Estimation of Miner Hash Rates and Consensus on Blockchains

A. Pinar Ozisik George Bissias Brian N. Levine
College of Information and Computer Sciences, UMass Amherst

ABSTRACT

We make several contributions that quantify the real-time hash rate and therefore the consensus of a blockchain. We show that by using only the hash value of blocks, we can estimate and measure the hash rate of all miners or individual miners, with quantifiable accuracy. We apply our techniques to the Ethereum and Bitcoin blockchains; our solution applies to any proof-of-work-based blockchain that relies on a numeric target *for the validation of blocks. We also show that if miners regularly broadcast status reports of their partial proof-of-work, the hash rate estimates are significantly more accurate at a cost of slightly higher bandwidth. Whether using only the blockchain, or the additional information in status reports, merchants can use our techniques to quantify in real-time the threat of double-spend attacks.*

1 INTRODUCTION

Blockchains, such as Bitcoin [27] and Ethereum [17], are among the most successful peer-to-peer (p2p) systems on the Internet. Bitcoin has been adopted more widely for e-commerce than any previous digital currency. Ethereum is also quickly gaining prominence as a blockchain that runs Turing-complete distributed applications. Ethereum applications include sub-currencies [12], distributed autonomous organizations with share holders [13], prediction markets [2], and games [18]; and it currently has around half the market capitalization of Bitcoin.

There are many advantages to blockchains, including decentralized operation. One of the primary limitations of blockchains is that the status of transactions, blocks, and branches is not immutable — a fork of blocks supported by greater proof of work (POW) can emerge at any time. Fortunately, the probability that there will be a change in consensus regarding which fork to build on decreases exponentially as the blockchain grows [27]. Specifically, consider an attacker, with a fraction of all mining power $0 < q < \frac{1}{2}$, that wishes to create a fork starting from a point z blocks deep on the main chain. Let the mining power building on the main chain be p, such that $q+p = 1$. The probability [20] that the attacker will eventually produce a fork with more POW is $(\frac{q}{p})^z$. Several models of blockchain security are largely based on the relative values of p, q, and z; see [19, 21].

For merchants who are vulnerable to attack, applying these models to a blockchain at a particular moment in time is not possible without assigning an accurate value to p or q. This is a challenging task because miners do not publish their hash rates in real-time in a format that is verifiable by third parties. Because no such information is available, merchants selling goods via Bitcoin and Ethereum have no guidance for when transactions are sufficiently safe from attacks. For example, the core Bitcoin client displays a transaction as confirmed once it is exactly 6 blocks deep in the blockchain [7], an overly simplistic choice that is used regardless of any other factors or conditions. Advice from the community is appropriately vague on when a block is confirmed [8].

To further complicate this problem, miners' hash rates can and do change dynamically (e.g., due to diurnal electricity rates), even though the POW algorithms are engineered to change their work targets glacially. In other words, merchants who seek a high probability that a transaction's status will not change should be concerned with not only block depth, but the instantaneous hash rate of the network. Meanwhile, honest consumers can grow impatient waiting for a large number of blocks to be appended to the blockchain in order to get their merchandise, especially when traditional bank-based commerce can take seconds.

Contributions. In this paper, we make several contributions based on novel approaches to estimating the real-time hash rate of miners and, therefore, the real-time consensus of a blockchain. We apply our techniques to both the actual Ethereum and Bitcoin blockchains, and in fact, they can be applied to any proof-of-work-based blockchain that relies on a numeric *target* for the validation of blocks [26, 31]. Our contributions are summarized as follows.

- First, we design a method of accurate hash rate estimation based on compact *status reports* issued by miners. The reports add no computational load to miners, and are stored neither on the blockchain nor at peers that receive them. They can be broadcast off-network, for example via RSS or Twitter. Just like block headers, reports are verifiable as authentic POW by third parties.
- Second, we show hash rates can be estimated from *only blocks* that are published to the blockchain. This approach requires *no cooperation from the miners*, but is less accurate than status reports.

- We evaluate the accuracy of the two approaches using a synthetic blockchain as ground truth. Additionally, we derive Chernoff bounds for the accuracy of our estimates from status reports. We show that both methods can be used together in support of incremental adoption by mining pools. Our blockchain-only method incurs no network costs. Status reports incur extra traffic depending on their rate. For example, if all active miners issue about 10 reports per block, the cost is about 0.03 KBps for Bitcoin and 6.6 KBps for Ethereum; note that none of the report data needs to be archived. The accuracy of both methods is tunable.
- We apply our estimates to calculating the probability of a double-spend attack against blocks as they garner descendants. We show that using status reports, 99% of blocks have a risk of 0.1% by depth of 13 for a worst-case estimate. We show that without status reports, merchants should wait much longer. We characterize the historical performance of Ethereum and Bitcoin, and show half of blocks require a depth of at least 40 before an attacker's success is below 0.1% for a worst case estimate. We also consider attacks against our approach. A public demo is at http://cs.umass.edu/~brian/blockchain.html that provides a quantified, realtime estimate of the security of blocks.

We begin with a background on blockchains and conclude with a discussion of limitations and related work.

2 BACKGROUND

The *blockchain* concept was introduced by Nakamoto [27] as a method of probabilistic distributed consensus [35]. Originally designed to be the backbone of the Bitcoin distributed cryptographic currency, blockchains have since been applied to a variety of scenarios. Bitcoin itself includes a scripting language that supports a limited set of custom smart contracts. Ethereum[17] is a blockchain-based distributed system that supports Turing-complete software and includes a global data store. Transactions in Ethereum represent the transfer of money or data among programs, allowing for a richer space of distributed applications. Other blockchains have been proposed and implemented to support anonymous transactions [31], hedge funds [13], medical records [3], and alternate currencies [12, 26]. Below we describe the aspects of Bitcoin and Ethereum that are relevant to us. Tschorsch et al. [34], Bonneau et al. [9], and Croman et al. [11] offer summaries of broader blockchain research issues.

2.1 Bitcoin

Accounts. Bitcoin users store bitcoins in accounts called *addresses*, which are created with an empty balance simply by generating a public/private key pair. The transfer of coin between addresses, via *transactions*, is recorded on a public ledger called a *blockchain*. Transactions are authenticated via a private key signature, and the balance of each account can be derived from the blockchain.

Adding to the blockchain. To be added to the blockchain, transactions are broadcast by users on Bitcoin's p2p network. A set of *miners* on the p2p network verify that each transaction is signed correctly, does not conflict with a previous transaction, does not move more coin than is contained in the address, and other functions. Each miner independently agglomerates a set of valid transactions into a candidate *block* and attempts to solve a predefined cryptographic puzzle as POW, which involves data from the candidate block and a specific *prior block*. The new transactions are only valid if they do not conflict with the set of transactions that are contained in all blocks that are direct ancestors.

The first miner to solve the problem broadcasts his solution to the network, and by virtue of the solution, is able to add the block to the ever-growing blockchain as a child of the prior block. The miners then start over, using the newly appended blockchain and the set of remaining transactions. The miners' incentive for *discovering* a new block is a reward of coins, called the *coinbase*, consisting of a predetermined *block reward* (currently worth 12.5 Bitcoins) and fees from transactions included in the block.

In Bitcoin, the POW computation is dynamically calibrated to take approximately ten minutes per block. When transactions appear in a block, they are *confirmed*, and each subsequent block provides additional confirmation. To announce a new block, a miner lists all transactions contained in the new block along with a header that contains an easily-verifiable POW solution. When a node or miner receives a new block, he validates each transaction in the block and the POW.

Notably, if there is a fork on the chain, honest miners always select the prior block as the last block containing the largest amount of POW. However, due to propagation delays in the network, miners can receive competing (but valid) block announcements, which bifurcates the chain, until one of the two forks is appended to first.

Block Headers. Any entity can elect to be a miner for Bitcoin, and there is no centralized party from whom to seek approval for mining. If all miners were to simply vote on which block should be appended to the main chain, then the mining process would fall vulnerable to a Sybil attack [14]. The POW puzzle addresses this problem by performing a kind of decentralized leader election: the miner that solves the puzzle can decide which block to append to the chain.

Proof-of-Work. Bitcoin uses a simple POW algorithm based on cryptographic hashing, proposed earlier by Douceur [14]. Specifically, miners apply a 256-bit cryptographic hash algorithm [23] to an 80-byte *block header*, and the puzzle is solved

if the resulting value is less than a known *target*, $0 < t < 2^{256}$. The header in Bitcoin consists of the Merkle root of the set of transactions, a timestamp, the target (stored as $2^{224}/t$), a *nonce*, and the hash of the prior block's header. If the hash is not less than the target, then a new nonce is selected to generate a new hash (the Merkle root can be adjusted as well). This process repeats until some miner finds a solution.

Each time a nonce is selected and the block header is hashed, the miner is sampling a value from a discrete uniform distribution with range $[0, 2^{256} - 1]$. The probability of solving the POW and discovering a block is the cumulative probability of selecting a value from $[0, t]$, which is $t/2^{256}$. Hence, in expectation, the number of samples needed to discover a block is $2^{256}/t$. Bitcoin adjusts the target so that on average it takes about 600 seconds to find a block. Typically, the target is described for convenience as a *difficulty*, defined to be $D = 2^{224}/t$. Bitcoin's difficulty is set once every two weeks.

2.2 Ethereum

Ethereum operates very similarly to Bitcoin. The following differences are relevant to the context of this paper. Ethereum miners solve a POW problem that is more complicated than Bitcoin in an attempt to disadvantage miners with custom ASICs. However, in the end, a miner still compares a hash value to the target. Specifically, the number of values in the block header is larger, resulting in a 508-byte header. It's not the hash of the header that is compared against the target, but the hash resulting from an Ethereum-specific algorithm called ETHASH [16], for which the hash of the block header is the primary input. In the end, the POW hash value is a sample from a discrete uniform distribution with range $[0, 2^{256} - 1]$, and the probability of block discovery is $t/2^{256}$.

A major difference of Ethereum is that the target is set such that the expected time between blocks is 15 seconds. This setting results in quicker confirmation times, but as a result, the probability that two miners announce blocks within the propagation time of a block announcement is much higher. Therefore, there are many abandoned forks in the chain. Ethereum uses a modified version of the GHOST [32] protocol for selecting the main fork of the blockchain: the main chain follows the block at each level with the most POW on its subtree. These differences do not affect the application of our algorithms; in fact, the presence of ommers is additional data which improves our estimates.

2.3 Double-spend Attacks

The fundamental blockchain double-spend attack [27] operates as follows. An attacker offers a transaction to a merchant in exchange for goods. The transaction appears on the blockchain in block B_1, with block B_0 denoting the prior block. The merchant releases goods purchased by the transaction to the attacker only after blocks $B_2, \ldots, B_z$ follow. Honest miners, with power $0 < p < 1$ add these $z - 1$ new blocks. The attacker, with mining power $q = 1 - p$ races to add a distinct sequence of blocks of length at least $z + 1$ that forks from B_0. Nakamoto derived the probability of the attacker's success of creating a longer fork, given infinite time. Nakamoto assumes that the miner's power is constant and that she never gives up on the attack. The attacker succeeds by producing any chain of length $z + 1$ or longer, but cannot announce the chain before the honest miners produce B_z since the merchant will not release purchased goods until then. This probability, where $\lambda = \frac{zq}{p}$, is

$$\mathcal{D}(q, z) = \begin{cases} 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (\frac{q}{p})^{z-k}\right) & \text{, if } q < \frac{1}{2} \\ 1 & \text{, if } q \geq \frac{1}{2} \end{cases} \quad (1)$$

When $z = 6$, an attacker with $q < 0.127$ has a less than 0.001 probability of success. Based on that value, the core bitcoind client displays transactions as confirmed once they appear in a block that is 6 deep in the chain.

3 PROBLEM STATEMENT

We seek to quantify the hash rate of miners in real-time, thereby quantifying the consensus of a blockchain towards its blocks and the transactions they hold. Our goal is to increase transparency so that merchants can judge their vulnerability to a double-spend attack; in short, we wish to answer the question, *what is the risk of releasing goods to a consumer for a transaction that is z block deep given the current hash rate of the network?*

Specifically, in Section 4, we propose that miners periodically issue status reports that are block headers with partial POW. Each report is exactly a block header except the nonce corresponds to the smallest hash value the miner found during the last σ seconds. We then answer these questions:

> *1. Given a set of status reports produced by a miner during a window of time, how many hashes (samples) were taken to produce the reports?*

We also quantify the network bandwidth costs for applying status reports to Bitcoin and Ethereum.

In Section 5, we ask how we can accomplish the task of estimating hash rate without status reports.

> *2. Given the set of valid blocks that were added to the blockchain during a window of time, how many hashes (samples) were taken by all miners to produce the blocks?*

We also ask how this approach can be combined with status reports for incremental deployment. In Section 6, we characterize the accuracy of our two estimators.

In Section 7, we apply our estimators to decide when to consider a transaction secure from double-spend attacks. Let block B_1 contain a transaction of interest, and let block B_0

Variable	Description
$\hat{h}$	estimation of network hash rate
$\hat{h}_m$	estimation of an individual miner's hash rate
σ	duration of time spent mining during interval I
θ_m	number of hashes performed *per miner* per status report
θ_σ	number of *network-wide* hashes performed during time duration σ
θ	number of hashes performed, i.e. shorthand for θ_m or θ_σ
S	the size of the hash space, i.e. $S = 2^{256} - 1$
V	for an arbitrary miner, a random variable representing the first order statistic (minimum hash value) after hashing θ times, where $V_i \sim Expon(\beta)$
β	exponential survival parameter of V_i
$\mathbf{V} = V_1, \ldots, V_n$	random sample of n first order statistics
$\overline{V}$	the *observed* sample mean of $\mathbf{V}$
$\hat{\beta}$	estimation of β, using status reports
$\hat{\theta}$	*estimated* θ from the sample population
Y	random variable equal to 0 if no block is produced during interval I; else the hash of the block. Note that Y_i is a function of V_i.
$\mathbf{Y} = Y_1, \ldots, Y_n$	random sample of n consecutive intervals, where a block is observed or not
$\overline{Y}$	the *observed* sample mean of $\mathbf{Y}$
$E[\mathbf{Y}]_\beta$	the expected value of $\mathbf{Y}$ parameterized by β
$\tilde{\beta}$	estimation of β, using only the blockchain

Figure 1: List of variables presented in Sections 4 and 5.

be its parent. Let B_i be the ith descendant from B_0 via B_1. We then estimate the hash rate of the network in a window prior to B_0, and the hash rate of the network in a window from B_0 to B_i, and we ask:

> *3. Assuming the difference in hash rate is being applied to a double-spend attack of the transaction based on a fork from B_0, what is the probability the attacker will succeed?*

We apply our algorithm to the real Bitcoin and Ethereum blockchains to characterize the typical delay required to ensure that the risk of a double-spend is suitably low.

4 STATUS REPORTS

In this section, we propose that active mining pools issue *status reports* periodically, which allow third parties to estimate their hash rate and to learn which specific block they are building on top of. Each report is exactly a block header except that the POW does not satisfy the current target. Instead, the minimum hash value in the report represents the hash found since the last block broadcast on the chain or their last report. To be clear, each status report does not directly report the minimum hash value; instead, reports are of the input values to the POW algorithm.

Below, we present and evaluate the accuracy of a method for estimating the hash rate of each miner using status reports. In the next section, we present a technique to make estimates from only blocks on the blockchain.

4.1 Hash Rate Estimation

As described in Section 2, the result of Bitcoin's and Ethereum's POW algorithms is a sample taken randomly from a discrete uniform distribution that ranges between $[0, 2^b - 1]$, where $b = 256$. The winning miner is the one that first produces a hash smaller than the target.

If a miner m periodically announces the smallest value he has discovered, we can estimate the hash rate, $\hat{h}_m$, he is lending toward finding a successor to a given block. In the next section, we detail how to accurately estimate the hash rate of cooperative miners using status reports.

Continuous model. In order to estimate a miner's hash rate, first we present a continuous model to describe the distribution of the smallest hash value he reports. Let I be an interval of σ seconds during which a miner attempts to mine a block. During that interval, the miner hashes the block θ_m times generating a sequence of hash values $\mathbf{Z} = Z_1, \ldots, Z_{\theta_m}$. Although we know σ, we do not know θ_m *a priori*. At the end of the interval, the miner sends status report $V = Z_{(1)}$, which denotes the lowest hash value achieved on the block during I. In other words, V is the random variable representing the first order statistic after hashing the block θ_m times. Our goal is to determine θ_m from a sample of first order statistics.

Let $S = 2^{256} - 1$ be the size of the hash space. The probability that any V is greater than some $v \in [0,\ S]$ is equal to $1 - v/S$. Thus, the probability that V is always greater than v across θ_m independent trials computed during I is given by $(1 - v/S)^{\theta_m}$. Now define $F_V(v \mid S)$ to be the CDF of V. It follows that

$$F_V(v \mid S) = 1 - \left(1 - \frac{v}{S}\right)^{\theta_m}. \tag{2}$$

Parameter θ_m can be expressed as a function of S:

$$\theta_m = S/\beta. \tag{3}$$

Consider how $F_V(v \mid S)$ changes as S increases. A common limit shows that

$$\lim_{S\to\infty} F_V(v \mid S) = 1 - e^{\frac{-v}{\beta}}. \tag{4}$$

Thus, $V \sim \texttt{Expon}(\beta)$, where β is the exponential *survival parameter*. Since the mean of the exponential distribution is equal to the survival parameter, we can see that the expected value of V is $\beta = S/\theta_m$. We next show how status reports spanning multiple intervals can be used to derive a robust estimator of θ_m, and ultimately $\hat{h}_m$.

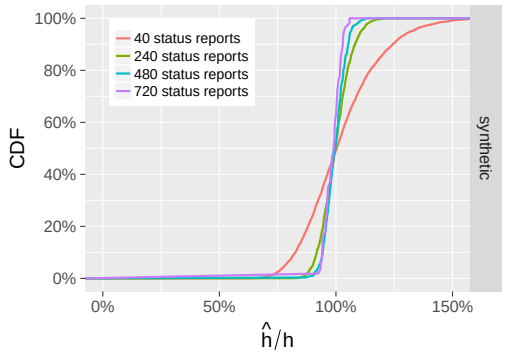

Figure 2: A CDF of the percentage of the estimated hash rate to the real hash rate, as the number of status reports increase. 100% on the x-axis denotes the line along which the estimated hash rate and real hash rate are equivalent.

Estimator for hash rate. Let $\mathbf{V} = V_1, \ldots, V_n$ with $V_i \sim \text{Expon}(\beta)$ be a random sample representing n status reports, each sent σ seconds apart by a miner. Because each V_i is $\text{Expon}(\beta)$ and i.i.d., the maximum likelihood estimator of β is equal to the sample mean, $\overline{V}$. Furthermore, the sample mean is an unbiased estimator of the population mean; so $\overline{V}$ is an unbiased estimator of β.

Since $\beta = S/\theta_m$, it follows then that a reasonable estimator for θ_m is given by

$$\hat{\theta}_m = \frac{S}{\overline{V}}. \tag{5}$$

Finally, the miner's hash rate $\hat{h}_m$ can be estimated as

$$\hat{h}_m = \frac{\hat{\theta}_m}{\sigma} = \frac{S}{\overline{V}\sigma}. \tag{6}$$

Empirical evaluation of accuracy. We implemented and evaluated this technique against a simulated miner. The miner sampled from a discrete uniform distribution at a fixed rate of h and issued status reports regularly. Our simulated recipient estimated the hash rate, $\hat{h}$, using Equation 6 given 40, 240, 480, or 720 reports. We measure the accuracy as $\hat{h}/h$ and show the result of tens of thousands of trials in Figure 2 as a CDF of the accuracy. A perfect estimate is at 100%. As the number of status reports increases, a greater fraction of our estimates are aligned with the real hash rate. More reports are required for high accuracy; however, when we apply this technique to a blockchain, we are interested in a window of blocks rather than a single block. Hence, the count of available status reports adds up quickly.

5 BLOCKCHAIN-ONLY ESTIMATION

In this section, we describe a blockchain-only method of estimating miner hash rates. For this estimator, we treat the entire network as a single miner and a block as a status report that can only be observed at certain intervals. Although this approach has no additional network costs and does not require cooperation from miners, it is less accurate than status reports. We then extend the technique to allow for hash rate estimation of an individual or a subset of miners. As we show, this extension allows for the incremental deployment of status reports.

5.1 Network Hash Rate Estimation

We would like to estimate the network hash rate, $\hat{h}$, using only the *observed* POW hash values that are represented by mined blocks. A critical step in this process is estimating the number of hashes that were performed network-wide using these observed values. Conceptually, we are treating the entire network as a single miner from whom we receive reports only when a block is mined.

Consider a window of time during which at least one block is announced. We segment time into σ-second intervals: $\mathcal{I} = I_1, \ldots, I_n$, and let $\mathbf{V} = V_1, \ldots, V_n$ be the minimum hash value achieved across the entire network for each interval. Note that a valid V_i is produced during every interval I_i, even though it is not broadcast to the network unless it is below the target. This notation is consistent with our definitions from Section 4.1.

Suppose that blocks were mined at the end of intervals $\mathcal{I}_B \subseteq \mathcal{I}$ so that the observed hash values are given by the set $\mathbf{O} = \{V_i \mid I_i \in \mathcal{I}_B\}$. In practice, a block could have been mined at any point during an interval in $\mathcal{I}_B$, but the distinction becomes less important as $\sigma \to 0$. Finally, assume that, network-wide, θ_σ hashes were performed during each interval. Our goal is to determine an estimator of θ_σ.

Estimator for β, the expected minimum POW hash. In Section 4.1, we showed that $V_i \sim \text{Expon}(\beta)$ as $S \to \infty$ and argued that $\overline{V}$ is a good estimator of β. But that approach does not work here since we are missing most of the values V_i. Consider instead a new sequence of random variables $\mathbf{Y} = Y_1, \ldots, Y_n$ defined as a function of $\mathbf{V}$:

$$Y_i = \mathbf{1}_{V_i \le t}(V_i) V_i = \begin{cases} V_i, & V_i \le t \\ 0, & \text{otherwise} \end{cases}, \tag{7}$$

where t is the target and $\mathbf{1}_C(x)$ is the indicator function equal to 1 when $x \in C$ and 0 otherwise. Since each Y_i is a function of V_i and $V_i \sim \text{Expon}(\beta)$, it is straightforward to derive the expected value of any given Y_i:

$$E[Y_i]_\beta = \frac{1}{\beta}\int_0^t v e^{-v/\beta} dv \tag{8}$$

$$= \beta - \beta e^{-t/\beta}\left(\frac{t}{\beta} + 1\right) \tag{9}$$

The sample mean $\overline{Y}$ is simpler to determine. It is the sum of the observed hash values O_i divided by the number of intervals:

$$\overline{Y} = \frac{\sum_i O_i}{|\mathcal{I}|}. \tag{10}$$

Thus, we can derive the *method of moments* (MoM) [10] estimator $\tilde{\beta}$ by equating $E[Y_i]_\beta$ and $\overline{Y}$:

$$\tilde{\beta} - \tilde{\beta}e^{-t/\tilde{\beta}}\left(\frac{t}{\tilde{\beta}} + 1\right) = \overline{Y}. \tag{11}$$

Unfortunately, it is difficult to solve Equation 11 for $\tilde{\beta}$ analytically. Moreover, the implicitly defined $\tilde{\beta}$ is not actually a function because $E[Y_i]_{\tilde{\beta}}$ not a one-to-one function of $\tilde{\beta}$. And so we note that

$$\frac{\partial^2 E[Y_i]_{\tilde{\beta}}}{\partial \tilde{\beta}^2} = \frac{t^2}{b^4}e^{-t/\tilde{\beta}}(\tilde{\beta} - t) \tag{12}$$

with roots at $t = 0$ and $t = \tilde{\beta}$. Because we assume that the target t is always positive, this means that $E[Y_i]_{\tilde{\beta}}$ has a single inflection point at $t = \tilde{\beta}$. In light of the fact that $E[Y_i]_{\tilde{\beta}}$ is monotonic for values of $\tilde{\beta}$ on either side of t, it is also possible to verify that $E[Y_i]_{\tilde{\beta}}$ is strictly increasing for $\tilde{\beta} < t$ and decreasing for $\tilde{\beta} > t$. Thus, Equation 11 implicitly defines two different functions for $\tilde{\beta}$: $\tilde{\beta}(\overline{Y})^-$ when $\tilde{\beta} < t$ and $\tilde{\beta}(\overline{Y})^+$ when $\tilde{\beta} > t$.

Because both sides of the function $\tilde{\beta}(\overline{Y})$ are monotonic, it is straightforward to solve each using binary search. Algorithm 1 defines a procedure for finding $\tilde{\beta}(\overline{Y})^+$, and $\tilde{\beta}(\overline{Y})^-$ can be found in a similar fashion.

A drawback of the estimator is that it forces the practitioner to guess if $\tilde{\beta}$ is greater or less than t in order to find its actual value. We find that in practice this is rarely an issue. Recall that β is the expected minimum hash value for a σ-second time interval, while t is the target minimum hash for the much longer block creation interval. Thus, unless the mining pool under consideration is mining at many times the current difficulty, we can be quite certain that $\beta \gg t$. Accordingly, it is also likely that $\tilde{\beta}$ will also be significantly larger than t, which makes $\tilde{\beta}(\overline{Y})^+$ the correct branch to use.

Estimating hash rate from $\tilde{\beta}$. Because $\beta = S/\theta_\sigma$, a good estimator for θ_σ is $\hat{\theta}_\sigma = S/\tilde{\beta}$. This estimates the number of hashes per σ-second interval, which we can use to estimate the hash rate of the entire block creation interval as

$$\hat{h} = \frac{S}{\tilde{\beta}\sigma}. \tag{13}$$

Empirical evaluation of accuracy. We used the technique described above to estimate the number of network-wide hashes on a synthetic blockchain, using a window of 100, 500, 1000, or 5000 seconds, as shown in Figure 3. As the window length increases, we incorporate data from additional blocks, allowing for a greater portion of estimates to be equal to the the real hash rate.

We also used our technique to estimate the number of hashes per block at regular intervals in the Bitcoin and Ethereum networks. As a comparison with a naive approach, we calculated the expected number of hashes based on the difficulty, D, of blocks and the current target, t. For Bitcoin, this value is $E[\hat{h}] = \frac{2^{256}}{t \cdot 600} = \frac{2^{32}D}{600}$; it is $E[\hat{h}] = 2^{32}D/15$ for Ethereum. The accuracy of our approach is apparent in Figures 4 and 5.

Algorithm 1 Find $\tilde{\beta}$

1: Let $\tau > 0$ be some small tolerance parameter
2: Set $L = 0$, $H = S$, and choose $\theta_\sigma = S/2$
3: Set $\tilde{\beta} = S/\theta_\sigma$
4: Calculate $E[Y]_{\tilde{\beta}}$ by substituting beta tilde as beta into Equation 9
5: **if** $E[Y]_{\tilde{\beta}} - \tau > \overline{Y}$ **then**
6: Set $L = \theta_\sigma$, $H = H$, and $\theta_\sigma = L + (H - L)/2$
7: Goto 3
8: **else if** $E[Y]_{\tilde{\beta}} + \tau < \overline{Y}$ **then**
9: Set $L = L$, $H = \theta_\sigma$, and $\theta_\sigma = L + (H - L)/2$
10: Goto 3
11: **else**
12: **return** $\tilde{\beta}$
13: **end if**

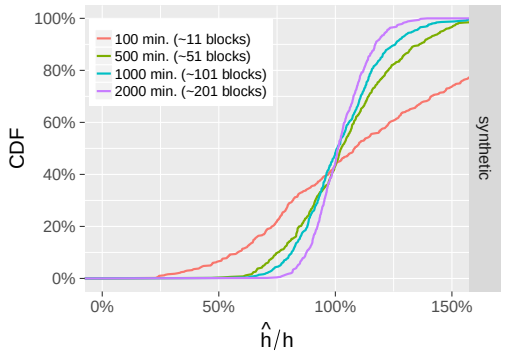

Figure 3: A CDF of network hash rate using method of moments on a synthetic blockchain.

5.2 Hash Rate Estimation of a Subset of Miners

We can estimate the hash rate of a single miner or a subset of miners with a simple extension of our technique from Section 5.1. Specifically, for a given miner m, we vary Eq. 5 such that the random variable $\mathbf{Y} = Y_1, \ldots, Y_n$ is defined as a function of $\mathbf{V}$ as follows:

$$Y_i = \mathbf{1}_{V_i \le t}(V_i)V_i = \begin{cases} V_i & V_i \le t \text{ and } \mathbf{1}_m(V_i) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where $\mathbf{1}_m(x)$ is the indicator function equal to 1 when x is mined by m and 0 otherwise. Therefore, we construct $\mathbf{Y}$ only using the blocks issued by a given miner. The same approach also works for a subset of all miners.

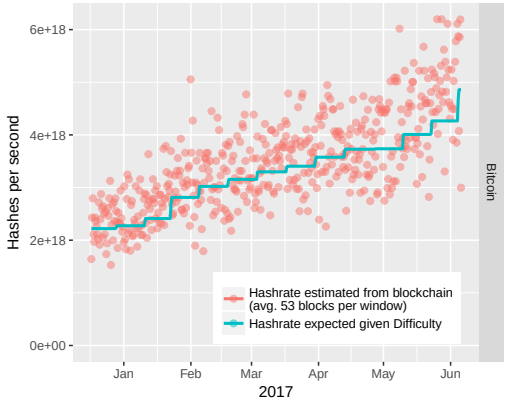

Figure 4: Network-wide hash rate, computed using MoM, of the Bitcoin network compared to the hash rate calculated based on network difficulty and target value. Points represent a window of 500 minutes (about 50 blocks); for clarity, only a sample of all possible windows is plotted.

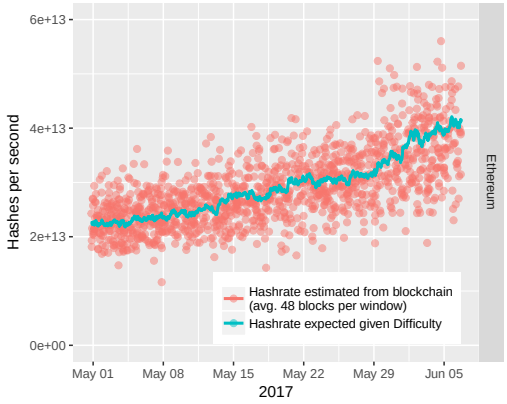

Figure 5: Network-wide hash rate, computed using MoM, of the Ethereum network compared to the hash rate calculated based on network difficulty and target value. Points represent a window of 12.5 minutes (about 50 blocks); for clarity, only a sample of all possible windows is plotted.

Additionally, the sample mean, $\overline{Y}$, must also be adjusted such that it is the sum of the observed hash values *issued by the given miner* divided by the number of intervals. Thus, once $\mathbf{Y}$ and $\overline{Y}$ only account for the blocks mined by a given miner, Eq. 11 and Alg. 1 are applicable.

Incremental Deployment. If no miners adopt status reports, network-wide hash rate can be calculated using the technique detailed in Section 5.1. For those miners who deploy status reports, the estimator in Section 4.1 can be used, while for those remaining, the MoM estimator via Eq. 14 provides a solution; the two values are then summed.

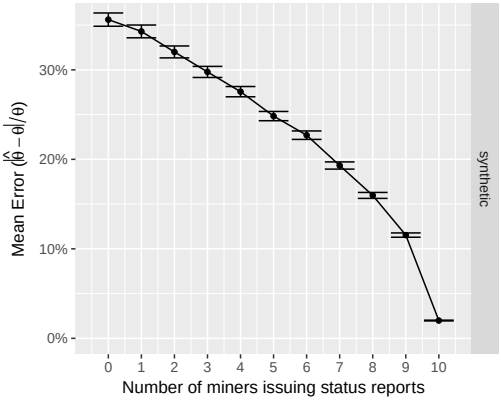

Figure 6: Mean error of network hash rate estimation as the number of miners issuing status reports increases. Synthetic blockchain with block discovery set to every 600 seconds. An 8-block window for MoM; status reports are issued about 320 times during the 8-block window. The accuracy of the network estimate increases dramatically as status reports are incrementally deployed. (Error bars are 95% c.i. of the mean over 8,800 trials for each point.)

We evaluated this approach using a synthetic block chain and 10 simulated miners, each with a fixed hash rate. Blocks were targeted for discovery every 600 simulated seconds. For hundreds of trials, we estimated the hash rate of the network using only blocks. In subsequent trials, we allowed an increasing number of miners to issue status reports every 15 seconds. For the remaining miners that did not issue reports, we used Eq. 14 to compute the number of hashes they performed, i.e., we used only the blocks issued by those miners not producing status reports. We then summed our estimates to calculate network-wide hash rate.

Figure 6 shows results of this experiment for a window of 5000 seconds (about 8 blocks, and about 320 status reports). Since 5000 seconds is a relatively small window for the MoM estimator (see Figure 3), the accuracy is initially low at 35%. The error in estimating the hash rate decreases rapidly as miners adopt the status report mechanism.

6 TAIL BOUNDS

As the CDFs plotted in Figures 2 and 3 show, our estimators have a distribution that is wide when there are too few reports or blocks, respectively, to work with. In this section, we characterize these tails. We provide Chernoff bounds for our status-reports-based method and employ empirical bootstrapping for our blockchain-only method.

These tail bounds provide a defense for thwarting the efforts of attackers who seek to push falsified results to the estimators. First, we describe the tail bounds, and then discuss their use against attackers.

In Section 7, we use these bounds to both characterize the variance of our estimator in practice, and as a method to thwart certain attacks.

6.1 Chernoff Bound for Status Reports

In Appendix A, we derive the upper and lower tail Chernoff bound on our estimate of β. The upper bound is the following

$$P\left(\frac{\beta-\hat{\beta}}{\hat{\beta}} \geq \pi\right) \leq \exp\left[\frac{n\pi}{1+\pi} - n\ln(1+\pi)\right], \quad (15)$$

where n is the number of status reports in a window and π represents the relative deviation of $\hat{\beta}$ from β. This result gives us a natural interpretation of the Chernoff bound, and a method to set it. In order to conservatively estimate a miner's mining power, it must be the case that $\hat{\beta}$, estimated by $\overline{V}$, in Equation 6, is large. A smaller $\hat{\beta}$ implies that we are more likely to *overestimate* a miner's mining power. Therefore, we want the fractional change from $\hat{\beta}$ to β (described by $(\beta-\hat{\beta})/\hat{\beta}$) to be bounded. When this fractional change is larger, our estimate for a miner's mining power is *higher*.

A merchant would use the bound as follows. First, from n status reports issued by a miner, she computes $\hat{\theta}_m$ using Eq. 5, and then sets $\hat{\beta}_m = S/\hat{\theta}_m$. Given n, she finds the value of π such that the RHS of Eq.6 is less than or equal to a threshold (e.g., 0.05). She then assumes $\beta_{mL} = \hat{\beta}_m/(\pi+1)$ as a lower bound with high probability. We now have an upper bound on the number of hashes performed by the miner as $\theta_{mH} = S/\beta_{mL}$.

In Section 7, we also make use of the lower bound with a similar formula, also derived in the Appendix:

$$P\left(\frac{\hat{\beta}-\beta}{\beta} \geq \pi\right) \leq \frac{1}{1+\pi}\exp\left[-n(\pi-\ln(1+\pi))\right]. \quad (16)$$

In this case, given n reports, the merchant solves for an appropriate π that meets her threshold, and then sets $\beta_{mH} = \hat{\beta}_m(\pi+1)$ and $\theta_{mL} = S/\beta_{mH}$.

6.2 Empirical Bounds for MoM

We found that tail bounds for Eq. 11 are not easily derived using standard techniques, such as Chernoff or Chebychev. We therefore calculate an empirical bound based on the well-known *bootstrap* [10, 15] technique.

Specifically, given a window of intervals containing a sequence of n blocks $\mathbf{O} = O_1, \ldots, O_n$, we create 10,000 new samples, each created by selecting n blocks *with replacement* from $\mathbf{O}$. For each new sample, we compute its sample mean $\overline{Y}$. We then select the 5th percentile of this distribution as $\overline{Y}_L$ and solve for $\tilde{\beta}_L$ using Eq. 11. Finally, we have $\theta_H = S/\tilde{\beta}_L$. Similarly, from the 95th percentile, we compute $\overline{Y}_H$ and $\tilde{\beta}_H$, as well as $\theta_L = S/\tilde{\beta}_H$. The approach is limited in accuracy for windows containing only a handful of blocks.

7 IMPLEMENTATION AND ANALYSIS

In this section, we apply our mining estimates to the problem of quantifying the threat posed by double-spend attacks. Specifically, we answer the question, *at what block depth can a merchant safely release goods to a consumer given current estimates of hash rates on the main chain?* We make use of historical data from the Bitcoin and Ethereum blockchains to characterize this value in practice. We also generated synthetic blockchain data so that we can evaluate the accuracy of our approach against known miner power.

7.1 Quantifying the Probability of a Double-spend Attack

In Section 2.3, we describe the double-spend attack against a transaction that appears in block B_1, which is a child of block B_0. Using the techniques from previous sections, we can quantify the amount of mining power devoted to mining on block B_1 and its descendants. Specifically, an estimate of all miners working on B_1 and descendants is available from Eq. 13 using the blockchain-only MoM estimator; or the same estimate can be more accurately calculated as the sum of individual miners' powers from status reports, using Eq. 6.

Let θ_0 be the network-wide hash rate in the case of MoM, or the sum of hash rates for all miners in the case of status reports, estimated from a *pre-window* of w blocks *ending with* B_0. Let θ_i be the hash rate calculated from a *post-window* of blocks *starting with* B_0 until B_i, where i is the latest block on the chain descendent from B_1. The merchant assumes that the amount of attacker mining power working against B_1 and its descendants is the complement of the proportion of mining power in the post-window to the pre-window:

$$q_i = 1 - \frac{\theta_i}{\theta_0}. \quad (17)$$

Of course, past performance is no prediction of what miners will do in the future: perhaps even after 1,000 blocks, the attacker will attempt to double-spend. To account for this possibility, we assume that from the honest miner's block $i+1$ onward, the attacker's mining power will be fixed at 12.7% (see Section 2.3). That is, we use a revision of Nakamoto's formula:

$$\mathcal{D}(q_i, z) = \begin{cases} 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!}\left(1 - (\frac{q*}{1-q*})^{z-k}\right), & \text{if } q_i < \frac{1}{2} \\ 1 & \text{, if } q_i \geq \frac{1}{2} \end{cases} \quad (18)$$

where $\lambda = \frac{zq_i}{1-q_i}$ and $q* = 0.127$. If the resulting probability is below the merchant's threshold (e.g., 0.1%), the goods are released to the consumer. For each block added to the chain, the merchant re-evaluates.

Bounding risk. A more conservative merchant, with perhaps more coin at risk, will account for error in the estimates.

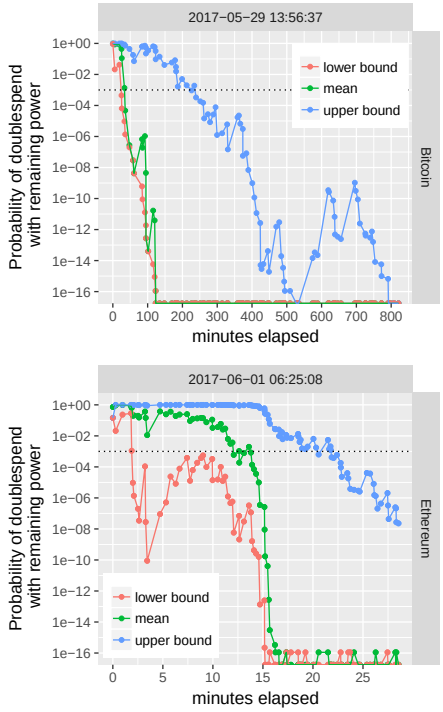

Figure 7: Results of applying Eq. 18 to an example block as its depth increases. The upper and lower bounds are calculated using bootstrapped estimates of the sample's 5th and 95th percentile, respectively (Eq. 19 and 20).

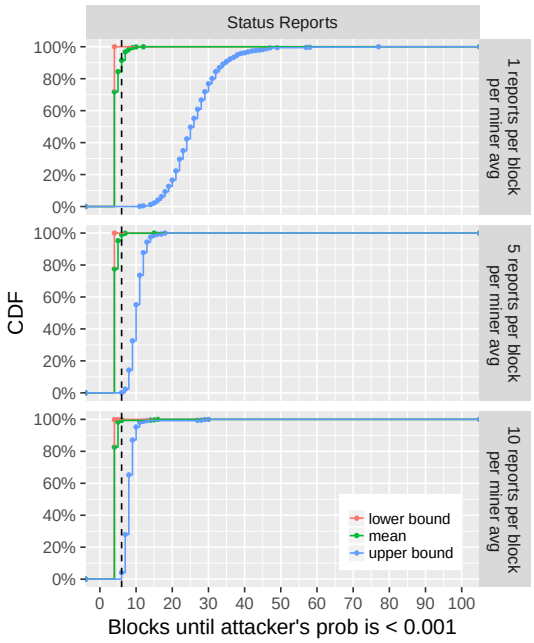

Figure 8: CDF for the depth required to defeat attackers with a given probability for synthetic data. The Chernoff bounds are weaker than the empirical bootstrapping method used for real blockchain data, but still demonstrate better results in the worst case.

For status reports, the merchant can make use of Chernoff bounds from Section 6.1; and for the MoM estimator, the merchant can use bootstrapped estimates of the sample's tail percentile from Section 6.2. The same approach can be used to account for attackers, as we detail below in Section 7.5.

To consider the worst-case scenario, we use an upper-bound estimate during the pre-window, and a lower-bound estimate during the post-window.

$$q_{iL} = 1 - \frac{\theta_{iL}}{\theta_{0H}}. \tag{19}$$

The consequence is that goods are released later.

A best-case scenario for an impatient merchant seeking the earliest time to release his goods is:

$$q_{iH} = 1 - \frac{\theta_{iH}}{\theta_{0L}}. \tag{20}$$

In other words, the two bounds characterize the error of estimating attacker mining power using q_i (Eq. 17). Both bounds can be applied to Eq. 18.

Implementation. We implemented our techniques for Bitcoin and Ethereum, using only information available on the blockchain. We released a public demo for Bitcoin at http://cs.umass.edu/~brian/blockchain.html. Once mining pools begin to release status reports, we will update to increase the accuracy of our estimates, as discussed in Section 5.2. The site is meant to show that we do not require updates to the existent protocols or underlying topology. (We will release source code at camera ready.)

Example output. Figure 7 is example of our technique applied to a single block as it ages. The figure shows the probability of attacker success (green line computed using Eq. 18) given block depth in terms of minutes, computed using a pre-window of 100 blocks. We also calculated the upper and lower bounds on success; the red and blue lines show Eqs. 19 and 20, respectively. To highlight a difference between Bitcoin and Ethereum, the figure is in minutes, instead of the number of blocks in the post window. Below, we examine the historical performance of the two networks.

7.2 Performance of Status Reports on Synthetic Data

We implemented our techniques for status reports on a synthetic blockchain to quantify its performance. The blockchain was parameterized to issue blocks about once every 600 simulated seconds. All miners issued status reports at rate of either 1, 5, or 10 times per block.

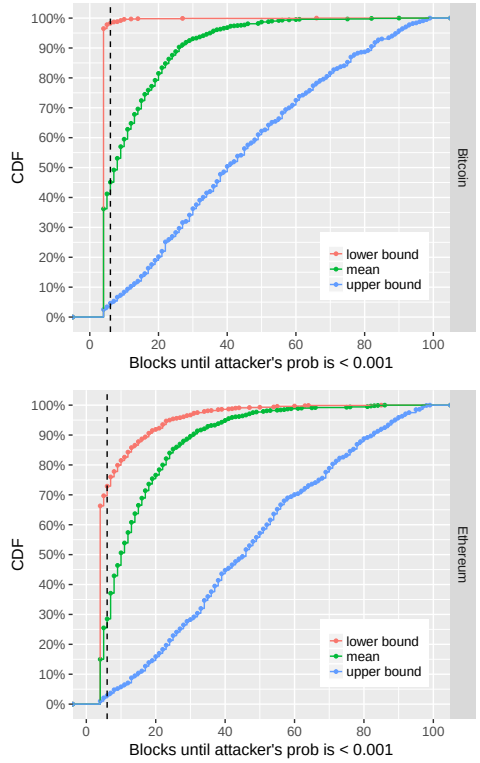

Figure 9: CDF for the depth required to defeat attackers with a given probability. Because Ethereum includes ommers to the blockchain, we are able to provide a tighter estimate.

We then sampled several hundred blocks from the chain and computed Eqs. 18, 19, and 20 using estimates from status reports and Chernoff bounds for each sample block as its depth increased. We used pre-windows of 100 blocks. We logged when each equation reached a set a threshold of 0.1% probability of attacker success. Figure 8 shows these results as a CDF for each equation.

The results show that when each mining pool issues on average 1 report between block announcements, at most 6 blocks are required 90% of the time to reach the 0.1% attack success threshold. This translates to one report every 15 seconds by each Ethereum mining pool, or one report every ten minutes for each Bitcoin mining pool. However, at this rate, the upper bound can be high. By increasing reports to about 10 per block per miner (about 2 every 3 seconds in Ethereum, and one every minute in Bitcoin), even the upper bound shows that 99% of blocks are within a depth of 13.

7.3 Historical Bitcoin and Ethereum Performance

We then performed the same analysis for historical data from Bitcoin and Ethereum. We computed values for Eqs. 18, 19, and 20 for over 2000 randomly selected blocks, each with a 100-block pre-window. For each block, we determined the depth required such that the probability of attacker success was less than 0.1%.

Figure 9 shows the CDF of these experiments for the two networks. First, the results are much poorer than the status report method. Second, it is notable that the two systems exhibit similar mean performance. For both systems, to ensure the probability of attacker success is lower than 0.1%, at least 10 blocks are required about 40% of the time; the upper bounds are much weaker, requiring at least 40 blocks 50% of the time. These values vary, but a strength of our technique is that it is calculated by merchants for specific blocks.

Ethereum's 15-second inter-block time results in a greater number of abandoned blocks and forks (called *ommers*); these are recorded to the blockchain as part the GHOST algorithm. Our algorithm makes use of the ommers as part of its estimate, and thus the bounds are tighter since there is more information in each window. While Figure 7 demonstrates that Ethereum more quickly reduces the risk of a double-spend attack in terms of wall clock time, Figure 9 shows that in terms of block depth, the two networks offer equivalent security because they implement the same algorithm.

The dashed vertical line represents bitcoind's choice of 6 blocks for declaring blocks as confirmed. According to the mean estimate, this choice is too early about 70% of the time in both networks. While there is error in the MoM technique, the experiment demonstrates that the 6-block choice is conservatively too early at least 30% of the time for Ethereum, according to our upper-bound estimate from Eq. 19. Because there are few forks in the Bitcoin blockchain, there is less data for making estimates, and we cannot make the same conservative claim for Bitcoin but expect it holds true.

7.4 Bandwidth Costs

Using only information available from the blockchain to estimate mining power requires no additional bandwidth.

The bandwidth required by status reports depends on the number and rate at which they are issued. The size of a status report is equal to the block header, plus a mining pool identifier and cryptographic signature. Status reports would be 80 bytes, the same size as block headers for Bitcoin; similarly, they would be 508 bytes for Ethereum. We assume that identifiers and signatures can be amortized by an SSL connection to a web site. Each report also can be easily encoded as 1–2 tweets on Twitter, which provides validated accounts and a secure global infrastructure for distributing short announcements. (It is unclear if this approach would violate Twitter's terms of service.) Secured RSS/json feeds from websites could also be used. In the case that two pools claim the same block, the claims can easily be resolved by the

real owner's signing of the claim with the block's coinbase private key. Note that bandwidth costs are independent of block size since they depend on only the block header.

To make a comparison of costs fair, we consider the frequency of status reports in terms of the network's block interval: e.g., 10 per 600 seconds in Bitcoin, and 10 per 15 seconds in Ethereum. We assume that 20 mining pools are active on each network, which mirrors current conditions. For Bitcoin, if ten 80-byte status reports are sent per miner per 600 seconds, then the total cost for recipients is 0.026 KBps. This additional traffic is small compared to keeping track of the blockchain itself, which is about 1 MB/600 sec = 1.7KBps.

For Ethereum, if ten 508-byte status reports are sent per miner per 15 seconds, then the total cost for recipients is 6.6 KBps. This additional traffic is much higher than keeping track of the blockchain itself, which is about 10KB / 15sec = 0.6 KBps. In absolute terms, this is still low — for example, streaming a song from Spotify at the lowest quality setting is about 12 KBps. Further, Ethereum performance would benefit significantly from just 1 report per miner per 15 seconds (i.e., costing 0.66 KBps).

7.5 Attacks on Hash Rate Estimates

Malicious miners may try to issue status reports that cause our hash rate estimates to be falsely higher or lower. In particular, a miner may want his mining rate to appear higher during the post-window, or lower during the pre-window.

Attacker Model. We assume our attacker has a total hash rate that is applied to the main chain in full during the pre-window, and then partially during the post-window. Outside of this restriction, attackers are challenging to model. For example, an attacker may take advantage in a change between the pre- and post-window in exchange rate of coin to a fiat currency that is used to pay electricity. Or an attacker may similarly take advantage of spot instances that fall in price on a cloud service, such as EC2. We do not take into account such economics or externalities. We discuss this limitation further in Section 8.

It is easy for an attacker to falsely lower their pre-window mining rate by simply not mining. Such an attacker is outside our model — but to put it another way, we assume the pre-window duration is sufficiently long to dissuade attackers from giving up income from not mining.

Attack analyses. An attacker might attempt to pre-mine the nonce values. However, the reports, just like blocks, are tied to a particular prior block, preventing the miner from creating status reports before a block is mined.

Similarly, an attacker might attempt to pre-mine within the window of a single block. For example, they may report the second order statistic as the second report. To prevent this attack, we can require status reports to include a nonce based on the previous status report. For example, if the report header contains a nonce n_i, then the nonce used for POW is the hashed concatenation: $n'_i = H(n'_{i-1}|n_i)$, where n_0 is the hash of the prior block and $H(\cdot)$ is a hash function.

A more advanced strategy is for an attacker to wait until she gets lucky. She mines until her status reports produce an overestimate, and then diverts resources to a double-spend attack. Or, the attacker stops mining towards a status report at each interval when finding a minimum hash that satisfies the target mean; for the remainder of the interval, an attacker uses her mining power to execute a double-spend attack.

For these more advanced strategies, the attacker is taking advantage of the tail of the estimator's distribution. Chernoff bounds are a powerful technique against such attacks. Yet, we do not claim that Chernoff bounds defend against all possible attacks. Consider an attacker with mining power x that devotes $2x/3$ to mining honestly, and $x/3$ to the double-spend attack. If the miner never reveals their $x/3$ power until the attack succeeds, our techniques would certainly fail to detect this attacker. We have not analyzed whether this attack is economically profitable.

In the case of status reports, an attacker can lower the merchant's estimate of their mining power by skipping status reports or sending a hash that is not their lowest. We hypothesize that this attack would be detectable because the rate estimated by the status reports and that from the blocks they add to the chain (see Section 5.2) would not be statistically equal. We do not expect this comparison to work for short windows of time, but neither have we determined the time necessary for any differences to be statistically significant.

In the case of blockchain-only MoM estimations, the bootstrap sample distribution provides protection similar to the Chernoff bounds as the tails are difficult for an attacker to avoid. Notably, attackers of the blockchain-only method face an additional hurdle: to affect the outcome, they must mine valid blocks.

8 LIMITATIONS

There are several limitations to our approach and contributions. As discussed in Section 7.5, Bitcoin and Ethereum are open blockchains that allow miners to join or leave at any time [35]. We are unable to account for latecomers that were previously silent during the pre-window or new to mining starting at the post-window.

Real attackers may employ strategies that are more complex than we considered. For example, mining pools may hide their hash rates by switching their mining resources between networks with the same proof of work algorithm, such as between Bitcoin and Litecoin (Ethereum's POW algorithm is designed to advantage GPUs rather than ASICs). We note therefore, that a conservative merchant may elect

to sum the hash rates across a series of blockchains. (Fortunately, our approach is not subject to the Sybil attack [14], as we do not separate out mining pool workloads.) But in general, externalities that we cannot observe or quantify may drive attacker strategy and behavior. None of our analysis considers the economic profitability of attacker strategy.

We have not studied the optimal duration of the pre-window in Section 7. A longer window assumes steady hash rates over long periods of time, while missing finer-grained fluctuations. A short window accounts for recent history while missing long-term trends. Additionally, in the same section, we do not offer a specific threshold for bootstrap or Chernoff bounds. On the other hand, this may be considered a parameter set by the merchant, who may tune the tradeoff between security and delay to their preference.

Finally, we offer no direct incentive for miners to issue status reports. However, doing so may encourage greater trust and use of blockchains, which benefits miners.

9 RELATED WORK

There have been many informal proposals to determine the amount of hash power that went into unconfirmed transactions — see [6] for a list. To the best of our knowledge, no algorithms have been formally defined, evaluated, or implemented; these suggestions are so preliminary that we cannot compare our approach. Further, none suggest a method that uses only existing blockchain information. (A preliminary version of this paper appeared as a tech report [28].)

The security of a 6-block wait for transaction confirmation has been studied by Rosenfeld [29] and discussed by Bonneau [8]; see also [7]. Many papers have examined the double-spend attack in a variety of contexts. Sompolinsky introduced the GHOST protocol [32], now incorporated in Ethereum. They showed that double-spend attacks become more effective as either the block size or block creation rate increase (when GHOST is not used). Sapirshtein et al. [30] first observed that some double-spend attacks can be carried out essentially cost-free in the presence of a concurrent selfish mining [19] attack. More recent work extends the scope of double-spends that can benefit from selfish mining to cases where the attacker is capable of *pre-mining* blocks on a secret branch at little or no opportunity cost [33] and possibly also under a concurrent eclipse attack [22].

Several past work have relied on stability in the blockchain as a requirement for a higher-level service. For example, sidechains[4] employ a confirmation period, which is "a duration for which a coin must be locked on the parent chain before it can be transferred to the sidechain," stating that "a typical confirmation period would be on the order of a day or two" but not reasoning why. Our technique offers a quantitative approach to selecting the confirmation period. Lightning networks [1, 24] and fair-exchange protocols [5] similarly assume that an initial commitment or refund transaction is not revokable (via a double-spend attack), and our approach would quantify that risk.

10 CONCLUSION

We designed and evaluated two methods to accurately estimate network hash rates to quantify the probabilistic consensus of blockchain systems. Our first method is based on short status reports issued by miners, while the second uses only blocks that are published to the blockchain. The latter approach is less accurate than status reports because there is less information available per window of time. To evaluate the accuracy of both methods, we derived bounds on our estimates and presented simulations using a synthetic blockchain as ground truth. We also showed that these methods can be used together in an incremental deployment strategy.

We also implemented our blockchain-only estimator to show the historical network-wide hash rate of Ethereum and Bitcoin. Finally, we provided a framework to estimate consensus on blocks and transactions, using our estimates. On Bitcoin and Ethereum, we analyzed the depth required before an attacker's success is estimated to be 0.001 or less. We emphasized the importance of status reports by demonstrating, using synthetic blockchain data, that this depth is significantly reduced.

A STATUS REPORT CHERNOFF BOUNDS

We derive a bound on the relative deviation of $\hat{\beta}$ from β.

Upper tail bound. Let $\mathbf{R} = \sum_{i=1}^{n} V_i$ with $V_i \sim \mathsf{Expon}(\beta)$. Jansen [25] shows that for any $\lambda \leq 1$,

$$P(\mathbf{R} \leq \lambda E[\mathbf{R}]) \leq \exp\left[-\frac{1}{\beta}E[\mathbf{R}](\lambda - 1 - \ln(\lambda))\right]. \quad (21)$$

In our case, $E[\mathbf{R}] = n\beta$ and $\mathbf{R} = n\hat{\beta}$, where $\hat{\beta}$ is the observed sample mean of the status reports. Let $\lambda = 1/(1+\pi)$. It follows that

$$\begin{aligned} P\left(\beta - \hat{\beta}/\hat{\beta} \geq \pi\right) &= P(n\hat{\beta} \leq \lambda n\beta) \\ &\leq \exp\left[-\frac{1}{\beta}n\beta(\lambda - 1 - \ln(\lambda))\right] \\ &= \exp\left[\frac{n\pi}{1+\pi} - n\ln(1+\pi)\right]. \quad (22) \end{aligned}$$

Lower tail bound. Jansen [25] shows that for any $\lambda \geq 1$,

$$P(\mathbf{R} \geq \lambda E[\mathbf{R}]) \leq \lambda^{-1}\exp\left[-\frac{1}{\beta}E[\mathbf{R}](\lambda - 1 - \ln(\lambda))\right] \quad (23)$$

If $\lambda = 1 + \pi$, then we have

$$\begin{aligned} P\left(\hat{\beta} - \beta/\beta \geq \pi\right) &= P(n\hat{\beta} \geq \lambda n\beta) \\ &\leq \lambda^{-1}\exp\left[-\frac{1}{\beta}n\beta(\lambda - 1 - \ln(\lambda))\right] \\ &= \frac{1}{1+\pi}\exp\left[-n(\pi - \ln(1+\pi))\right]. \quad (24) \end{aligned}$$

B CHERNOFF BOUND FOR BLOCK-ESTIMATED HASH RATE

In Section 5 we defined random sample Y as the set of partially observable status reports that result from block announcements. Here we wish to develop a Chernoff bound for the deviation of estimator $\tilde{\beta}$ from the sample mean $\overline{Y}$. In particular we wish to find an upper bound on $P((\beta - \tilde{\beta})/\tilde{\beta} \geq \pi)$, the probability that the relative difference between the estimate of β and its true value is greater than π.

As in Section 5, define $O = \{V_i \mid I_i \in \mathcal{I}_B\}$ where $\mathcal{I}_B$ is the set of k intervals that produce observable status reports. Let $W = \sum_{i \in k} O_k$ and $Z = \sum_{i \in k} O_k$. Note that

$$W = \sum_{i=1}^{n} Y_i = n\overline{Y} = k\overline{O} = \sum_{i=1}^{n} O_i = Z \tag{25}$$

where n is the number of intervals.

Now working in conjunction with Equation 25 we have the following.

$$\begin{aligned} P\left(\frac{\beta-\tilde{\beta}}{\tilde{\beta}} \geq \pi\right) &= P\left(\beta \geq (\pi+1)\left[\overline{Y} + e^{-t/\tilde{\beta}}(t-\tilde{\beta})\right]\right) \\ &= P\left(\beta \geq \tfrac{(\pi+1)k}{n}\overline{O} + (\pi+1)e^{-t/\tilde{\beta}}(t-\tilde{\beta})\right) \\ &= P\left(\tfrac{n}{(\pi+1)k}\left[\beta - (\pi+1)e^{-t/\tilde{\beta}}(t-\tilde{\beta})\right] \geq \overline{O}\right) \\ &= P\left(Z \leq \tfrac{n}{(\pi+1)}\left[\beta - (\pi+1)e^{-t/\tilde{\beta}}(t-\tilde{\beta})\right]\right). \end{aligned} \tag{26}$$

Or, for an absolute bound,

$$\begin{aligned} P\left(\beta - \tilde{\beta} \geq \delta\right) &= P\left(\beta \geq \delta + \overline{Y} + e^{-t/\tilde{\beta}}(t-\tilde{\beta})\right) \\ &= P\left(\beta - e^{-t/\tilde{\beta}}(t-\tilde{\beta}) - \delta \geq \tfrac{k}{n}\overline{O}\right) \\ &= P\left(\overline{O} \leq \tfrac{n}{k}\left[\beta - e^{-t/\tilde{\beta}}(t-\tilde{\beta}) - \delta\right]\right) \\ &= P\left(Z \leq n\left[\beta - e^{-t/\tilde{\beta}}(t-\tilde{\beta}) - \delta\right]\right) \end{aligned} \tag{27}$$

Let $f_O(x)$ denote the density function of O_i. For each V_i, $V_i \sim \mathsf{Expon}(\beta)$, and thus

$$\begin{aligned} f_O(x) &= P(O_i = x) \\ &= P(V_i = x \mid V_i \leq t) \\ &= \tfrac{P(V_i \leq t \mid V_i = x)P(V_i = x)}{P(V_i \leq t)} \\ &= \begin{cases} 0 & x > t \\ \frac{e^{-x/\beta}}{\beta(1-e^{-t/\beta})} & x \leq t \end{cases} \end{aligned} \tag{28}$$

Because the O_i are iid, the general form of the Chernoff bound for Z can be expressed as

$$P(Z \leq a) \leq \min_{\tau > 0} e^{\tau a}\left[E[e^{-\tau O_i}]\right]^n, \tag{29}$$

where O_i is an arbitrary random variable from sample O. According to Equation 28, $E[e^{-\tau O_i}]$ is given by

$$\begin{aligned} E[e^{-\tau O_i}] &= \int_0^\infty e^{-\tau x} f_O(x)dx \\ &= \tfrac{1}{\beta(1-e^{-t/\beta})}\int_0^t e^{-\tau x}e^{-x/\beta}dx \\ &= \tfrac{e^{t/\beta} - e^{-t\tau}}{(e^{t/\beta}-1)(\beta\tau+1)} \end{aligned} \tag{30}$$

Now define

$$a = \frac{n}{(\pi+1)}\left(\beta - (\pi+1)e^{-t/\tilde{\beta}}(t+\tilde{\beta})\right) \tag{31}$$

for $\pi > 0$. Because $Z = n\overline{O}$, it follows that

$$\begin{aligned} &P\left(\tfrac{\beta-\tilde{\beta}}{\tilde{\beta}} \geq \pi\right) \\ &= P(Z \leq a) \\ &= \min_{\tau > 0} \exp\left[n\tau\left(\tfrac{\beta}{\pi+1} - e^{-t/\tilde{\beta}}(t+\tilde{\beta})\right)\right]\left[\tfrac{e^{t/\beta}-e^{-t\tau}}{(e^{t/\beta}-1)(\beta\tau+1)}\right]^k \end{aligned} \tag{32}$$

REFERENCES

[1] 2015. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf. (July 2015).

[2] 2016. Gnosis. https://www.gnosis.pm. (November 2016).

[3] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. 2016. "MedRec: Using Blockchain for Medical Data Access and Permission Management. In *Proc. Intl. Conf. on Open and Big Data*. 25–30.

[4] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. 2014. Enabling Blockchain Innovations with Pegged Sidechains. Technical report. (Oct 22 2014).

[5] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. 2012. Bitter to better—how to make bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security*. Springer, 399–414.

[6] Bryan Bishop. 2015. bitcoin-dev maillling list: Weak block thoughts... https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-September/011158.html. (Sep 2015).

[7] bitcoin 2015. Confirmation. https://en.bitcoin.it/wiki/Confirmation. (February 2015).

[8] Joseph Bonneau. 2015. How long does it take for a Bitcoin transaction to be confirmed? https://coincenter.org/2015/11/what-does-it-mean-for-a-bitcoin-transaction-to-be-confirmed/. (November 2015).

[9] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J.A. Kroll, and E.W. Felten. 2015. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *IEEE S&P*. 104–121. http://doi.org/10.1109/SP.2015.14

[10] George Casella and Roger L. Berger. 2002. *Statistical inference*. Brooks Cole, Pacific Grove, CA. http://opac.inria.fr/record=b1134456

[11] Kyle Croman et al. 2016. On Scaling Decentralized Blockchains . In *Workshop on Bitcoin and Blockchain Research*.

[12] Digix. 2017. https://www.dgx.io/. (Last retrieved June 2017).

[13] DigixDAO. 2017. https://www.dgx.io/dgd/. (Last retrieved June 2017).

[14] J. Douceur. 2002. The Sybil Attack. In *Proc. Intl Wkshp on Peer-to-Peer Systems (IPTPS)*.

[15] Bradley Efron. 1982. *The jackknife, the bootstrap and other resampling plans*. Society for industrial and applied mathematics (SIAM).

[16] Ethash. 2017. https://github.com/ethereum/wiki/wiki/Ethash. (Last retrieved June 2017).

[17] ethereum. Ethereum Homestead Documentation. http://ethdocs.org/en/latest/. (????).

[18] Etheria. 2017. http://etheria.world. (Last retrieved June 2017).

[19] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. *Financial Cryptography* (2014), 436–454. http://doi.org/10.1007/978-3-662-45472-5_28

[20] William Feller. 1968. *An Introduction to Probability Theory and its Applications: Volume I*. Vol. 3. John Wiley & Sons London-New York-Sydney-Toronto.

[21] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. 2015. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 281–310.

[22] Arthur Gervais, Ghassan O. Karame, Karl Wust, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. 2016. On the Security and Performance of Proof of Work Blockchains. https://eprint.iacr.org/2016/555. (2016).

[23] Hashcash. 2017. https://en.bitcoin.it/wiki/Hashcash. (Last retrieved June 2017).

[24] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. 2017. TumbleBit: An untrusted Bitcoin-compatible anonymous payment hub. In *Proc. ISOC Network and Distributed System Security Symposium (NDSS)*.

[25] Svante Janson. 2014. *Tail Bounds for Sums of Geometric and Exponential Variable*. Technical Report. Uppsala University.

[26] Litecoin. 2017. https://litecoin.org. (Last retrieved June 2017).

[27] Satoshi Nakamoto. 2009. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf. (May 2009).

[28] A. Pinar Ozisik, Gavin Andresen, George Bissias, Amir Houmansadr, and Brian Neil Levine. 2016. *A Secure, Efficient, and Transparent Network Architecture for Bitcoin*. Technical Report UM-CS-2016-006. University of Massachusetts, Amherst, MA. https://web.cs.umass.edu/publication/details.php?id=2417

[29] Meni Rosenfeld. 2012. Analysis of hashrate-based double-spending. https://bitcoil.co.il/Doublespend.pdf. (December 2012).

[30] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. 2015. Optimal Selfish Mining Strategies in Bitcoin. https://arxiv.org/pdf/1507.06183.pdf. (July 2015).

[31] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *IEEE S&P*. 459–474. http://dx.doi.org/10.1109/SP.2014.36

[32] Yonatan Sompolinsky and Aviv Zohar. 2015. Secure high-rate transaction processing in Bitcoin. *Financial Cryptography and Data Security* (2015). http://doi.org/10.1007/978-3-662-47854-7_32

[33] Yonatan Sompolinsky and Aviv Zohar. 2016. Bitcoin's Security Model Revisited. https://arxiv.org/abs/1605.09193. (May 2016).

[34] F. Tschorsch and B. Scheuermann. 2016. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Communications Surveys Tutorials* PP, 99 (2016), 1–1. https://doi.org/10.1109/COMST.2016.2535718

[35] Marko Vukolić. 2015. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*. Springer, 112–125.