Peace Through Superior Puzzling:
An Asymmetric Sybil Defense

Diksha Gupta
Department of Computer Science
University of New Mexico
Albuquerque, USA
dgupta@unm.edu

Jared Saia
Department of Computer Science
University of New Mexico
Albuquerque, USA
saia@cs.unm.edu

Maxwell Young
Dept. of Computer Science and Eng.
Mississippi State University
MS, USA
myoung@cse.msstate.edu

***Abstract*—A common tool to defend against Sybil attacks is proof-of-work, whereby computational puzzles are used to limit the number of Sybil participants. Unfortunately, current Sybil defenses require significant computational effort to offset an attack. In particular, good participants must spend computationally at a rate that is proportional to the spending rate of an attacker.**

In this paper, we present the first Sybil defense algorithm which is asymmetric in the sense that good participants spend at a rate that is asymptotically less than an attacker. In particular, if T is the rate of the attacker's spending, and J is the rate of joining good participants, then our algorithm spends at a rate of $O(\sqrt{TJ} + J)$.

We provide empirical evidence that our algorithm can be significantly more efficient than previous defenses under various attack scenarios. Additionally, we prove a lower bound showing that our algorithm's spending rate is asymptotically optimal among a large family of algorithms.

I. Introduction

The last decade has seen explosive growth in systems that are permissionless in that participants are free to join and leave at will. All such systems are open to the well-known ***Sybil attack*** [32], in which an adversary uses a large number of forged IDs to take control of the system. One of the most popular tools for Sybil defense is proof-of-work (PoW): IDs must periodically solve computational puzzles, thereby limiting the number of forged IDs.

Unfortunately, current PoW-based Sybil defenses suffer from a key weakness: the computational effort expended by the good IDs in solving puzzles must at least be equal to the computational effort of an attacker.

We present the first algorithm to address this problem. Our algorithm is a Sybil defense that is asymmetric in the sense that the good IDs spend at a rate that is asymptotically less than the attacker. As our title implies, such an asymmetry may serve as a convincing deterrent against Sybil attacks.

Specifically, we prove that our algorithm ***Geometric Mean COMputation (*GMCom*)*** spends at a rate of $O(\sqrt{TJ} + J)$, where T is the spending rate of the attacker, and J is the join rate for good IDs. We also prove a lower bound showing this rate is asymptotically optimal among a large family of algorithms.

This work is supported by the National Science Foundation grants CNS-1318880, CCF-1320994, CCF 1613772, CNS 1816076, and by a research gift from C Spire.

A. Our Model and Problem

Our system consists of ***identifiers (IDs)***, and an ***adversary***. All ***good*** IDs follow our algorithm, and all ***bad*** IDs are controlled by the adversary.

Puzzles. We assume a source of computational puzzles of varying difficulty, whose solutions can not be stolen or pre-computed. These is a standard assumption in PoW systems [7], [54], [62]. We sketch briefly how this assumption is typically achieved. First, there is a globally known hash function. Solving a puzzle requires finding an input to the function that produces a sufficiently small output. Informally, the more difficult the puzzle the smaller the required output. The input found is the puzzle solution. To avoid stealing, a digital-signature public-key is included in the solution. To avoid pre-computation of solutions, a current, globally-known random seed is also included in the puzzle solution. For more details, see Section II and Section VI.

Adversary. A single adversary controls all bad IDs. This pessimistically represents perfect collusion and coordination by the bad IDs. Bad IDs may arbitrarily deviate from our protocol, including sending incorrect or spurious messages.

The adversary controls an α-fraction of computational power, where $\alpha > 0$ a small constant. Thus, in a single round where all IDs are solving puzzles, the adversary can solve an α-fraction of the puzzles. This assumption is standard in past PoW literature [7], [36], [62], [76].[1]

Our algorithms employ public key cryptography, and so our adversary is computationally bounded. Further, we assume the adversary knows our algorithm, but does not know the private random bits of any good ID.

Communication. Communication among good IDs occurs through a broadcast primitive, **Diffuse**, which enables a good ID to send a message to all IDs. As in past work, we assume that the time to diffuse a message is small in comparison with the time to solve a puzzle. Such a primitive is a standard assumption in PoW schemes [14], [34], [36]. Our adversary can read messages diffused by good IDs before sending its own, and can also send messages directly to any ID.

[1] We use $\alpha = 1/14$ in our analysis (see Lemma 6); however, this fraction can likely be increased, and this is left as an area of future work.

Time is discretized into ***rounds***, where a round is an upper-bound on the time required to solve a puzzle of computational cost 1, and diffuse its solution. All IDs are assumed to be synchronized, but our algorithms can tolerate a small amount of skew; details are in Section 2.9 of [42].

Joins and Departures. The system is dynamic with IDs joining and departing over time, and so system membership may change from round to round. Rounds are grouped into ***epochs***. Succinctly, if the set of IDs at the start of an epoch is A, and the current set of IDs is B, then an epoch ends when $|A \otimes B| \geq |A|/3$, where $A \otimes B = (A \cup B) - (A \cap B)$ is the symmetric difference.

For $i \geq 1$, let $\boldsymbol{J_i}$ be the join rate of good IDs during epoch i; that is, the number of good IDs that join over epoch i divided by the duration of epoch i. We make the following assumptions with respect to the good IDs:

- **A1.** For any epoch, the departure rate of good IDs is within a factor of $(1 \pm \frac{1}{2})$ of the join rate of good IDs.
- **A2.** In any round in an epoch, at most $\epsilon_0\, x$ good IDs depart, where $\epsilon_0 > 0$ is a small constant, and x is the number of good IDs at the beginning of the epoch.
- **A3.** For $i \geq 1$, $J_{i-1}/2 \leq J_i \leq 2J_{i-1}$.
- **A4.** For any $i \geq 1$, $\ell_i \leq 2|S_{i-1}|/J_i$, where ℓ_i is the length of epoch i, and S_{i-1} is the set of IDs in the system at the end of epoch $i-1$.
- **A5.** Let $C > 0$ be some fixed constant. Then, in any epoch, for any integer $x \geq 1$, at most Cx good IDs join by time x/J_i in the epoch.

Additionally, we assume when a departure event occurs for a good ID, the departing ID is selected uniformly at random from the set of good IDs in the system. Further, we assume our algorithm has good estimates of join and departure rates, and also the size of the symmetric difference discussed above. In [42], we show how to efficiently achieve such estimates using sampling and heartbeat messages.

Finally, the minimum number of good IDs in the system at any point is assumed to be at least some value $\boldsymbol{n_0}$.

Our Problem. We must maintain the following two invariants.

Population Invariant: The fraction of bad IDs in the system is always less than $1/2$.

Committee Invariant: There is a committee that is known to all IDs; has size $\Theta(\log n_0)$; and contains less than a $1/2$ fraction of bad IDs.

Why are these invariants useful? The population invariant bounds the amount of system resources consumed by bad IDs.

The committee invariant allows for a scalable solution to the ***Byzantine consensus*** problem [53] (see Section I-A for details). This allows a committee to agree on and execute operations in the system despite the presence of bad IDs.

B. Our Results

We can now state our result for our algorithm GMCOM.

Theorem 1. GMCOM *has the following properties for a number of ID joins and departures that is polynomial in n_0, with error probability that is polynomially small in n_0.*

(1) The population and committee invariants are maintained.

(2) Consider any subset of epochs not containing epoch 1. Then the algorithmic spending rate is $O(\sqrt{T\,J}+J)$, where T is the adversarial spending rate, J is the good ID join rate, and all rates are taken over the subset of epochs.

Our lower bound is as follows. We define a ***purge-based*** algorithm to be any algorithm where (1) IDs pay some cost of $\Omega(1)$ to join; and (2) after a constant fraction of the population changes, all IDs must pay $\Omega(1)$ to remain in the system (else they are purged). Then we prove:

Theorem 2. *For any purge-based algorithm, there is an adversarial strategy ensuring the following for any epoch. The algorithmic spending rate is $\Omega(\sqrt{T\,J} + J)$, where the algorithmic spending rate, T, and J are taken over the epoch.*

C. Model Discussion

We emphasize that assumptions A1-A5 allow for a highly-dynamic system, since the number of good IDs that can join or depart in any single round may be linear in the system size at the beginning of the epoch; additionally, there are no constraints on the behavior of the bad IDs. Furthermore, there is nothing special the constants used in A1-A5, and they can be modified at the expense of increasing the hidden constants in our asymptotic resource costs.

With regard to n_0 and the guarantees made in in Theorem 1, we note that, in practice, there are distributed systems for which n_0 is sizable. For example, measurements of the Mainline DHT find a minimum size of over 14 million IDs [78], and data collection on the Bitcoin network indicates a network of more than $5,000$ IDs over the past 2 years [17].

Finally, in the event where multiple IDs enter the system near-simultaneously such that their ordering cannot be determined, these joins are assumed to be serialized and agreed upon by the committee via Byzantine consensus.

D. Related Work

The Sybil Attack. Our work applies to the Sybil attack [32]. In addition to our recent work [43], there is large body of literature on defenses (see surveys [9], [31], [47], [59], [64]). *Critically, none of these prior defenses are asymmetric*.

PoW is a natural tool for combatting Sybil attacks since computing power costs money, whether obtained via Amazon AWS [4] or a botnet rental [5]. This is borne out by Bitcoin and other cryptocurrencies, where an adversary has a strong incentive to wield as much computational power as possible.

Beyond PoW, several other approaches have been proposed. In a wireless setting, Sybil attacks can be mitigated via *radio-resource testing* which relies on the inability of the adversary to listen to many communication channels simultaneously [37], [38], [61], [64]. However, this approach may

fail if the adversary can monitor most or all of the channels. Furthermore, again, none of these defenses are asymmetric.

Several results leverage social networks to yield Sybil resistance (see the survey [79]). However, social-network information may not be available in some settings. Another approach is the use of network measurements to verify the uniqueness of IDs [11], [30], [35], [55], [72], [77], but these techniques rely on accurate measurements of latency, signal strength, or round-trip times, and this may not always be possible. Containment strategies for overlays are examined in [28], [71], but the extent to which good participants are protected from the malicious actions of Sybil IDs is limited.

PoW and Alternatives. As a choice for PoW, computational puzzles provide certain advantages. First, verifying a solution is much easier than solving the puzzle itself. This places the burden of proof on devices who wish to participate in a protocol rather than on a verifier. In contrast, bandwidth-oriented schemes, such as [76], require verification that a sufficient number of packets have been received before any service is provided to an ID; this requires effort by the verifier that is proportional to the number of packets.

A recent alternative to PoW is ***proof-of-stake (PoS)*** where security relies on the adversary holding a minority stake in an abstract finite resource [1]. When making a group decision, PoS weights each participant's vote using its share of the resource; for example, the amount of cryptocurrency held by the participant. A well-known example is ALGORAND [36], which employs PoS to form a committee. A hybrid approach using both PoW and PoS has been proposed in the Ethereum system [3]. We note that PoS can only be used in systems where the "stake" of each participant is globally known. Thus, PoS is typically used only in cryptocurrency applications.

There has been a significant amount of related work on consensus [51], [67], [68] and scalable blockchains [39], [45], [57], [80]. When the number of bad participants is assumed to be a bounded minority, several adversarial fault-tolerant systems exist (see the survey [75]).

Byzantine Consensus. The Byzantine consensus problem [53] is described as follows. Each good ID has an initial input bit and the goal is for (a) all good IDs to decide on the same bit; and (b) this bit to equal the input bit of at least one good ID.

Byzantine consensus enables participants in a distributed network to reach agreement on a decision, even in the presence of a malicious minority. Thus, it is a fundamental building block for many cryptocurrencies [19], [33], [36], [41]; trustworthy computing [20], [22]–[25], [52], [73]; peer-to-peer networks [2], [65]; and databases [60], [69], [81]. Establishing Byzantine consensus via the use of committees is a common approach; for examples, see [36], [49], [56].

II. Our Algorithm

The pseudocode for our algorithm, Geometric Mean COMputation (GMCom), is provided in Figure 1. We begin with describing the main ingredients of our algorithm, followed by an overview of GMCom. We analyze GMCom in Section III-A and Sections III.

A. Preliminaries

GenID. To initialize our system, we make use of an algorithm created by Andrychowicz and Dziembowski [7], which we call GenID. This algorithm, used in Step 1 of our algorithm, creates an initial set of IDs, no more than an α-fraction of which are bad. The GenID algorithm also selects a committee of logarithmic size that has a majority of good IDs. The GenID algorithm has significant, but polynomial, computational cost; thus we use it only once during the lifetime of our system.

The Committee. Our algorithm maintains a committee of size $\Theta(\log n_0)$ with a majority of good IDs as in [43]. This committee uses Byzantine consensus: to maintain values for key variables, to decide when an epoch ends, to issue random seeds for puzzles, and create a new committee.

B. Overview of GMCom

The execution of GMCom is broken into ***epochs***, as defined in Section I-A. During the epoch, each joining ID must solve a puzzle of difficulty $\max\left\{\lceil J_{\text{cur}}/\tilde{J}_{i-1}\rceil, 1\right\}$, where J_{cur} is the current join rate, and $\tilde{J}_{i-1}$ is approximately the join rate of good IDs in the previous epoch. Intuitively, this means that if the current join rate is high compared to the join rate in the last epoch, the entrance cost will be high. We provide more detailed intuition about this cost in Section II-C.

Let $\mathcal{S}_i$ be the set of IDs at the end of epoch i, and let $\mathcal{S}_{\text{cur}}$ be the current set of IDs. Then recall that epoch i ends when $|\mathcal{S}_{\text{cur}} \otimes \mathcal{S}_i| \geq |\mathcal{S}_{i-1}|/3$. At this point, a ***purge*** is initiated by the committee as follows. All IDs are immediately issued a puzzle of difficulty 1, and each ID must respond with a valid solution within 1 round, or else be removed from the system.

Recall that in order to guarantee that puzzle solutions are fresh, we use a random string $\boldsymbol{r}$ that must be incorporated into puzzle solutions, and is periodically changed. This random string is generated by the committee in Step 2(a). Note that since the committee has a majority of good IDs, we can use a secure multiparty protocol to generate r; for example, see [74].

Recall further that to prevent puzzles from being stolen, each puzzle solution also incorporates the public key of the digital signature of the ID that solved it. Thus, the ID that solved the puzzle will receive credit for it, and correctly be added to the system. Additionally, any ID that is in the system can digitally sign a message in a way that validates its membership. Notably, this is true for the IDs in the committee, which must digitally sign all messages they diffuse.

Finally, we note that, although our pseudocode and analysis assumes perfect estimates of all system variables, this is not strictly necessary in practice. In particular, the results in Theorem 1 still hold even if we only have estimates of rates are off by small constant factors from their true values, and estimates of ID sets that have a small symmetric difference with their true values.

GMCom

Key Variables

i: epoch number
$\mathcal{S}_i$: set of IDs at end of epoch i (after the purge)
$\mathcal{S}_{\text{cur}}$: current set of IDs
J_{cur}: current number of join events in epoch i divided by current time elapsed in epoch i.
$\tilde{J}_i$: estimate of good join rate in epoch i

Initialization:

$\mathcal{S}_0 \leftarrow$ set of IDs returned by initial call to GenID. The initial committee is also selected by GenID.
$\tilde{J}_0 \leftarrow \infty$.
$i \leftarrow 1$.

The committee maintains all variables above and makes all decisions using Byzantine Consensus.

For each epoch i:

1. Each joining ID solves an entrance puzzle of difficulty $\max\left\{\lceil J_{\text{cur}}/\tilde{J}_{i-1}\rceil, 1\right\}$ and diffuses the solution.
2. When $|\mathcal{S}_{\text{cur}} \otimes \mathcal{S}_{i-1}| \geq |\mathcal{S}_{i-1}|/3$ do:

   Perform Purge:

   (a) The committee generates and diffuses a random string r to be used in puzzles for this purge and entrance for the next epoch.
   (b) $\mathcal{S}_i \leftarrow$ set of IDs returning difficulty 1 puzzle solutions within 1 round.
   (c) The current committee selects a new committee of size $\Theta(\log n_0)$ from $\mathcal{S}_i$ uniformly at random.

   Update Variables:

   (d) $\tilde{J}_i \leftarrow \left(|\mathcal{S}_{i-1} \otimes \mathcal{S}_i| - \alpha(|\mathcal{S}_{i-1}| + |\mathcal{S}_i|)\right)/(\text{duration of epoch } i)$.
   (e) $i \leftarrow i + 1$.

Fig. 1: Pseudocode for GMCom.

C. Intuition for Algorithm Design

We highlight two related challenges in designing GMCom: (i) the formulation of an entrance-cost function that yields an asymmetric-cost guarantee, and (ii) a robust method for estimating the true join rate of good IDs which is necessary for evaluating this entrance-cost function.

Entrance-Cost Function. In the absence of an attack, the entrance cost should be small. In this case, J_{cur} is at most a constant factor greater than $\tilde{J}_{i-1}$, since the good ID join rate changes by at most a constant factor from epoch to epoch (Assumption A3), and these join events are roughly evenly-distributed over the epoch (Assumption A5). Consequently, each good ID will spend $O(1)$ to join the system.

In contrast, when there is a large attack, the entrance-cost function should impose a larger cost on the adversary. Consider the case where a batch of many bad IDs is rapidly injected into the system. This drives up the entrance cost since J_{cur} grows quickly, while $\tilde{J}_{i-1}$ remains fixed over the current epoch. Therefore, the adversary incurs a cost that is roughly quadratic in the number of bad IDs it injects.

How do the good IDs fare? A good ID that joins immediately after the batch of bad IDs will pay an entrance fee that is approximately equivalent to what the adversary paid for its last bad ID (rather than the whole batch). In particular, this good ID will pay a cost proportional to the square root of the adversary's cost. Furthermore, since join events by good IDs are roughly evenly distributed (Assumption A5), the cost for subsequent good IDs that join will quickly decrease.

These extreme two cases provide intuition for the asymmetric cost guaranteed by GMCom. Showing that this guarantee holds while interpolating between these cases, and incorporating multiple epochs and purge costs, is the subject of our analysis in Section III.

Estimating Symmetric Difference. To calculate the entrance cost in epoch i, GMCom requires knowledge of the good ID join rate from the previous epoch, J_{i-1}. However, since good IDs cannot be discerned from bad IDs upon entering the system, the adversary may inject bad IDs in an attempt to obscure the true join rate of good IDs.

Our analysis in the beginning of Subsection III-B addresses this challenge. In order to obtain a robust estimate $\tilde{J}_{i-1}$ of J_{i-1}, we leverage the adversary can provide solutions for at most an α-fraction of the puzzles issued during a purge.

How Puzzles Are Used. Although, each puzzle is constructed in the same manner, they are used in two distinct ways by our algorithm. First, when a new ID wishes to join the system, it must provide a solution for a ***entrance puzzle***.

The solution to the puzzle is $\boldsymbol{K_v}||s||\boldsymbol{\mathcal{T}}$, where $\boldsymbol{K_v}$ is the public key of v, s is a nonce selected buy v in order to solve the puzzle, and $\boldsymbol{\mathcal{T}}$ is the timestamp of when the puzzle solution

was generated. The value of $\mathcal{T}$ in the solution to an entrance puzzle must be within some small margin of the current time. In practice, this margin would primarily depend on network latency.

We note that, in the case of a bad ID, this solution may have been precomputed by the adversary by using a future timestamp. This is not a problem since the purpose of this puzzle is only to force the adversary to incur a computational cost at some point, and to deter the adversary from reusing puzzle solutions. Importantly, the entrance puzzle is not used to preserve guarantees on the fraction of good IDs in the system.

The second way in which puzzles are used is to limit the fraction of bad IDs in the system; this is referred to as a ***purge puzzle***, which has cost 1. An announcement is periodically made by the committee that *all* IDs already in the system should solve a purge puzzle. When this occurs, a ***random string*** $\boldsymbol{r}$ of $\Theta(\log n_0^\gamma)$ bits is generated by the committee and included as part of the announcement. The value r must also be appended to the inputs for all requested solutions in this round; that is, the input is $\boldsymbol{K_v||s||r}$. These random bits ensure that the adversary cannot engage in a pre-computation attack — where it solves puzzles and stores the solutions far in advance of launching an attack — by keeping the puzzles unpredictable. For ease of exposition, we omit further discussion of this issue and consider the use of these random bits as implicit whenever a purge puzzle is issued.

While the same r is used in the puzzle construction for all IDs, we emphasize that a *different* puzzle is assigned to each ID since the public key used in the construction is unique. Again, this is only of importance to the second way in which puzzles are used. Using the public key in the puzzle construction also prevents puzzle solutions from being stolen. That is, ID K_v cannot lay claim to a solution found by ID K_w since the solution is tied to the public key K_w.

Can a message m_v from ID K_v be spoofed? This is prevented in the following manner. ID K_v signs m_v with its private key to get $\texttt{sign}_v$, and then sends $(m_v||\texttt{sign}_v||K_v)$ via DIFFUSE. Any other ID can use K_v to check that the message was signed by the ID K_v and thus be assured that ID K_v is the sender. Note that, although we make use of public key cryptography, we do not need a public-key infrastructure. We point the reader to Section **??** for an example puzzle specification.

III. UPPER BOUNDS

In this section, we begin with arguing the correctness of GMCOM, followed by proof of our estimation $\tilde{J}_{i-1}$ to be "close" to J_{i-1}, and then finally, we prove Theorem 1. Throughout the section, we let log be the natural log function.

A. Correctness Proofs

In this section, we prove that with high probability GMCOM preserves the population goal and committee goal over the lifetime of the system. For any epoch i, we let $\boldsymbol{B_i}$ and $\boldsymbol{G_i}$ respectively denote the number of bad and good IDs in the system immediately after the purge at the end of epoch i, and we let $\boldsymbol{N_i = B_i + G_i}$. Due to the use of GENID for initializing the system (recall Subsection II-A),particular, $B_0 < (3/10)N_0$.

TABLE I: A summary of our notation. Symbols below the bold-faced line are used exclusively in the proofs.

Notation	Description
T	Adversarial spending rate.
J	Good IDs joining rate.
α	Fraction of total computational power with the adversary.
J_i	Join rate of good IDs during epoch i.
ℓ_i	Length of epoch i.
$\mathcal{S}_i$	Set of all IDs in the system at the end epoch i.
n_0	Minimum number of good IDs in the system at any time.
J_{cur}	Current number of join events in this epoch divided by current time elapsed in this epoch.
$\tilde{J}_i$	Estimated join rate of good IDs for epoch i.
$\mathcal{S}_{\text{cur}}$	Set of all IDs in the system in the current instant of time.
$\tilde{J}_{\text{cur}}$	Estimate of good join rate in current epoch.
B_i	Number of bad IDs in the system at the end of epoch i.
G_i	Number of good IDs in the system at the end of epoch i.
N_i	Number of IDs in the system at the end of epoch i.
n_i^a	Number of IDs that arrive over epoch i.
b_i^a	Number of bad IDs that arrive over epoch i.
g_i^a	Number of good IDs that arrive over epoch i.
n_i^d	Number of IDs that depart over epoch i.
b_i^d	Number of bad IDs that depart over epoch i.
g_i^d	Number of good IDs that depart over epoch i.
$\mathcal{T}_i$	Total computational cost to the adversary in epoch i.

To simplify our presentation, our claims are proved to hold with probability at least $1 - \tilde{O}(1/n_0^{\gamma+2})$, where $\tilde{O}$ hides a poly$(\log n_0)$ factor. Of course, we wish the claims of Theorem 1 to hold with probability at least $1 - 1/n_0^{\gamma+1}$ such that a union bound over n_0^γ joins and departures yields a w.h.p. guarantee. By providing this "slack" of an $\tilde{\Omega}(1/n_0)$-factor in each of the guarantees of this section, we demonstrate this is feasible while avoiding an analysis cluttered with specific settings for the constants used in our arguments.

Throughout, we assume $\mu = n_0^\ell$ for some constant $\ell \geq 1$, since μ is a polynomial in n_0 (recall Section **??**). The next lemma assumes bounds of $\alpha \leq 1/14$. Although a larger value of α may be tolerable, this bound is chosen in order to simplify the analysis and provide a clean presentation.

Lemma 3. *For all $i \geq 0$, $B_i < N_i/3$.*

Proof. This is true by assumption for $i = 0$. For $i > 0$, note that the adversary has computational power less than $G_i/2$. Thus, after Step 2 that ends epoch i, we have $B_i < G_i/2$. Adding $B_i/2$ to both sides of this inequality yields $(3/2)B_i < N_i/2$, from which, $B_i < N_i/3$. □

Let $\boldsymbol{n_i^a, g_i^a, b_i^a}$ denote the total, good, and bad IDs that arrive over epoch i. Similarly, let $\boldsymbol{n_i^d, g_i^d, b_i^d}$ denote the total, good, and bad IDs that depart over epoch i.

Note that the committee will always have an accurate value for all of these variables except for possibly b_i^d — recall from Section I-A that bad IDs do not need to give notification when

they depart — and, consequently, n_i^d; in these two cases, the committee may hold values which are underestimates of the true values.

Lemma 4. *The fraction of bad IDs is always at most* $1/2$.

Proof. Fix some epoch $i > 0$. Recall that an epoch ends when $|\mathcal{S}_{i-1} \otimes S_{cur}| \geq |\mathcal{S}_{i-1}|/3$ where $|\mathcal{S}_{i-1}| = N_{i-1}$. Therefore, we have $b_i^a + g_i^d \leq N_{i-1}/3$. We are interested in the maximum value of the ratio of bad IDs to total IDs at any point during the epoch. Thus, we pessimistically assume all additions of bad IDs and removals of good IDs come first in the epoch. We are then interested in the maximum value of the ratio:

$$\frac{B_{i-1} + b_i^a}{N_{i-1} + b_i^a - g_i^d}.$$

By Lemma 3, $B_{i-1} \leq N_{i-1}/3$. Thus, the we want to find the maximum of $\frac{N_{i-1}/3 + b_i^a}{N_{i-1} + b_i^a - g_i^d}$, subject to the constraint that $b_i^a + g_i^d \leq N_{i-1}/3$. This ratio is maximized when the constraint achieves equality, that is when $g_i^d = N_{i-1}/3 - b_i^a$. Plugging this back into the ratio, we get

$$\frac{N_{i-1}/3 + b_i^a}{N_{i-1} + b_i^a - g_i^d} \quad \leq \quad \frac{N_{i-1}/3 + b_i^a}{2N_{i-1}/3 + 2b_i^a} = 1/2$$

Therefore, the maximum fraction of bad IDs is $1/2$ at any point during any arbitrary epoch $i > 0$.

Finally, we note that this argument is valid even though $\mathcal{S}_{cur}$ may include bad IDs that have departed *without* notifying the committee (recall this is possible as stated in Section I-A). Intuitively, this is not a problem since such departures can only lower the fraction of bad IDs in the system; formally, the critical equation in the above argument is $b_i^a + g_i^d \leq N_{i-1}/3$, and this does not depend on b_i^d. □

Next, we prove the committee invariant.

Lemma 5. *There is always an honest majority in the committee with probability at least* $1 - O\left(n_0^{-(\gamma+1)}\right)$.

Proof. For epoch $i = 0$, committee goal holds true from the use of GENID to initialize the system (recall Subsection II-A; for details, see Lemma 6 of [6]).

Fix an epoch $i > 0$. Recall that a new committee is elected by the existing committee at the end of the current epoch by selecting $c \log |S_i|$ IDs independently and uniformly at random from the set S_i, and for a sufficiently large constant $c > 0$ which we define concretely later on in this argument. Then, let X_G be a random variable which denotes the number of good IDs elected to the new committee in epoch i. Then:

$$E[X_G] \quad = \quad \frac{|G_i|}{|S_i|} c \log |S_i| \; = \; (1-\alpha) c \log |S_i| \qquad (1)$$

where the last inequality follows from the fact that the computational power with the adversary is at most α. Next, we bound the number of good IDs in the committee using Chernoff Bound [58] as:

$$\begin{aligned} & Pr\left(X_G < (1-\delta)(1-\alpha)c \log |S_i|\right) \\ \leq \quad & \exp\left\{-\frac{\delta^2 (1-\alpha) c \log |S_i|}{2}\right\} \\ = \quad & \frac{1}{|S_i|^{(1-\alpha)c\delta^2/2}} \\ = \quad & O\left(n_0^{-(\gamma+1)}\right) \end{aligned}$$

where the first step holds for any constant $0 < \delta < 1$, the second step follows from Equation 1, and finally, the last step holds for all $c \geq \frac{28}{13} \frac{(\gamma+1)}{\delta^2}$. For $\delta = 1/100$, we obtain can bound the number of good IDs in the committee to be at least $9/10 c \log |S_i|$ with probability at least $1 - O(n_0^{-(\gamma+1)})$.

Let Y_g be a random variable which denotes the number of good IDs that depart from the committee when the number of departures of good IDs from the system is at most $|S_{i-1}|/3$. Then, since the departures of good IDs occurs independently and uniformly at random from the system (See Section I-A), we obtain:

$$E[Y_g] \quad \leq \quad \frac{|S_i|}{3}\left(\frac{c \log |S_i|}{|S_i|}\right) \quad = \quad \frac{c}{3} \log |S_i| \qquad (2)$$

Next, we obtain upper bound on the number of departures of good IDs from the committee using Chernoff Bound as:100

$$\begin{aligned} & Pr\left(Y_g > (1+\delta') \frac{c \log |S_i|}{3}\right) \\ \leq \quad & \exp\left\{-\frac{\delta'^2 c}{6} \log |S_i|\right\} \\ = \quad & \frac{1}{|S_i|^{\delta'^2 c/6}} \\ = \quad & O\left(n_0^{-(\gamma+1)}\right) \end{aligned}$$

where the first step holds for any constant $0 < \delta' < 1$ and the last step holds for all $c \geq \frac{6(\gamma+1)}{\delta'^2}$.

Thus, for $\delta' = 1/5$, with probability $1 - O(n_0^{-(\gamma+1)})$, the minimum number of good IDs in the committee is greater than $(9/10)c \log |S_i| - (4/10)c \log |S_i|$. Next, use a union bound over n_0^γ epochs to show that the committee goal invariant is maintained through out the lifetime of the system. □

B. Bounds on Estimation of Good ID Join Rate

In this section, we prove that the committee can obtain a good estimate of the join rate for good IDs.

Lemma 6. *For any epoch* $i \geq 1$, $J_i/12 \leq \tilde{J}_i \leq 3J_i$.

Proof. Fix an epoch $i \geq 1$. Note that $\tilde{J}_i$ equals:

$$\begin{aligned} \frac{|\mathcal{S}_{i-1} \otimes \mathcal{S}_i| - \alpha(|\mathcal{S}_{i-1}| + |\mathcal{S}_i|)}{\ell_i} \quad & \leq \quad \frac{|G_{i-1} \otimes G_i|}{\ell_i} \\ & \leq \quad \frac{3J_i \ell_i}{\ell_i} \\ & \leq \quad 3J_i \end{aligned}$$

Where the second step holds since $|G_{i-1} \otimes G_i|$ is no more than the number of join and leave events by good IDs in epoch i, which is at most ℓ_i times the rate of good joins plus the rate of good departures; and by Assumption A1, the sum of these two rates is at most $3J_i$.

Next, we show the lower bound. Observe that:

$$\begin{aligned}\tilde{J}_i &= \frac{|\mathcal{S}_{i-1} \otimes \mathcal{S}_i| - \alpha(|\mathcal{S}_{i-1}| + |\mathcal{S}_i|)}{\ell_i} \\ &\geq \frac{|\mathcal{S}_{i-1}|/3 - \alpha|\mathcal{S}_{i-1}| - \alpha|\mathcal{S}_i|}{\ell_i} \\ &\geq \frac{|\mathcal{S}_{i-1}|/3 - \alpha|\mathcal{S}_{i-1}| - \alpha(4/3)|\mathcal{S}_{i-1}|}{\ell_i} \\ &\geq \frac{(1/3 - (7/3)\alpha)|\mathcal{S}_{i-1}|}{\ell_i} \qquad (3)\end{aligned}$$

Where the second step holds by the condition that triggers a purge test. In the third step $|\mathcal{S}_i| \leq (4/3)|\mathcal{S}_{i-1}|$ holds by the bound on $|\mathcal{S}_i \otimes \mathcal{S}_{i-1}|$.

Also, by Assumption A4:

$$J_i \leq \frac{2|\mathcal{S}_{i-1}|}{\ell_i}. \qquad (4)$$

Substituting $\frac{|\mathcal{S}_{i-1}|}{\ell_i}$ from Eq. 4 into Eq 3, we have:

$$\begin{aligned}\tilde{J}_i &\geq (1/3 - (7/3)\alpha)(J_i/2) \\ &\geq J_i/12\end{aligned}$$

where the last step holds for $\alpha \leq \frac{1}{14}$. □

Lemma 7. *For any epoch $i \geq 2$, $\frac{1}{24}J_i \leq \tilde{J}_{i-1} \leq 6J_i$*

Proof. Fix an epoch $i \geq 2$. For the upper bound, observe:

$$\begin{aligned}\tilde{J}_{i-1} &\leq 3J_{i-1} && \text{by Lemma 6} \\ &\leq 6J_i && \text{by Assumption A3 in Section I-A.}\end{aligned}$$

Similarly, we can obtain the lower bound:

$$\begin{aligned}\tilde{J}_{i-1} &\geq \frac{J_{i-1}}{12} && \text{by Lemma 6} \\ &\geq \frac{J_i}{24} && \text{by Assumption A3 in Section I-A}\end{aligned}$$

which completes the proof. □

C. Cost Analysis

Let $\mathcal{T}_i$ denote the computational cost to the adversary incurred during epoch i.

Each epoch i is divided into ***sub-epochs***. Sub-epoch $j \geq 1$ begins when $(j-1)/J_i$ time has elapsed in the epoch and ends when j/J_i time has elapsed. $\mathcal{T}_i^j$ is the computational cost paid by the adversary from the beginning of epoch i until the end of sub-epoch j of epoch i.

Let b_i^j be the number of bad IDs that have joined from the beginning of epoch i until the end of sub-epoch j of epoch i. Finally, let b_i be the number of bad IDs that join in epoch i.

Lemma 8. *For any sub-epoch $j \geq 1$ in any epoch $i \geq 2$, $b_i^j \leq \sqrt{12\, j\, \mathcal{T}_i^j}$.*

Proof. The j^{th} sub-epoch ends at time $t_j = j/J_i$. So the entrance cost for the k^{th} bad ID joining in sub-epoch j is at least:

$$\begin{aligned}\max\left\{\frac{k/t_j}{\tilde{J}_{i-1}}, 1\right\} &\geq \frac{k}{\tilde{J}_{i-1}\, t_j} \\ &\geq \frac{k}{6\, J_i\, t_j} && \text{By Lemma 7} \\ &\geq \frac{k}{6\, j}\end{aligned}$$

Thus we have:

$$\begin{aligned}\mathcal{T}_i^j &\geq \sum_{k=1}^{b_i^j} \frac{k}{6\, j} \\ &\geq \frac{(b_i^j)^2}{12\, j}\end{aligned}$$

Solving for b_i^j in this inequality completes the proof. □

We use the following fact in the next lemma and the proof of Theorem 1.

Fact 9. *Suppose that u and v are x-dimensional vectors in Euclidean space. For all $x \geq 1$, $\sum_{j=1}^{x} \sqrt{u_j v_j} \leq \sqrt{\left(\sum_{j=1}^{x} u_j\right)\left(\sum_{j=1}^{x} v_j\right)}$.*

Proof. Using the Cauchy-Schwarz inequality, we have:

$$\left(\sum_{j=1}^{n} \sqrt{u_j v_j}\right)^2 \leq \left(\sum_{j=1}^{n} u_j\right)\left(\sum_{j=1}^{n} v_j\right)$$

Taking the square-root of both sides, we get:

$$\sum_{j=1}^{x} \sqrt{u_j v_j} \leq \sqrt{\left(\sum_{j=1}^{x} u_j\right)\left(\sum_{j=1}^{x} v_j\right)}$$

which completes the argument. □

Lemma 10. *For any epoch $i \geq 2$, the total entrance cost paid by the good IDs over the entire epoch is $O\left(\sqrt{\mathcal{T}_i J_i\, \ell_i} + J_i\, \ell_i\right)$.*

Proof. Fix an epoch $i \geq 2$ and let $j \geq 1$. Let the j^{th} sub-epoch end at time $t_j = j/J_i$. By Assumption A5, the entrance cost paid by a good ID in sub-epoch j is at most:

$$\begin{aligned}&\max\left\{\frac{(b_i^j + Cj)/t_{j-1}}{\tilde{J}_{i-1}}, 1\right\} \\ &\leq 1 + \left(\sqrt{12\, j\, \mathcal{T}_i^j} + Cj\right)/(\tilde{J}_{i-1}\, t_{j-1}) \\ &\leq 1 + 24\left(\sqrt{12\, j\, \mathcal{T}_i^j} + Cj\right)/(J_i\, t_{j-1}) \\ &\leq 1 + 24\, C\left(\sqrt{12\, j\, \mathcal{T}_i^j} + j\right)/(j-1)\end{aligned}$$

where the second line follows from Lemma 8, the third line follows from Lemma 7, and the fourth line follows since $t_{j-1} = (j-1)/J_i$.

Summing over all sub-epochs in epoch i, and using Assumption A5, the total entrance cost paid by the good IDs is at most:

$$\begin{aligned}&\sum_{j=1}^{J_i\,\ell_i} C\left(1+24\,C\left(\sqrt{12\,j\,\mathcal{T}_i^j}+j\right)/(j-1)\right)\\&=\sum_{j=1}^{J_i\,\ell_i} O\left(\sqrt{\mathcal{T}_i^j/j}+1\right)\\&=O\left(\sqrt{\mathcal{T}_i J_i\,\ell_i}+J_i\,\ell_i\right)\end{aligned}$$

as desired since $\sum_{j=1}^{J_i\,\ell_i}\sqrt{\mathcal{T}_i^j/j}=O(\sqrt{\mathcal{T}_i J_i\,\ell_i})$ follows by Fact 9. □

Lemma 11. *For any epoch $i\geq 2$, $|\mathcal{S}_{i-1}|\leq 8\sqrt{12\,\mathcal{T}_i\,J_i\,\ell_i}+10J_i\,\ell_i$.*

Proof. Recall that the number of bad IDs that remain in the system at the end of epoch $i-1$ is at most $\alpha|\mathcal{S}_{i-1}|$ and that the number of bad IDs that enter in epoch i is b_i. Therefore, the number of join and leave events in epoch i due to bad IDs is at most $2b_i+\alpha|\mathcal{S}_{i-1}|$.

The departure rate of good IDs is at most $(3/2)J_i$ by Assumption 1, so the number of good IDs that join or depart in an epoch is at most $(5/2)J_i\ell_i$. Thus, the total join and leave events in epoch i is at most $2b_i+\alpha|\mathcal{S}_{i-1}|+(5/2)J_i\,\ell_i$.

From Step 2 of our algorithm, $|\mathcal{S}_{i-1}\otimes\mathcal{S}_i|=|\mathcal{S}_{i-1}|/3$. Therefore:

$$\frac{|\mathcal{S}_{i-1}|}{3}\leq 2b_i+\alpha|\mathcal{S}_{i-1}|+(5/2)J_i\,\ell_i.$$

Note that $j^*=J_i\,\ell_i$ is the last sub-epoch of epoch i. Hence by Lemma 8, $b_i=b_i^{j^*}=\sqrt{12\,J_i\,\ell_i\,\mathcal{T}_i}$. Solving for $|\mathcal{S}_{i-1}|$:

$$\begin{aligned}|\mathcal{S}_{i-1}|&\leq\frac{2b_i+(5/2)J_i\,\ell_i}{1/3-\alpha}\\&\leq 4(2b_i+(5/2)J_i\,\ell_i)\\&\leq 4\left(2\sqrt{12\,J_i\,\ell_i\,\mathcal{T}_i}+(5/2)J_i\,\ell_i\right)\\&\leq 8\sqrt{12\,J_i\,\ell_i\,\mathcal{T}_i}+10J_i\,\ell_i\end{aligned}$$

where the second line follows from $\alpha\leq 1/14$. □

Lemma 12. *For any epoch $i\geq 2$, the total purge computational cost to good IDs is $O(\sqrt{\mathcal{T}_i\,J_i\,\ell_i}+J_i\,\ell_i)$.*

Proof. We know that the number of good IDs that solve the purge puzzle is at most $|\mathcal{S}_i|$. Thus, the total purge computational cost to good IDs in epoch i is at most:

$$\begin{aligned}|\mathcal{S}_i|&\leq\frac{4}{3}|\mathcal{S}_{i-1}|\\&\leq(4/3)(8\sqrt{12\,\mathcal{T}_i\,J_i\,\ell_i}+10J_i\,\ell_i)\\&<11\sqrt{12\,\mathcal{T}_i\,J_i\,\ell_i}+14J_i\,\ell_i\end{aligned}$$

Where the first step follows since $|\mathcal{S}_{i-1}\otimes\mathcal{S}_i|=|\mathcal{S}_{i-1}|/3$, and the second step follows by Lemma 11. □

D. Proof of Theorem 1

Finally, we are ready to prove Theorem 1.

Proof. **Correctness.** Follows from Lemma 4 and Lemma 5.

Spending Rate. Consider any subset, E of epochs of all the epochs during the lifetime of the system. By Lemmas 10 and 12, the total computational cost paid by good IDs in all epochs in E is:

$$\begin{aligned}&\sum_{i\in E} O\left(\sqrt{\mathcal{T}_i J_i\,\ell_i}+J_i\,\ell_i\right)\\&=\;O\left(\sqrt{\sum_{i\in E}\mathcal{T}_i\sum_{i\in E}J_i\ell_i}+\sum_{i\in E}J_i\ell_i\right)\end{aligned}$$

Which follows from Fact 9 setting $u_i=T_i$ and $v_i=J_i\ell_i$ for $i\in E$. To calculate the average computational cost per round, we divide by $\sum_{i\in E}\ell_i$ to obtain:

$$\begin{aligned}&O\left(\frac{\sqrt{\sum_{i\in E}\mathcal{T}_i\sum_{i\in E}J_i\ell_i}+\sum_{i\in E}J_i\ell_i}{\sum_{i\in E}\ell_i}\right)\\=\;&O\left(\sqrt{\left(\frac{\sum_{i\in E}\mathcal{T}_i}{\sum_{i\in E}\ell_i}\right)\left(\frac{\sum_{i\in E}J_i\ell_i}{\sum_{i\in E}\ell_i}\right)}+\frac{\sum_{i\in E}J_i\ell_i}{\sum_{i\in E}\ell_i}\right)\\=\;&O\left(\sqrt{T\,J}+J\right)\end{aligned}$$

The last step follows since over all epochs in E, we have $T=\left(\sum_{i\in E}\mathcal{T}_i\right)/\left(\sum_{i\in E}\ell_i\right)$ and $J=\left(\sum_{i\in E}J_i\ell_i\right)/\left(\sum_{i\in E}\ell_i\right)$. □

IV. Lower Bounds

In this section, we provide a lower bound that applies to the class of algorithms which have the following attributes:

- **B1.** Each new ID must pay an entrance fee in order to join the system and this is defined by a ***cost function f***, which takes as input a join rate.
- **B2.** The algorithm executes over epochs (recall Section I-A), but we consider more general epochs that are delineated when the condition $|\mathcal{S}_i\otimes\mathcal{S}_{\text{cur}}|\geq\delta\,|\mathcal{S}_i|$ holds, for any positive $\delta<1/2$ (recall, GMCom uses $\delta=1/3$).
- **B3.** At the end of each epoch, each ID must pay $\Omega(1)$ to remain in the system.

We emphasize that B1 captures any cost function where the change in join cost during an epoch varies only with the join rate during that epoch. GMCom's cost function has this property, since $\tilde{J}_{i-1}$ is fixed throughout epoch i. With regard to B2 and B3, recall that we wish to preserve the population invariant (i.e., a majority of good IDs). It is hard to imagine an algorithm that preserves this invariant without a (likely simultaneous) computational test being imposed on all IDs.

A. Lower-Bound Analysis

Restating in terms of the conditions above, we have:

Theorem 2. *Suppose our algorithm satisfies conditions B1-B3, then there exists an adversarial strategy that forces $G=\Omega(\sqrt{T\,J}+J)$.*

Proof. Fix an epoch i. Let n be the number of IDs in the system at the beginning of epoch i. Let ρ be any adversarial join rate during this epoch. We show that the adversary can always force the spending rate of the good nodes to be $\Omega(\sqrt{T\,J})$, where T is the spending rate of the adversary in epoch i, and J is the join rate of good IDs in epoch i. The adversarial strategy is for bad IDs to join uniformly at a rate of ρ during the epoch, and then drop out right before the purge.

We first calculate the algorithmic spending rate due to purge puzzle costs in epoch i. We use the fact that an epoch ends when the system sees a δ-factor symmetric difference, for $\delta < 1/2$, and that the average rate of joins in epoch i is $\Theta(\rho + J)$. Hence, by Attribute B2, we calculate the length of epoch i as:

$$\ell_i \quad = \quad O(n/(J+\rho))$$

By Assumption B3, each good ID pays a purge cost of $\Omega(1)$ at the end of each epoch. Hence the average spending rate due to purge costs in epoch i is $\Omega(n/\ell_i)$ which is:

$$\Omega\left(\frac{n}{n/(J+\rho)}\right) \quad = \quad \Omega(J+\rho) \tag{5}$$

We now have two cases:

Case 1: $f(\rho + J) > \rho/J$.
The overall rate of joins and departures of IDs is $\rho + J$ during this epoch. The adversarial join rate is ρ , so we have:

$$T \quad = \quad \rho f(\rho + J).$$

Rearranging we get:

$$f(\rho + J) \quad = \quad T/\rho. \tag{6}$$

Since $f(\rho + J) > \rho/J$, we have:

$$\begin{aligned} \rho \quad &\leq \quad J\, f(\rho + J) \\ &= \quad JT/\rho \quad \text{by Equation 6.} \end{aligned}$$

On solving the above for ρ, we get:

$$\rho < \sqrt{T\,J}. \tag{7}$$

By assumption $A1$, the good ID join rate is $\Theta(J)$. Thus, the spending rate for the algorithm due to entrance costs is $\Omega(J\, f(\rho + J))$. Adding in the spending rate for purge costs of $\Omega(J + \rho)$ from Equation 5, we get that the average cost to the algorithm is:

$$\begin{aligned} G \quad &= \quad \Omega(J\, f(\rho + J) + (J + \rho)) \\ &= \quad \Omega(J\, T/\rho + (J + \rho)) \qquad \text{by Equation 6} \\ &= \quad \Omega\left(\sqrt{T\,J} + J\right) \qquad \text{by Equation 7} \end{aligned}$$

Case 2: $f(\rho + J) \leq \rho/J$.
In this case, we have:

$$T \quad \leq \quad \rho\, f(\rho + J) \leq \rho^2/J$$

Rearranging, we get:

$$\rho \quad \geq \quad \sqrt{T\,J} \tag{8}$$

By Equation 5, we have that the algorithmic spending rate due to purge costs is $\Omega(J + \rho)$. Thus:

$$\begin{aligned} G \quad &= \quad \Omega(\rho + J) \\ &= \quad \Omega\left(\sqrt{T\,J} + J\right) \qquad \text{by Equation 8.} \end{aligned}$$

□

V. Experiments

We simulate GMCom to evaluate its performance with respect to the computational cost incurred from solving puzzles. Given this goal, we do not model the execution of Byzantine consensus or committee formation. In Subsection V-B, we validate the asymptotic behavior of the asymmetric spending rate. Second, in Subsection V-A, we compare GMCom with two PoW-based Sybil defenses.

Our simulation code is written in MATLAB and run on an Mac machine with High Sierra (v. 10.13.6) using a 1.7 GHz Intel Core i5 processor and 4 GB of 1333 MHz DDR3 RAM.

A. Validating Asymptotics

We simulate GMCom to validate that it exhibits the asymmetric spending rate $O(\sqrt{T\,J} + J)$. The system is initialized with $10,000$ good IDs, and J is set to 2 IDs per second. The number of good IDs remains fixed throughout the simulation, thus 2 good IDs depart every second. We denote the average cost to the good IDs by $\boldsymbol{G}$.

Regarding the adversary, we assume $\alpha = 1/14$, and T ranges over $[2, 2^{100}]$, where for each such value of T, the system is simulated for $10,000$ seconds. The adversary solves entrance puzzles to add bad IDs to the system. We pessimistically ignore the cost paid by the adversary for solving purge puzzles.

Figure 2 illustrates our results. The blue line depicts the average computational cost to good IDs per second obtained by executing GMCom when the adversary spends T per second. The green and red line are the plots for $G = T$ and $G = \sqrt{T}$.

Note that the x-axis and y-axis are both $\log$ scaled. Initially, the blue line increases slowly due to T not being substantially larger than J, and hidden constants. However, as T grows, we observe behavior very close to $G = \sqrt{T}$, which validates the asymptotic behavior of the asymmetric cost.

B. Comparison with Existing PoW-Based Sybil Defenses

To evaluate GMCom against other Sybil defenses that use PoW, we implement and evaluate two contemporary algorithms, Ccom and SybilControl.

Overview of Ccom. This algorithm is a precursor to GMCom, where the entrance puzzle always has a computational cost of 1. Apart from this, Ccom is identical to GMCom.

Overview of SybilControl. Under this algorithm, each ID must solve a puzzle to join the system. Additionally, each ID periodically tests its neighbors with a puzzle, removing from its list of neighbors those IDs that fail to provide a solution within a time limit; these tests are not coordinated between IDs. An ID may be a neighbor to more than one ID and

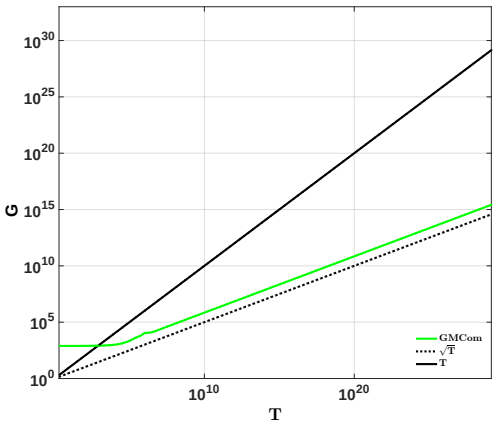

Fig. 2: Average cost for GMCOM vs. average cost to adversary.

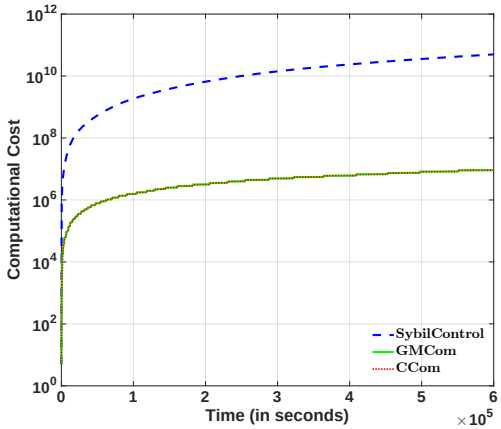

Fig. 3: Cumulative cost for algorithms in Scenario I.

so receive multiple puzzles; these are combined into a single puzzle whose solution satisfies all the received puzzles.

In our experiments, we assume a computational cost of k for solving a puzzle of difficulty k.

1) The Bitcoin Network: We simulate CCOM, SYBILCONTROL, and GMCOM on a real-world dataset for the Bitcoin network [63] consisting of roughly 7 days of join/departure-event timestamps.[2] The computational cost is examined under Scenario I when there are no bad IDs (i.e., no attack), and Scenario II when bad IDs are present (i.e., an attack).

In Scenario I, all events in the dataset are treated as good IDs joining/departing. Figure 3 depicts the cumulative computational cost to the good IDs for the algorithms. Importantly, in comparison to SYBILCONTROL, the cumulative cost to the good IDs under GMCOM and CCOM is less by roughly 4 orders of magnitude after 13 hours of simulation time, and this gap continues to widen with time; note the logarithmic y-axis. This result demonstrates that GMCOM and CCOM — which perform identically when there is no attack — is more efficient than SYBILCONTROL when there are no bad IDs.

In Scenario II, joins/departures occur as in Scenario I, but the following attack also occurs. From time $t/6$ to $t/3$ seconds, where $t = 604,970$ (≈ 7 days spanned by the dataset), every 20 seconds, the adversary adds a number of bad IDs that is a $1/3$-fraction of the current system size. This forces a purge test every 20 seconds in CCOM and GMCOM, while an ID issues a puzzle every 5 seconds in SYBILCONTROL.

Figure 4(a) depicts the cumulative cost to the good IDs for Scenario II. Notably, the overall cost of SYBILCONTROL is much higher than that of CCOM and GMCOM before the system is attacked. When the attack commences, the cumulative cost to SYBILCONTROL becomes comparable to that of the other two algorithms, but then quickly surpasses them, and continues to grow after the attack ends. We observe that the computational costs to good IDs for CCOM and GMCOM are always comparable.

Finally, Figure 4(b) illustrates the ratio of algorithmic cost to that of the adversary for Scenario II. Notably, GMCOM has a cost ratio that is roughly 3 orders of magnitude smaller than either SYBILCONTROL or CCOM. This can be attributed to the asymmetric cost that benefits good IDs under GMCOM, but which is not guaranteed by either CCOM or SYBILCONTROL.

2) Peer-to-Peer Networks: We compare the performance of SYBILCONTROL, CCOM and GMCOM on three different peer-to-peer (P2P) networks: BitTorrent, Skype and Bitcoin.

Our BitTorrent experiments employ data from two BitTorrent servers: Debian and Flatout. The Weibull distribution is used to model session time, with shape parameter values 0.38 and 0.59, and scale parameter values 42.2 and 41.0 for Debian and FlatOut, respectively. In our Skype experiments, we assume a Weibull distribution for the session time of supernodes, with a median session time of 5.5 hours, and a shape parameter of 0.64. For our Bitcoin experiments, we generate the session time for $10,000$ good IDs by randomly sampling from the real-world data obtained from [63].

For each value of $T \in \{2^0, 2^1, ..., 2^{16}\}$, at each step in the simulation, the adversary adds the maximum number of bad IDs that its budget allows. Our plots are generated from our Monte-Carlo simulations, where each value was averaged over 20 separate simulations.

Figures 4(c) - (f) illustrate the computational cost to good IDs as the computational budget of the adversary per second varies. We observe that GMCOM demonstrates significant performance gains over both CCOM and SYBILCONTROL; again, note the logarithmic y-axis.

VI. FURTHER DISCUSSION ON PUZZLES

All IDs have access to a hash function, $\boldsymbol{h}$, about which we make the standard *random oracle assumption* [13], [21], [50]. Succinctly, this assumption is that when first computed on an input, x, $h(x)$ is selected independently and uniformly at random from the output domain, and that on subsequent computations of $h(x)$ the same output value is always returned. We assume that both the input and output domains are the real numbers between 0 and 1. In practice, h may be a cryptographic hash function, such as SHA-2 [66], with inputs and outputs of sufficiently large bit lengths.

In general, an ID must find an input x such that $h(x)$ is less than some threshold. Decreasing this threshold value will increase the difficulty, since one must compute the hash

[2]Data obtained by personal correspondence with Till Neudecker [63].

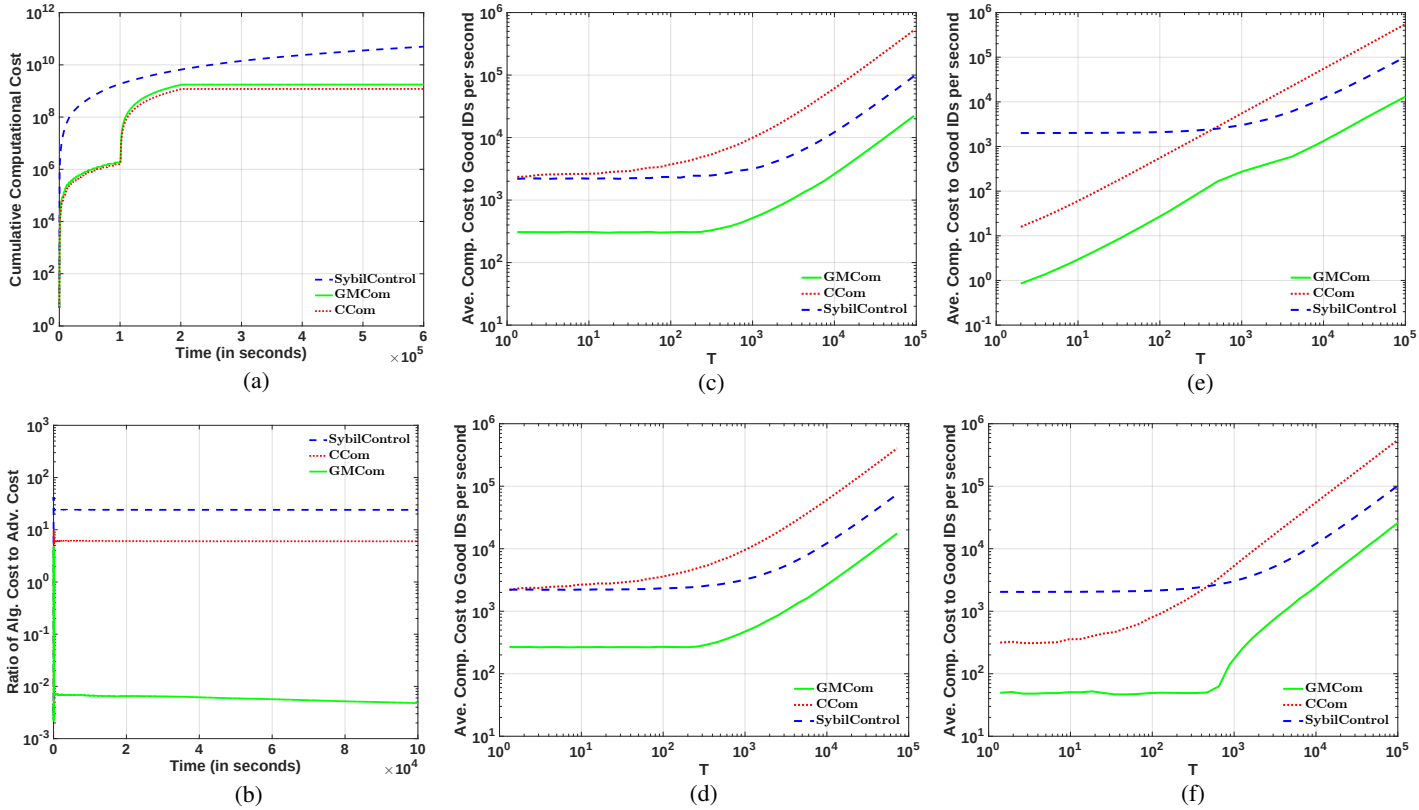

Fig. 4: Bitcoin experiments: (a) Cumulative computational cost and (b) ratio of algorithmic cost to adversarial cost for Scenario II. Average computational cost to good IDs per second vs. adversarial computational cost per second for the following peer-to-peer networks: (c) BitTorrent Debian, (d) BitTorrent Flatout, (e) Skype, and (f) Bitcoin.

function on more inputs to find an output that is sufficiently small. We note that many other types of puzzles exist (for examples, see [46], [48], [70]) and our results are likely compatible with other designs.

We assume that each good ID can perform μ hash-function evaluations per round for $\mu > 0$. Additionally, we assume that μ is of some size polynomial in n_0 so that $\log \mu = \Theta(\log n_0)$. It is reasonable to assume large μ since, in practice, the number of evaluations that can be performed per second is on the order of millions to low billions [15], [16], [44].

For any integer $\rho \geq 1$, we define a ***ρ-round puzzle*** to consist of finding $\ell = C \log \mu$ solutions, each of difficulty $\tau = \rho(1-\delta)\mu/(C \log \mu)$, where $\delta > 0$ is a small constant and C is a sufficiently large constant depending on δ and μ.

Let X be a random variable giving the expected number of hash evaluations needed to compute ℓ solutions. Then X is a negative binomial random variable and we have the following concentration bound (see, for example, Lemma 2.2 in [10]).

For every $0 < \epsilon \leq 1$, it holds that

$$Pr(|X - E(X)| \geq \epsilon E(X)) \leq 2e^{-\epsilon^2 \ell/(2(1+\epsilon))}$$

Given the above, we can show that every good ID will solve a ρ-round puzzle with at most $\rho\mu$ hash function evaluations, and that the adversary must compute at least $(1-2\delta)d\rho\mu$ hash evaluations to solve every ρ-round puzzle. This follows from a union bound over $O(n_0^\gamma)$ joins and departure events, for C sufficiently large as a function of δ and γ. Note that for small δ, the difference in computational cost is negligible, and that μ is also unnecessary in comparing costs. Thus, for ease of exposition, **we assume that each ρ-round puzzle requires computational cost ρ to solve**.

Finally, each participant v uses a ***public key K_v***, generated via public key encryption, as its ID. The input to a puzzle always incorporates K_v and s, where the ***solution string s*** is selected by v (for good IDs, s will be a random string).

A. *Space efficient estimation of symmetric difference*

The amount of churn experienced by the system is estimated through the use of two sample sets - S'_{i-1} and S'_{cur}. These sets are generated by the committee using a hash function $\boldsymbol{h'}$ computed using secure multi-party computation (MPC) such as in [8], [12], [18], [26], [27], [29], [40]. Note that h' is unknown to the IDs and thus, the adversary cannot precompute the sample sets. The committee adds an ID, say v to the sample set if the hash of its ID $h'(v) \leq \frac{c' \log n_0}{|S_{i-1}|}$, for some constant $c' > 0$. A fresh sample set, $\boldsymbol{S'_{i-1}}$, is generated at the beginning of every epoch. As new IDs join the system, they are sampled by the committee using h' and added to the current sample set, $\boldsymbol{S'_{cur}}$.

Initially, the committee knows the existing membership denoted by S_{cur}, which has initially n_0 IDs. The committee generates S'_{i-1} and updates the current sample set, S'_{cur}

as described above. At any time during the execution, if $|(S'_{cur} \otimes S'_{i-1})|$ exceeds $S'_{i-1}/4$, the system undergoes a purge test. Note that this fraction is different from that mentioned in Step 2 of GMCom. As before, the committee broadcasts r to conduct purge puzzle. The IDs that fail to submit valid purge puzzle solutions are de-registered — that is, they are effectively removed from the system.

VII. Conclusion and Future Work

We have presented an asymmetric algorithm that defends against Sybil attacks using PoW. Our algorithm has a computational spending rate of $O(\sqrt{T\,J} + J)$, where T is the rate of adversarial computational spending, and J is the rate of joining good IDs. We have provided empirical evidence that our algorithm is as efficient as state-of-the-art Sybil defenses in all cases, and is significantly more efficient under massive attacks. Finally, we have proved a lower bound showing that our algorithm's computational cost is asymptotically optimal among a large class of Sybil-defense algorithms.

Many open problems remain including the following. Can we extend our results to create an asymmetric algorithm for maintaining a secure blockchain? Might the techniques used here be adaptable for use in mitigating other security challenges such as denial-of-service attacks?

References

[1] I. Abraham and D. Malkhi. The Blockchain Consensus Layer and BFT. *Bulletin of the EATCS: Distributed Computing Column*, 2017.

[2] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. FARSITE: Federated, Available, and Reliable Storage for Incompletely Trusted Environment. In 5^{th} USENIX *Symposium on Operating Systems Design and Implementation (OSDI)*, pages 1–14, 2002.

[3] Alyssa Hertig. Ethereum's Big Switch: The New Roadmap to Proof-of-Stake. www.coindesk.com/ethereums-big-switch-the-new-roadmap-to-proof-of-stake/.

[4] Amazon. Amazon Web Service (AWS). https://aws.amazon.com/.

[5] R. Anderson, C. Barton, R. Böhme, R. Clayton, M. J. Van Eeten, M. Levi, T. Moore, and S. Savage. Measuring the Cost of Cybercrime. In *The Economics of Information Security and Privacy*, pages 265–300. Springer, 2013.

[6] M. Andrychowicz and S. Dziembowski. Distributed cryptography based on the proofs of work. *IACR Cryptology ePrint Archive*, 2014:796, 2014.

[7] M. Andrychowicz and S. Dziembowski. Pow-Based Distributed Cryptography with No Trusted Setup. In *Annual Cryptology Conference*, pages 379–399. Springer, 2015.

[8] B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. *Automata, languages and Programming*, pages 152–163, 2010.

[9] J. Aspnes, C. Jackson, and A. Krishnamurthy. Exposing Computationally-Challenged Byzantine Impostors. Technical report, Tech. Report YALEU/DCS/TR-1332, Yale University http://www.cs.yale.edu/homes/aspnes/papers/tr1332.pdf, 2005.

[10] B. Awerbuch and C. Scheideler. Towards a Scalable and Robust DHT. In *Proceedings of the 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 318–327, 2006.

[11] R. A. Bazzi and G. Konjevod. On the Establishment of Distinct Identities in Overlay Networks. In *Proceedings* 24^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 312–320, 2005.

[12] Z. Beerliová-Trubıniová and M. Hirt. Efficient multi-party computation with dispute control. In *TCC*, volume 3876, pages 305–328. Springer, 2006.

[13] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS)*, pages 62–73. ACM, 1993.

[14] BitcoinWiki. BitcoinWiki Network. https://en.bitcoin.it/wiki/Network#Standard_relaying.

[15] BitcoinWiki. Mining Hardware Comparison, 2016. https://en.bitcoin.it/wiki/Mining_hardware_comparison.

[16] BitcoinWiki. Non-Specialized Hardware Comparison, 2016. https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison.

[17] Bitnodes. Bitnodes is currently being developed to estimate the size of the Bitcoin network by finding all the reachable nodes in the network., 2018. bitnodes.earn.com/dashboard/.

[18] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, et al. Secure multiparty computation goes live. In *Financial Cryptography*, volume 5628, pages 325–343. Springer, 2009.

[19] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pages 104–121, 2015.

[20] C. Cachin and J. A. Poritz. Secure Intrusion-Tolerant Replication on the Internet. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, pages 167–176, 2002.

[21] R. Canetti. Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information. In *Proceedings of the Annual International Cryptology Conference*, pages 455–469. Springer, 1997.

[22] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. *Operating Systems Review*, 33:173–186, 1998.

[23] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[24] A. Clement, M. Marchetti, E. Wong, L. Alvisi, and M. Dahlin. Byzantine Fault Tolerance: The Time is Now. In *Proceedings of the Second Workshop on Large-Scale Distributed Systems and Middleware (LADIS)*, pages 1–4, 2008.

[25] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. In *Proceedings of the Sixth USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 153–168, 2009.

[26] I. Damgard and Y. Ishai. Scalable secure multiparty computation. In *Crypto*, volume 4117, pages 501–520. Springer, 2006.

[27] I. Damgård, Y. Ishai, M. Krøigaard, J. Nielsen, and A. Smith. Scalable multiparty computation with nearly optimal work and resilience. *Advances in Cryptology–CRYPTO 2008*, pages 241–261, 2008.

[28] G. Danezis, C. Lesniewski-laas, M. F. Kaashoek, and R. Anderson. Sybil-Resistant DHT Routing. In *Proceedings of the* 10^{th} *European Symposium On Research In Computer Security (ESORICS)*, pages 305–318, 2005.

[29] V. Dani, V. King, M. Movahedi, and J. Saia. Quorums quicken queries: Efficient asynchronous secure multiparty computation. In *International Conference on Distributed Computing and Networking*, pages 242–256. Springer, 2014.

[30] M. Demirbas and Y. Song. An RSSI-based Scheme for Sybil Attack Detection in Wireless Sensor Networks. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks (WOWMOM)*, pages 564–570, 2006.

[31] J. Dinger and H. Hartenstein. Defending the Sybil Attack in P2P Networks: Taxonomy, Challenges, and a Proposal for Self-Registration. In *Proceedings of the First International Conference on Availability, Reliability and Security (ARES)*, pages 756–763, 2006.

[32] J. Douceur. The Sybil Attack. In *Proceedings of the Second International Peer-to-Peer Symposium (IPTPS)*, pages 251–260, 2002.

[33] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse. Bitcoin-NG: A Scalable Blockchain Protocol. In *Proceedings of the* 13^{th} *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 45–59, 2016.

[34] J. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. In *Proceedings of 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 281–310, 2015.

[35] S. Gil, S. Kumar, M. Mazumder, D. Katabi, and D. Rus. Guaranteeing Spoof-Resilient Multi-Robot Networks. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[36] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, pages 51–68, 2017.

[37] S. Gilbert, C. Newport, and C. Zheng. Who Are You? Secure Identities in Ad Hoc Networks. In *Proceedings of the* 28^{th} *International Symposium on Distributed Computing (DISC)*, pages 227–242, 2014.

[38] S. Gilbert and C. Zheng. SybilCast: Broadcast on the Open Airwaves. In *Proceedings of the* 25^{th} *Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 130–139, 2013.

[39] G. Golan-Gueta, I. Abraham, S. Grossman, D. Malkhi, B. Pinkas, M. K. Reiter, D.-A. Seredinschi, O. Tamir, and A. Tomescu. Sbft: a scalable decentralized trust infrastructure for blockchains. *CoRR*, abs/1804.01626, 2018.

[40] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, pages 86–97, 1998.

[41] S. Gorbunov and S. Micali. Democoin: A Publicly Verifiable and Jointly Serviced Cryptocurrency. Cryptology ePrint Archive, Report 2015/521, 2015. http://eprint.iacr.org/2015/521.

[42] D. Gupta, J. Saia, and M. Young. Proof of Work Without All the Work. *arXiv preprint arXiv:1708.01285*, 2017.

[43] D. Gupta, J. Saia, and M. Young. Proof of Work Without All the Work. In *Proceedings of the* 19^{th} *International Conference on Distributed Computing and Networking (ICDCN)*, 2018.

[44] Hashcat. Benchmark Results, 2016. http://thepasswordproject.com/oclhashcat_benchmarking.

[45] Intel. PoET 1.0 Specification, 2018. sawtooth.hyperledger.org/docs/core/releases/1.0/architecture/poet.html.

[46] Y. I. Jerschow and M. Mauve. Modular Square Root Puzzles: Design of Non-Parallelizable and Non-Interactive Client Puzzles. *Computers and Security*, 35:25–36, June 2013.

[47] R. John, J. P. Cherian, and J. J. Kizhakkethottam. A Survey of Techniques to Prevent Sybil Attacks. In *Proc. of the Intl. Conference on Soft-Computing and Networks Security (ICSNS)*, pages 1–6, 2015.

[48] G. O. Karame and S. Čapkun. Low-Cost Client Puzzles Based on Modular Exponentiation. In *Proceedings of the 15th European Conference on Research in Computer Security (ESORICS)*, pages 679–697, 2010.

[49] V. King, J. Saia, V. Sanwalani, and E. Vee. Scalable Leader Election. In *Proceedings of the* 17^{th} *Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, pages 990–999, 2006.

[50] N. Koblitz and A. J. Menezes. The Random Oracle Model: A Twenty-Year Retrospective. *Designs, Codes and Cryptography*, 77(2-3):587–610, 2015.

[51] 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 *25th USENIX Security Symposium (USENIX Security 16)*, pages 279–296, Austin, TX, 2016. USENIX Association.

[52] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: Speculative Byzantine Fault Tolerance. In *Proceedings of* 21^{st} *Symposium on Operating Systems Principles*, pages 45–58, 2007.

[53] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[54] F. Li, P. Mittal, M. Caesar, and N. Borisov. SybilControl: Practical Sybil Defense with Computational Puzzles. In *Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing*, pages 67–78, 2012.

[55] Y. Liu, D. R. Bild, R. P. Dick, Z. M. Mao, and D. S. Wallach. The Mason Test: A Defense Against Sybil Attacks in Wireless Networks Without Trusted Authorities. *IEEE Transactions on Mobile Computing*, 14(11):2376–2391, 2015.

[56] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A Secure Sharding Protocol For Open Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 17–30, 2016.

[57] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 31–42, New York, NY, USA, 2016. ACM.

[58] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge university press, 2017.

[59] A. Mohaisen and J. Kim. The Sybil Attacks and Defenses: A Survey. *Smart Computing Review*, 3(6):480–489, 2013.

[60] H. G. Molina, F. Pittelli, and S. Davidson. Applications of Byzantine Agreement in Database Systems. *ACM Transactions on Database Systems (TODS)*, 11:27–47, 1986.

[61] D. Mónica, J. ao Leitão, L. Rodrigues, and C. Ribeiro. On the Use of Radio Resource Tests in Wireless Ad-Hoc Networks. In *Proceedings of the 3rd Workshop on Recent Advances on Intrusion-Tolerant Systems*, pages F21–F26, 2009.

[62] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. http://bitcoin.org/bitcoin.pdf.

[63] T. Neudecker, P. Andelfinger, and H. Hartenstein. A Simulation Model for Analysis of Attacks on the Bitcoin Peer-to-Peer Network. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1327–1332, 2015.

[64] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil Attack in Sensor Networks: Analysis & Defenses. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 259–268, 2004.

[65] Oceanstore. The Oceanstore Project. http://oceanstore.cs.berkeley.edu.

[66] N. I. of Standards and Technology. Secure Hash Standard (SHS). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf, 2015.

[67] R. Pass and E. Shi. Hybrid consensus: Efficient consensus in the permissionless model. *IACR Cryptology ePrint Archive*, 2016:917, 2016.

[68] R. Pass and E. Shi. Thunderella: Blockchains with optimistic instant confirmation. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 3–33, Cham, 2018. Springer International Publishing.

[69] N. Preguiça, R. Rodrigues, C. Honorato, and J. Lourenço. Byzantium: Byzantine-Fault-Tolerant Database Replication Providing Snapshot Isolation. In *Proceedings of the Fourth Conference on Hot Topics in System Dependability*, page 9. USENIX Association, 2008.

[70] J. Rangasamy, D. Stebila, L. Kuppusamy, C. Boyd, and J. Gonzalez Nieto. *Efficient Modular Exponentiation-Based Puzzles for Denial-of-Service Protection*, pages 319–331. Springer Berlin Heidelberg, 2012.

[71] C. Scheideler and S. Schmid. A distributed and oblivious heap. In *Proceedings of the 36th Internatilonal Collogquium on Automata, Languages and Programming: Part II*, ICALP '09, pages 571–582, 2009.

[72] M. Sherr, M. Blaze, and B. T. Loo. Veracity: Practical Secure Network Coordinates via Vote-based Agreements. In *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, USENIX'09, pages 13–13, 2009.

[73] SINTRA. Distributed Trust on the Internet. www.zurich.ibm.com /security /dti/.

[74] K. Srinathan and C. P. Rangan. Efficient asynchronous secure multiparty distributed computation. In *INDOCRYPT 2000, Lecture Notes in Computer Science*, volume 1977, pages 117–129. Springer-Verlag, 2000.

[75] G. Urdaneta, G. Pierre, and M. van Steen. A Survey of DHT Security Techniques. *ACM Computing Surveys*, 43(2):1–53, 2011.

[76] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. DDoS Defense by Offense. *ACM Transactions on Computer Systems (TOCS)*, 28(1):3, 2010.

[77] H. Wang, Y. Zhu, and Y. Hu. An Efficient and Secure Peer-to-Peer Overlay Network. In *Proceedings of the IEEE Conference on Local Computer Networks*, pages 764–771, 2005.

[78] L. Wang and J. Kangasharju. Measuring Large-Scale Distributed Systems: Case of BitTorrent Mainline DHT. In *IEEE 13th International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10, 2013.

[79] H. Yu. Sybil Defenses via Social Networks: A Tutorial and Survey. *SIGACT News*, 42(3):80–101, Oct. 2011.

[80] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. V. Renesse. REM: Resource-efficient mining for blockchains. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1427–1444, Vancouver, BC, 2017. USENIX Association.

[81] W. Zhao. A Byzantine Fault Tolerant Distributed Commit Protocol. In *Proceedings of the* 3^{rd} *IEEE International Symposium on Dependable, Autonomic and Secure Computing*, pages 37–46, 2007.