(y; k, v)$, where v is the expected minimum proof value. The inverse of that distribution is $\texttt{Quantile-Gamma}(p; k, v)$, which returns the value y for which $\texttt{Gamma}(y; k, v) = p$. Therefore, a natural choice for our bounding proof value is

$$x = \texttt{Quantile-Gamma}(p; k, v). \tag{18}$$

The following theorem establishes the expected number of proofs forwarded per block when we propagate only the proofs lower than x.

THEOREM 3: *For proof value threshold x defined by Eq. 18, the expected number of proofs announced to the network is $\texttt{Quantile-Gamma}(p; k, 1)$.*

PROOF: Note from Eq. 5 that $v = S/h$ when t_k is tuned for blocks to be generated in a single interval I (i.e., $r = 1$). We expect h hashes per block interval, and each has probability x/S of being below x. Therefore, the random variable representing the number of proofs forwarded by all miners follows distribution $\texttt{Binomial}(n = h, p = x/S)$, which has expectation:

$$\begin{aligned} \frac{hx}{S} &= \frac{1}{v} \cdot \texttt{Quantile-Gamma}(p; k, v) \\ &= \texttt{Quantile-Gamma}(p; k, v/v) \\ &= \texttt{Quantile-Gamma}(p; k, 1) \end{aligned} \tag{19}$$

□

Notably, for $p = 0.999999$, the value remains quite low. For example, when $k = 40$, only about 80 proofs are sent on the network. Further, the value is independent of h, the expected number of hashes required to mine a block, as well as S, the size of the hash space. We can also use a Chernoff bound for the binomial distribution to bound the deviation in the number of messages M. Let $z = \texttt{Quantile-Gamma}(p; k, 1)$. We have,

$$\Pr[M \geq (1 + \epsilon)z] \leq e^{\frac{-z\epsilon^2}{2+\epsilon}}. \tag{20}$$

This is a tight bound, and it decreases exponentially with z and similarly with k. For example, when $k = 2$ and $p = 0.999999$, then $z \approx 16.7$, and we see that $P(M > 1.9z) \leq 0.0095$. For $k = 3$, the probability decreases to 0.004, and so on.

VIII. Incentivizing Honest Behavior with Rewards

In this section, we show that there exists a *reward scheme* (coinbase, ignoring fees) that incentivizes miners to: *(i)* continue mining for increased reward, rather than stopping once any proof is discovered; *(ii)* use the lowest proof they know of as support; and *(iii)* immediately broadcast all sufficiently low proofs. With this reward scheme in place, Bobtail thwarts attacks described in Section III. At a high level, the rewards scheme is as follows. Recall that a proof V_i is the hash of proof set $\mathcal{P}_i$. And *support* s_i, for a proof set $\mathcal{P}_i$, is the lowest proof that the miner has received to date among those sharing the same prior block. We assign rewards as follows.

- To the miners of each proof set $\mathcal{P}_1, \ldots, \mathcal{P}_k$ in the proof package $\mathcal{K}$, we assign *primary reward* R; all proofs in a given package receive the same amount, but the amount may vary from block to block.
- To the miners of each proof set whose support is V_1, we award a *bonus reward* B, which is again the same for every proof pointing to V_1, but may vary by block.

Below, we evaluate the scheme with respect to all three properties first assuming honest miner behavior. Specifically, we determine the expected primary and bonus rewards accrued by an honest miner across all proofs in a given block. We further derive the expected *total reward* T, which is the sum of expected primary and bonus rewards for a miner following the honest strategy. We then show that dishonest miners can expect to receive lower rewards.

A. Analysis of Honest Miners

We begin with a basic result that is useful in contemplating reward distribution.

LEMMA 5: *In expectation, a fraction x of the mining power will generate a fraction x of all proofs as well as*

a fraction x of the k lowest proofs.

PROOF: Without loss of generality, assume a single miner M controls fraction x of the mining power. All hashes generated are uniformly distributed throughout space S. Therefore, of all the hashes that fall within an arbitrary interval of S, miner M expects to have generated fraction x. The interval $[0, S]$ contains all proofs; it is therefore clear that M expects to generate fraction x of all proofs. Moreover, the set of all proofs K that are at or below the kth order statistic defines an interval, $[0, V_k]$. Thus, M expects to generate fraction x of proofs in K as well, which constitutes fraction x of the set of the k lowest proofs.

□

We next analyze the reward payout with respect to our desired mining properties under the assumption that all miners behave honestly, i.e. according to the protocol. Consider a miner M who possesses fraction x of the total mining power. According to Lemma 5, M can expect to have generated fraction x of the k proofs in the proof package. Therefore, M will earn xkR primary reward in expectation. Calculating the expected bonus reward requires the following observation.

LEMMA 6: *The rank (i.e., position in an ascending list by value) of a proof is uncorrelated with the time it is generated.*

PROOF: Let $\mathbf{P} = \mathcal{P}_1^*, \mathcal{P}_2^*, \ldots$ be the set of all proof sets generated during time interval I listed in the order that they appear chronologically such that $\mathcal{P}_i^*$ was generated before $\mathcal{P}_j^*$ if $i < j$. Define $V_i^* = h(\mathcal{P}_i^*)$, and let $\mathbf{V}(\mathbf{P}) = V_1^*, V_2^*, \ldots$ denote the set of all proofs generated during I. It will suffice to show that the probability that $\mathcal{P}_i^*$ achieves a given value $\nu \in \mathbf{V}(\mathbf{P})$, conditioned on all values $\mathbf{V}(\mathbf{P})$, is uniform for all $\mathcal{P}_i^*$.

Being drawn from a uniform distribution, we have that $\Pr[V_i^* = \nu]$ is equal for all proofs $\mathcal{P}_i^*$. Next define $\mathbf{V}_{\setminus\nu}(\mathbf{P}) = \mathbf{V}(\mathbf{P}) \setminus \{\nu\}$, which implies $\Pr[\mathbf{V}(\mathbf{P}) \mid V_i^* = \nu] = \Pr[\mathbf{V}_{\setminus\nu}(\mathbf{P})]$ because $\Pr[V_i^*]$ and $\Pr[V_j^*]$ are independent for $i \neq j$. Thus

$$\begin{aligned} \Pr[V_i^* = v \mid \mathbf{V}(\mathbf{P})] &= \frac{\Pr[\mathbf{V}(\mathbf{P}) \mid V_i^* = \nu] \;\; \Pr[V_i^* = \nu]}{\Pr[\mathbf{V}(\mathbf{P})]} \\ &= \Pr[V_i^* = \nu] \; \frac{\Pr[\mathbf{V}_{\setminus\nu}(\mathbf{P})]}{\Pr[\mathbf{V}(\mathbf{P})]} \\ &= c, \end{aligned} \tag{21}$$

for some constant c.

□

We can use Lemma 6 to show that half of a miner's proof sets in the proof package are expected to be generated after V_1. Thus honest miner M, with fraction x of the hash rate, is expected to generate $\frac{xk}{2}$ proofs that use V_1 as support. It follows that M's expected bonus reward is equal to $\frac{xkB}{2}$. Finally, the expected total reward for the honest miner is given by

$$T_H = xk\,(R + {}^{B}/_{2})\,. \tag{22}$$

From this expression for total reward, we can see that honest mining delivers all three desired mining properties. First, a miner's reward is proportional to her hash rate, which encourages her to mine as much as possible rather than stopping once a proof is found. Second, her total reward is an increasing function of the number of her proofs that point to V_1. And third, because total reward is also an increasing function of the number of proofs in the proof package, she is incentivized to release her proofs as soon as possible so as to give them the greatest chance of being included.

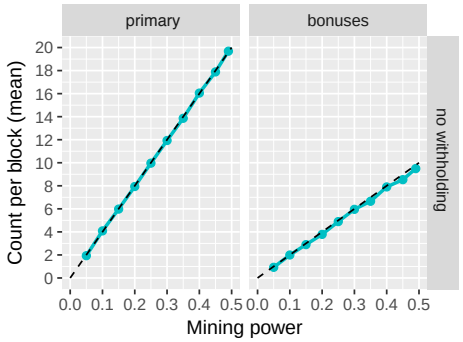

Fig. 9: A Monte Carlo simulation of rewards issued by Bobtail for an honest miner with a given fraction of the mining power. The dotted lines show the predicted value of R and B from Eq. 22.

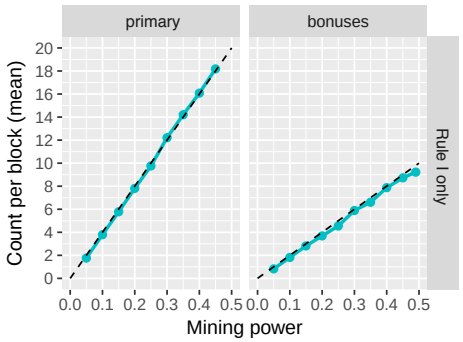

Fig. 10: In these Monte Carlo results, we have modified the simulation so that miners follow Rule I from Section VIII-B; i.e., all miners prioritize their own proofs when assembling the block. With the rule in place, rewards are still issued proportionally.

Fig. 9 shows the results of this rewards scheme from a Monte Carlo simulation of honest miners. The dotted lines show the values predicted by Eq. 22. Next, we demonstrate that dishonest miners earn only fewer rewards.

B. Thwarting Proof Withholding Attacks

Bobtail allows for an attack where a malicious miner withholds proof sets for a competitive advantage. In this section, we demonstrate that our design of Bobtail ensures that the economic reward for withholding attackers is substantially lower than that of honest miners.

In the withholding attack, the malicious miner does not immediately announce her own proof sets to the other miners. This behavior can be advantageous in two ways. First, it gives her more time to mine V_1, which would allow her to control the set of transactions included in the block, $\mathcal{T}_1$. Second, it allows the attacker to *pack* more of her own proof sets into the proof package if she does manage to mine V_1. The attacker

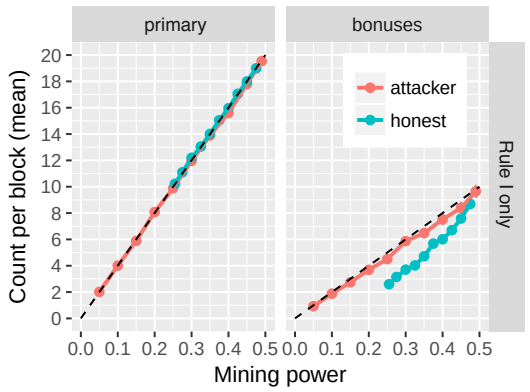

Fig. 11: In this Monte Carlo simulation, we include attacking miners who attempt to withhold their proofs and we **do not** include Rule II from Section VIII-B: although there are no gains for the attacking miners, the honest miners lose out on bonuses.

Fig. 12: In this Monte Carlo simulation, we include attacking miners who attempt to withhold their proofs. In these simulations, we do include Rules I and II from Section VIII-B. As a result, attacking miners give up some of their rewards to honest miners.

mines until either she is able to assemble a block as the miner of V_1; or it is clear that the honest miners are more likely to release a block without her withheld proof sets. In the latter case, she disseminates her withheld proof sets, hoping that some will be included in the proof package.

To thwart the attack, Bobtail includes two simple rules (originally described in Section II-B).

- Rule I: it is considered honest behavior for miners to prioritize inclusion of their own proofs when assembling a block. As a result, no miner benefits more from this behavior than any other.
- Rule II: after prioritizing their own proof sets, if multiple subsets of k proof sets can be used to assemble a block, an honest miner will select proof sets from other miners in the order that they were received locally over the network.

In other words, the proof sets of the withholding attacker are likely left out if withheld too long.

We evaluated this attack using a Monte Carlo simulation, a variation of the one used in earlier sections. Fig. 10 shows the allocation of rewards and bonuses when all miners follow Rule I above: there is no difference from Fig. 9. In short, although it seems selfish to prioritize one's own proof sets, if all miners do it then there is no advantage. Hence, we have eliminated this attack.

Fig. 11 demonstrates the need for Rule II. We modified our simulation so that attackers withhold their proof sets. As a result, the honest miners lose out on bonuses because they do not know the correct supporting proof to list. There is no immediate gain for attackers for doing so, but note that they have effectively lowered the total hash rate of the network, while packing a higher fraction of their proof sets into the proof package than their hash rate should allow. Thus, once the difficulty adjusts to this lower overall hash rate, attackers will gain higher reward than if they behaved honestly. In other words, without Rule II in place, a Bobtail-specific version of the selfish mining attack [23] is possible.

Fig. 12 shows result of the withholding attack when Rules I and II are both in place. Withholding of proof sets from honest miners results in the attacking miner receiving lower primary and bonus rewards. In fact, honest miners gain from the attack, further incentivizing honest behavior.

C. Thwarting DoS Attacks

Bobtail is robust to a denial-of-service (DoS) attack from the holder of the true 1OS who refuses to publish a block. First, the attack is easy for honest miners to detect. In the type of mining environment seen in Bitcoin today — where most of the hash rate is attributable to a set of known miners — one could easily imagine a practical countermeasure would be for honest miners to ban attackers who conduct DoS attacks, refusing to forward any of their proofs. Second, there exists an opportunity cost to attackers in the form of lost bonus reward. And third, even when the attacker has as much as 50% of the hash rate, the expected increase in inter-block time due to DoS attack does not exceed twice the target time. We presently explore the last two points in more detail.

Opportunity cost. Suppose that the attacker holds the 1OS (lowest proof value overall). In a DoS attack, he will release this proof and any others he generates, but will never release an assembled block. Yet the protocol stipulates that honest miners should not accept a block if V_1 is greater than the 1OS (see the first *additional rule* in Section II-B). And in any case, honest miners will also include the 1OS as support, which again means that no greater proof value can be used as V_1. Therefore, miners will be unable to assemble a block until one of them produces a *lower* proof than the attacker's current 1OS (one that is valid as V_1 in the block). At that point, any proof (be it from the attacker or from honest) can be added to the block and receive primary reward, but no proof will receive a bonus because it cannot have pointed to V_1, which just arrived. This amounts to an opportunity cost for the attacker, which we term *bonus cost*, and it is the basis of the following result.

> **THEOREM 4:** *In expectation, and regardless of the attacker's hash rate, at most 1 out of every k blocks can be subject to DoS attack without incurring a bonus cost to the attacker.*

PROOF: Suppose that the attacker has hash rate x. If $x \leq \frac{1}{k}$, then clearly the attack can succeed no more often than once out of every k blocks. Next, suppose that $x > \frac{1}{k}$. According to Lemma 5, in expectation, the attacker will mine fraction x of the k proofs in each block. If the attacker generates the 1OS, but also generates other proofs after it, then the others will incur a bonus cost. This is because they *would have* received bonus if the attacker released the block, but receive none when he conducts a DoS attack. However, with probability $\frac{1}{xk}$, the attacker will generate the 1OS *after* generating all his other proofs, in which case he would not have received any bonus anyhow. This is the only way that a block can be subject to DoS attack without incurring bonus cost. The attacker can successfully DoS attack no more often than the frequency with which he generates the 1OS, which occurs with probability x. It follows that the total fraction of blocks upon which a DoS attack can be inflicted without bonus cost is $\frac{x}{xk} = \frac{1}{k}$.

□

Increase in inter-block time. Even when the block is withheld by the attacker, eventually a proof lower than the attacker's lowest will be generated, whose owner will also be capable of mining the block. Even when the attacker controls a large fraction of the hash rate, the expected waiting time for this lower proof is a low-order multiple of the typical block time.

> **THEOREM 5:** *Assume that target t_k is tuned so that the expected block time is T seconds. When the attacker has up to 50% of the total hash rate, the expected time for honest miners to replace the 1OS for a block subject to DoS attack does not exceed $2T$.*

PROOF: Eq. 5 establishes that $r = vh/S$ where S is the size of the hash space, h is the expected number of hashes required to mine a block, and v and r are (respectively) the expected value and frequency with which the minimum hash is generated during interval I. According to Theorem 7, $X_1 \sim \texttt{Expon}(1/r)$ is a random variable representing the number of intervals I required to find a proof low enough to assemble a block. If I is one second, then blocks are expected after T intervals so that $E[X_1] = T$. Now let $X_1^* \sim \texttt{Expon}(1/r^*)$ be a random variable denoting the number of intervals required once the attacker removes his portion of hash rate. If the attacker holds 50% of the total hash rate, then $h^* = h/2$ and

$$E[X_1^*] = \frac{1}{r^*} = \frac{S}{v(h/2)} = 2\frac{S}{vh} = 2\frac{1}{r} = 2E[X_1] = 2T.$$

Therefore, in expectation, honest miners are capable of replacing a 1OS generated by the attacker in time equal to twice the target block inter-arrival time T.

□

IX. Limitations

Bobtail has limitations and disadvantages compared to existing PoW systems. Bobtail's reduction in variance decreases with k by $\frac{8k+4}{6(k^2+k)} = O\left(\frac{1}{k}\right)$, but comes at the cost of a larger block. Existing Bitcoin blocks consist of an 80 byte header and body that contains the transaction set. In Bobtail, the header $\mathcal{H}$ contains an additional 32-byte supporting proof s_1. The Bobtail body includes the transaction set $\mathcal{T}$ like Bitcoin, and additionally, for each value of $k > 1$, the size of proof package $\mathcal{P}$ is increased by one proof set consisting of four 32-byte values. Similarly, the coinbase reward $\mathcal{C}$ increases by a 32-byte address per k. The header must also be signed, which is another 32-byte value. Hence, Bobtail blocks are approximately $128(k-1) + 32k + 144$ bytes not including the size of $\mathcal{T}$; for example when $k = 40$, the header would be approximately 6.3KB. This overhead is small relative to the size of the block: 0.61% of Bitcoin's 1 MB blocks, and 0.02% of Bitcoin Cash's 32 MB blocks. Bobtail's increased header size is significant for light-clients (e.g. [1, 2]), which validate only block headers, but we note that these clients can discard headers once they have been validated.

The additional information contained in Bobtail blocks would add little additional overhead to block propagation because redundant information can be propagated using the Graphene protocol [38]. Graphene provides a compressed representation of a block's transactions, taking advantage of the fact that transactions are typically propagated and stored at peers ahead of the block. It can be applied to Bobtail advantageously in two ways. First, Bobtail can improve Graphene compression as follows. Whether the lowest proof set will ultimately become V_1 is unknown until a final block is assembled. However, the transactions of the lowest proof set at any given time can be prioritized for propagation by peers, ensuring better Graphene performance and faster block propagation. Second, the proof sets contained in $\mathcal{K}$ can be numerous, but like transactions, they are always propagated ahead of an assembled block. Graphene can be applied to propagating $\mathcal{K}$, where the proof sets take the place of transactions in the protocol.

Although we have investigated numerous potential attack vectors, our security analysis of Bobtail is not comprehensive. It is likely that techniques similar to those employed by Sapirshtein et al. [41], Gervais et al. [26], and Zhang et al. [44] are required to achieve this end. We note that because these works each use a Markov Decision Process (MDP) to model block creation only, their results do not apply to Bobtail. In other words, none of these works have a model that captures the complexity of Bobtail's proof creation and dissemination process. Indeed, we have communicated directly with the authors of [44], who have acknowledged that Bobtail was not modeled in that recent work. A post-publication revision of the aforementioned paper acknowledges this fact.[4]

X. Related Work

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

Foundations of PoW. A large number of papers have explored applications of proof-of-work (PoW). Dwork and Naor [19] first suggested PoW in 1992, applying it as a method to thwart spam email. A number of subsequent works similarly applied PoW to thwarting denial-of-service (DoS) attacks [4, 13, 15, 25, 27, 43]. Our approach can be adopted into many of these past works to improve computational variance. Jakobsson and Juels [29] and Jules and Brainerd [30] examine the security properties of PoW protocols, and base their theorems on the average

[4] https://www.esat.kuleuven.be/cosic/publications/article-3005.pdf

work required; our approach would provide stronger guarantees under their theorems since the variance is lower. Laurie and Clayton [32] examine the practical limitations in deploying PoW solutions in DoS scenarios. Douceur [18] noted in 2002 that proofs of work can mitigate Sybil attacks. Also in 2002, Back [5] applied PoW to cryptocurrencies. Back noted the high variance of computational PoW and regarded it as an open problem. Nakamoto's Bitcoin [35] built on these ideas.

Similar protocols. Bobtail is situated among a number of related PoW protocols introduced in recent years, beginning with the seminal Bitcoin whitepaper [35]. FruitChains [39] is the most similar work to ours but lacks many of the benefits of our approach. Like Bobtail, FruitChains thwarts withholding attacks that enable selfish mining, and it also reduces the variance of how often rewards are issued to miners. Unlike Bobtail, FruitChains offers no reduction in the variance of block times, and therefore does not mitigate doublespend attacks. Bitcoin-NG [22] also offers low-variance transaction announcements via PoW-based leader election. However, because inter-leader time variance would still follow an exponential distribution, unlike Bobtail, Bitcoin-NG does not reduce doublespend or selfish mining vulnerability. Furthermore, unlike our approach, Bitcoin-NG sets up the elected leader as a single point of failure and attack.

Leader election. In Bobtail, multiple miners collaborate to generate the PoW required to mine a block. They do this by submitting partial PoW, called proofs, to the network. Anyone with hardware capable of generating proofs can participate, yet only the miner who generates the lowest proof, the 1OS, can determine the set of transactions in a block. In this sense, the Bobtail mining process is a form of *leader election* whereby miners *propose* blocks as they produce proofs and the lowest of these proofs determines the leader whose proposal is accepted. This form of leader election is similar to many existing PoW systems including Bitcoin [35] and its derivatives except that, in the latter, each proof is independently sufficient to propose a block and appoint a leader. All of these approaches contrast with Bitcoin-NG [22], which first elects a leader who then proposes blocks *after* election until the next leader is elected. Finally, we note that leader election in Bobtail contrasts sharply with classic Byzantine Fault Tolerant (BFT) protocols [42]. Among many other differences, BFT restricts the leader election pool while Bobtail does not.

Broader impact. Our work is complementary to and improves upon most any work that combines PoW with BFT [11, 16, 31]. Some blockchains are not based on computational proof of work, and our solution does not apply to them. These include proof-of-storage [34], proof-of-stake [6, 7], and blockless [10] schemes. However, currently, almost all the wealth stored in cryptocurrencies is in computational PoW blockchains that our approach does apply to, including Bitcoin, Bitcoin Cash, Litecoin, Ethereum, and Ethereum Classic.

Reducing variance. Much more limited is the work related to reducing variance in random computational processes. In 2003, Abadi et al. [3] suggested memory-bound functions as a better foundation for avoiding the variance in CPU resources among users. Indeed, the ETHASH [20] PoW algorithm in Ethereum [21] adopted a PoW function that requires more memory than most ASICSs provide. 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 [14] proposes a PoW puzzle based on Merkle trees that requires an exact number of steps and therefore has no computational variance. Without variance, the same miner would always win, and therefore the method is unsuitable for blockchains.

XI. Conclusion

We have designed and characterized a novel method of low-variance blockchain mining called Bobtail. We have derived expressions for the expectation and variance of the Bobtail mining proof of work and its mining time for any value of k. Using these expressions, we have shown that Bobtail reduces variance in block inter-arrival time by a factor of $O(1/k)$, compared to using $k = 1$. We have also shown that forks are inadvertently created by Bobtail miners no more often than existing systems, and that dishonest miners receive significantly lower rewards due to minor protocol adjustments. Furthermore, we have demonstrated that low-variance mining significantly reduces the effectiveness of doublespend and selfish mining attacks, and that our design thwarts withholding and denial-of-service attacks. Finally we have introduced a policy for miner coordination in Bobtail that keeps network traffic to a minimum.

Acknowledgment

The authors would like to thank David Thibodeau for his careful review of this paper on multiple occasions as well as the many helpful discussions we had with him.

References

[1] "Electrum Bitcoin Wallet," https://electrum.org.

[2] "Neutrino Bitcoin Cash Wallet," https://neutrino.cash.

[3] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, "Moderately hard, memory-bound functions," *ACM Trans. Internet Technol.*, vol. 5, no. 2, pp. 299–327, May 2005. [Online]. Available: http://doi.acm.org/10.1145/1064340.1064341

[4] T. Aura, P. Nikander, and J. Leiwo, "Dos-resistant authentication with client puzzles," in *Revised Papers from the 8th International Workshop on Security Protocols*, 2001, pp. 170–177. [Online]. Available: http://dl.acm.org/citation.cfm?id=647218.720854

[5] A. Back, "Hashcash - Amortizable Publicly Auditable Cost-Functions," 2002. [Online]. Available: http://www.hashcash.org/papers/amortizable.pdf

[6] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 142–157.

[7] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake [Extended Abstract]," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[8] "Bitcoin (BTC)," https://github.com/bitcoin/bitcoin.

[9] Bitcoin Unlimited, "Bitcoin cash implementation," https://github.com/BitcoinUnlimited/BitcoinUnlimited.

[10] X. Boyen, C. Carr, and T. 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] V. Buterin and V. Griffith, "Casper the friendly finality gadget," https://arxiv.org/abs/1710.09437, Oct 2017.

[12] G. Casella and R. L. Berger, *Statistical inference*. Pacific Grove, CA: Brooks Cole, 2002. [Online]. Available: http://opac.inria.fr/record=b1134456

[13] L. Chen and W. Mao, "An auditable metering scheme for web advertisement applications," *Information Security*, pp. 475–485, 2001.

[14] F. Coelho, "An (Almost) Constant-Effort Solution- Verification Proof-of-Work Protocol Based on Merkle Trees," in *Progress in Cryptology – AFRICACRYPT*, June 2008, pp. 80–93.

[15] D. Dean and A. Stubblefield, "Using client puzzles to protect tls," in *Proc. Conference on USENIX Security Symposium*, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251327.1251328

[16] C. Decker, J. Seidel, and R. Wattenhofer, "Bitcoin Meets Strong Consistency," in *Proc. International Conference on Distributed Computing and Networking (ICDCN)*, January 2016.

[17] J. A. D. Donet, C. Pérez-Sola, and J. Herrera-Joancomartí, "The Bitcoin P2P Network," in *Proc. International conference on financial cryptography and data security*, 2014.

[18] J. Douceur, "The Sybil Attack," in *Proc. International Workshop on Peer-to-Peer Systems (IPTPS)*, Mar. 2002.

[19] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Proc. Annual Intl. Cryptology Conference*, 1992, pp. 139–147.

[20] ETHASH, https://github.com/ethereum/wiki/wiki/Ethash, Aug 3 2017.

[21] "Ethereum Homestead Documentation," http://ethdocs.org/en/latest/.

[22] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: A Scalable Blockchain Protocol," in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, 2016, pp. 45–59.

[23] I. Eyal and E. G. Sirer, "Majority is not Enough: Bitcoin Mining is Vulnerable," in *Proc. International conference on financial cryptography and data security*. Springer, 2014, pp. 436–454.

[24] "Fast Internet Bitcoin Relay Engine (FIBRE)," http://bitcoinfibre.org/.

[25] M. Franklin and D. Malkhi, "Auditable metering with lightweight security." in *Proc. Financial Cryptography*, 1997, pp. 151–160.

[26] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. ACM Conference on Computer and Communications Security*, 2016, pp. 3–16.

[27] B. Groza and B. Warinschi, "Cryptographic Puzzles and DoS Resilience, Revisited," *Designs, Codes and Cryptography*, vol. 73, no. 1, pp. 177–207, Oct. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10623-013-9816-5

[28] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse Attacks on Bitcoin's Peer-to-Peer Network," in *Proc. USENIX Security Symposium*, 2015, pp. 129–144.

[29] M. Jakobsson and A. Juels, "Proofs of Work and Bread Pudding Protocols," in *Proc. Conference on Secure Information Networks: Communications and Multimedia Security*, 1999, pp. 258–272. [Online]. Available: http://dl.acm.org/citation.cfm?id=647800.757199

[30] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks." in *Proc. Networks and Distributed Security Systems*, 1999, pp. 151–165.

[31] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proc. USENIX Security Symposium*, 2016, pp. 279–296.

[32] B. Laurie and R. Clayton, ""Proof-of-work" proves not to work; version 0.2," in *Proc. Workshop on Economics and Information Security*, 2004.

[33] Litecoin, https://litecoin.org/.

[34] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *Proc. IEEE Security and Privacy*, 2014, pp. 475–490.

[35] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf, May 2009.

[36] A. P. Ozisik, G. Andresen, G. Bissias, A. Houmansadr, and B. N. Levine, "Graphene: A New Protocol for Block Propagation Using Set Reconciliation," in *Proc. of International Workshop on Cryptocurrencies and Blockchain Technology (ESORICS Workshop)*, Sept 2017.

[37] A. P. Ozisik and B. N. Levine, "An Explanation of Nakamoto's Analysis of Double-spend Attacks," University of Massachusetts, Amherst, MA, Tech. Rep. arXiv:1701.03977, January 2017.

[38] A. P. Ozisik, B. N. Levine, G. Bissias, G. Andresen, G. Bissias, D. Tapp, and S. Katkuri, "Graphene: Efficient Interactive Set Reconciliation Applied to Blockchain Propagation," in *Proc. ACM SIGCOMM*, August 2019.

[39] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proc. ACM Symposium on Principles of Distributed Computing*, 2017, pp. 315–324.

[40] P. Rizun, "Subchains: A Technique to Scale Bitcoin and Improve the User Experience," *Ledger*, vol. 1, pp. 38–52, 2016.

[41] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal Selfish Mining Strategies in Bitcoin," in *Proc. Financial Cryptography and Data Security (See also https://arxiv.org/pdf/1507.06183.pdf)*, Feb 2016.

[42] M. Vukolić, " The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication," in *International Workshop on Open Problems in Network Security*, 2015.

[43] X. Wang and M. K. Reiter, "Defending against denial-of-service attacks with puzzle auctions," in *Proc. IEEE Symposium on Security and Privacy*, 2003, pp. 78–92.

[44] R. Zhang and B. Preneel, "Lay Down the Common Metrics: Evaluating Proof-of-Work Consensus Protocols' Security," in *IEEE Symposium on Security and Privacy*, 2019.

APPENDIX

In this appendix, we prove that V_i, the value of the ith order statistic, is gamma distributed (Theorem 6). We also prove that X_i, which is the number of hash intervals required for the ith order statistic to fall below target v, is gamma distributed (Theorem 7). Both theorems are applied in Section IV. We then derive the joint distribution of the ith and jth order statistics in Lemma 8, which is applied in Section IV-C.

A. Properties of Bobtail Order Statistics

We begin with a supporting lemma. Consider the distribution of H an arbitrary random variable chosen from the sequence of block hashes $H_1, \ldots, H_h$. We have $f_H(t;S) = 1/S$ and $F_H(t;S) = t/S$.

LEMMA 7: *The probability density function (PDF) of the ith order statistic, V_i, from h samples (i.e., hashes) is*

$$\begin{aligned} f_{V_i}(t;S,h) &= \tfrac{h!}{(i-1)!(h-i)!} f_H(t) \left(F_H(t)\right)^{i-1} \left(1-F_H(t)\right)^{h-i} \\ &= \tfrac{h!}{(i-1)!(h-i)!} \tfrac{1}{S} \left(\tfrac{t}{S}\right)^{i-1} \left(1-\tfrac{t}{S}\right)^{h-i}. \end{aligned} \tag{23}$$

The above result is well known; see for example, Casella and Berger [12].

When hash interval I corresponds to the desired block time, say 600 seconds for Bitcoin, there will be many hashes performed during the interval. So it is reasonable to consider how the distribution for V_i changes in the limit that h approaches infinity.

THEOREM 6: *In the limit that h approaches infinity, $V_i \sim \texttt{Gamma}(i, v)$, where v is the expected value of the minimum hash.*

PROOF: Define $g(t;i,v)$ to be the PDF of the gamma distribution with shape parameter i and scale parameter v. If the number of hashes approaches infinity, then so must the

size of the hash space, and yet S must always be larger than h. Therefore, we assume that $h = S/v$ for arbitrary parameter $v > 1$. Under this assumption we can equivalently consider the limit that S approaches infinity. We have

$$\begin{aligned} f_{V_i}(t;S,h) &= \lim_{h\to\infty} f_{V_i}(t;S,h) \\ &= \lim_{S\to\infty} \tfrac{(S/v)!}{(i-1)!\left(\frac{S}{v}-i\right)!}\tfrac{1}{S}\left(\tfrac{t}{S}\right)^{i-1}\left(1-\tfrac{t}{S}\right)^{\frac{S}{v}-i} \\ &= \lim_{S\to\infty} \tfrac{(S/v)!}{S^i(i-1)!\left(\frac{S}{v}-i\right)!}t^{i-1}\left(1-\tfrac{t}{S}\right)^{\frac{S}{v}-i} \\ &= \tfrac{t^{i-1}}{(i-1)!}\left[\lim_{S\to\infty}\tfrac{\left(\frac{S}{v}\right)\ldots\left(\frac{S}{v}-i+1\right)}{S^i}\right]\left[\lim_{S\to\infty}\left(1-\tfrac{t}{S}\right)^{\frac{S}{v}-i}\right] \\ &= \tfrac{t^{i-1}}{(i-1)!v^i}e^{\frac{-t}{v}} \\ &= g(t;i,v), \end{aligned} \tag{24}$$

The second-to-last step follows from the fact that

$$\lim_{S\to\infty}\frac{\left(\frac{S}{v}\right)\ldots\left(\frac{S}{v}-i+1\right)}{S^i} = \lim_{S\to\infty}\frac{\left(\frac{S}{v}\right)^i}{S^i} = \frac{1}{v^i}, \tag{25}$$

and the common limit

$$\lim_{S\to\infty}\left(1-\frac{t}{S}\right)^{\frac{S}{v}} = e^{\frac{-t}{v}}, \tag{26}$$

which implies that

$$\begin{aligned} \lim_{S\to\infty}\left(1-\tfrac{t}{S}\right)^{\frac{S}{v}-i} &= \left[\lim_{S\to\infty}\left(1-\tfrac{t}{S}\right)^{-i}\right]\left[\lim_{S\to\infty}\left(1-\tfrac{t}{S}\right)^{\frac{S}{v}}\right] \\ &= 1\cdot e^{\frac{-t}{v}}. \end{aligned} \tag{27}$$

When $i = 1$, $V_1 \sim \texttt{Gamma}(t;1,v) = \texttt{Exponential}(t;v)$. And since the expected value of an exponential random variable is equal to the value of its scale parameter, we can see that v is simply the expected value of the minimum hash.

□

Next, define X_i as the number of intervals required for V_i to fall below v, and consider the PDF of X_i, $f_{X_i}(x;S,v)$. After x hash intervals, let E, L, and G be, respectively, the events that the ith order statistic is equal to v, the order statistics below i are less than v, and the order statistics above i are greater than v. Furthermore, let $\mathcal{O}$ be the set of all divisions of $H_1,\ldots,H_h$ into distinct sets $\{H \mid H = V_i\}$, $\{H \mid H < V_i\}$, and $\{H \mid H > V_i\}$. We have

$$\begin{aligned} f_{X_i}(x;S,v) &= \sum_{o\in\mathcal{O}} P[E(x),L(x),G(x) \mid o] \\ &= \tbinom{hx}{i-1}(hx-i+1)\cdot\Pr[E(x)|o]\cdot\Pr[L(x)|o]\cdot\Pr[G(x)|o] \\ &= \tfrac{(hx)!}{(i-1)!(hx-i)!}\cdot\Pr[E(x)|o]\cdot\Pr[L(x)|o]\cdot\Pr[G(x)|o] \\ &= \tfrac{(hx)!}{(i-1)!(hx-i)!}\tfrac{1}{S}\left(\tfrac{v}{S}\right)^{i-1}\left(1-\tfrac{v}{S}\right)^{hx-i} \end{aligned} \tag{28}$$

Assuming that I is large, it again makes sense to consider the limit as h approaches infinity.

THEOREM 7: *In the limit that h approaches infinity, $X_i \sim \texttt{Gamma}(i,1/r)$ where r is the expected rate, in units of I, at which V_i falls below v.*

PROOF: The probability that any given hash *succeeds*, i.e. falls below v, is given by $p = \frac{v}{S}$. Again, we would like to consider the limit as h approaches infinity. But in doing so, we must ensure that p remains constant. In other words, the probability of hash success must diminish as h increases. So there must exist some constant r such that $\frac{r}{h} = p = \frac{v}{S}$. It follows that

$$f_{X_i}(x;S,v) = \tfrac{(hx)!}{(i-1)!(hx-i)!}\tfrac{r}{h}\left(\tfrac{r}{h}\right)^{i-1}\left(1-\tfrac{r}{h}\right)^{hx-i} \tag{29}$$

Arguing in similar fashion as for V_i, we find that

$$\lim_{h\to\infty} f_{X_i}(x;S,v) = g(x;i,1/r).$$

Thus, $E[X_i] = 1/r$, which implies that r should be interpreted as the expected rate at which V_i falls below v during a single interval I.

□

B. Joint Distribution

Here we derive the limiting joint distribution of the ith and jth order statistics V_i and V_j, which are applied in Section IV-C to derive the variance of W_k.

LEMMA 8: *In the limit that h approaches infinity, the joint distribution of the ith and jth order statistics of uniform random samples $H_1,\ldots,H_h$ is given by*

$$f_{V_i,V_j}(t_i,t_j;v) = g(t_i;i,v)g(t_j-t_i;j-i,v). \tag{30}$$

where v is the expected value of the minimum hash.

PROOF: It is well known[5] that the joint distribution of the ith and jth order statistics, out of h total samples, is given by

$$\begin{aligned} f_{V_i,V_j}(t_i,t_j;v) =& \tfrac{h!}{(i-1)!(j-1-i)!(h-j)!}f_H(t_i)f_H(t_j)[F_H(t_i)]^{i-1} \\ &\times[F_H(t_j)-F_H(t_i)]^{j-1-i}[1-F_H(t_j)]^{n-j}. \end{aligned} \tag{31}$$

Thus, we have

$$\begin{aligned} & f_{V_i,V_j}(t_i,t_j;S,v) \\ &= \tfrac{t_i^{i-1}(t_j-t_i)^{j-1-i}}{S^j}\tfrac{h!}{(i-1)!(j-1-i)!(h-j)!}\left(1-\tfrac{t_j}{S}\right)^{h-j} \\ &= \tfrac{\frac{S}{v}\ldots\left(\frac{S}{v}-j+1\right)}{S^j}\tfrac{t_i^{i-1}(t_j-t_i)^{j-1-i}}{(i-1)!(j-1-i)!}\left(1-\tfrac{t_j}{S}\right)^{\frac{S}{v}-j}. \end{aligned} \tag{32}$$

Finally, assuming $j > i$, and reasoning in the limit as $S\to\infty$ in the same manner as in Theorem 6,

$$\begin{aligned} f_{V_i,V_j}(t_i,t_j;v) &= \lim_{S\to\infty} f_{V_i,V_j}(t_i,t_j;S,v) \\ &= \tfrac{1}{v^j}\tfrac{t_i^{i-1}(t_j-t_i)^{j-1-i}}{(i-1)!(j-1-i)!}e^{-\frac{t_j}{v}} \\ &= \tfrac{t_i^{i-1}}{v^i(i-1)!}e^{-\frac{t_i}{v}}\tfrac{(t_j-t_i)^{j-1-i}}{v^{j-i}(j-1-i)!}e^{-\frac{t_j-t_i}{v}} \\ &= g(t_i;i,v)g(t_j-t_i;j-i,v). \end{aligned} \tag{33}$$

□

[5] See Casella and Berger [12], Theorem 5.4.6.