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*** [21], 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), whereby IDs must periodically solve computational puzzles, in order to limit 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 equal 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 following sense: *the good IDs spend at a rate that is asymptotically less than the attacker*.

In particular, we present an algorithm, ***Geometric Mean COMputation (*GMCom*)***, that 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 for a large family of algorithms.

Why might an asymmetric result be useful? PoW requires an energy expenditure, and this ultimately translates into a monetary cost. This is true whether an adversary uses its own machines, for which energy costs are well-documented (see [22], [60]), or offloads the effort by renting a botnet (see [24]). Therefore, as our title implies, an asymmetric defense may serve as a convincing deterrent against Sybil attacks by inflicting a *higher* cost on the adversary than on the good participants in the system.

This work is supported by the National Science Foundation grants CNS-1318880, CCF-1320994, CCF 1613772, and CNS 1816076.

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 cannot be stolen or pre-computed. This is a common assumption in PoW systems [6], [42], [49]. For completeness, we now describe the standard way in which this assumption is achieved.

All IDs have access to a hash function, h, about which we make the *random oracle assumption* [10], [38]. 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 [53], with inputs and outputs of sufficiently large bit lengths.

Solving a puzzle requires that an ID find an input x such that $h(x)$ is less than some threshold. The input found is the ***puzzle solution***. Decreasing this threshold value will increase the difficulty, since one must compute the hash function on more inputs to find an output that is sufficiently small.

We assume that each good ID can perform μ hash-function evaluations per round for $\mu > 0$, where a ***round*** is the amount of time it takes to solve our easiest computational puzzle plus the time to communicate the solution to the rest of the network (see DIFFUSE described below). 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 [12], [13], [32].

Our technical puzzle generation process follows that of [6]. For any integer $\rho \geq 1$, we define a ***ρ-hard puzzle*** to consist of finding $C \log \mu$ solutions using a threshold of

$\rho(1-\delta)\mu/(C\log\mu)$, 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 (r.v.) giving the expected number of hash evaluations needed to compute $C\log\mu$ solutions. Note that X is a negative binomial r.v. and so, the following concentration bound holds (see, for example, Lemma 2.2 in [8]) for every $0 < \epsilon \leq 1$:

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

Given the above, one can show that every good ID will solve a ρ-hard puzzle with at most $\rho\mu$ hash function evaluations, and that the adversary must compute at least $(1-2\delta)\rho\mu$ hash evaluations to solve every ρ-hard puzzle. This follows from a union bound over $O(n_0^\gamma)$ join and departure events by IDs, where γ is any fixed positive constant, and for C being a sufficiently large constant depending on δ, μ 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-hard puzzle requires computational cost ρ to solve**.

Finally, we must be able to address an adversary who attempts (1) to falsely claim puzzle solutions computed and transmitted by good IDs, and (2) to pre-compute solutions to puzzles. These details are deferred until Section II-B when we describe our algorithm.

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 $\alpha > 0$ is 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 [6], [26], [49], [62].[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 [11], [25], [26]. Our adversary can read messages diffused by good IDs before sending its own, and can also send messages directly to any ID. All IDs are assumed to be synchronized, and time is discretized into rounds as defined above.

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***. Informally, an epoch corresponds to an amount of time over which a significant fraction of the system membership (good and bad IDs) has changed. 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.

Let $\boldsymbol{\ell_i}$ denote the duration of epoch i. Let $\boldsymbol{J_i}$ be the join rate of good IDs in epoch i; that is, the number of good IDs that join in epoch i divided by ℓ_i.We make the following assumptions about the good IDs which are used to prove the asymmetric property of GMCOM in Sections III-B and III-C:

- **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.** For $i \geq 1$, $J_{i-1}/2 \leq J_i \leq 2J_{i-1}$.
- **A3.** For any $i \geq 1$, $\ell_i \leq 2|S_{i-1}|/J_i$, where $\boldsymbol{S_{i-1}}$ is the set of all IDs in the system at the beginning of epoch i.
- **A4.** 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.

We assume that the probability that a departing good ID is in the committee equals the fraction of total good IDs that are in the committee. Additionally, we assume that in a single round, an $O(1/\log n_0)$-fraction of good IDs depart. Finally, the minimum number of good IDs in the system at any point is assumed to be at least some value $\boldsymbol{n_0}$. These assumptions are used in Section III-A to prove the invariants defined below.

Invariant Goals. We seek to maintain the following two invariants.

Population Invariant: The fraction of bad IDs in the system is always bounded away from $1/2$.

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 [41] (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

In our algorithm, an ***iteration*** consists of an epoch, plus a purge test and the selection of a new committee, both of which occur after the epoch; we describe the purge test and committee selection later in Section II-B.

Fix a subset of iterations $\mathcal{I}$, and let L be the total length of time of those iterations. The ***adversarial spending rate***, $\boldsymbol{T_\mathcal{I}}$, is the cost to the adversary for solving puzzles whose solutions are used in any iteration of $\mathcal{I}$ divided by L. The ***good ID join rate***, $\boldsymbol{J_\mathcal{I}}$, is the number of good IDs that join over the iterations in $\mathcal{I}$ divided by L. Finally, the ***algorithmic spending rate*** is the total cost to the good IDs for solving puzzles whose solutions are used in any iteration of $\mathcal{I}$ divided by L.

We now state our result for our algorithm GMCOM.

[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.

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) For any subset of iterations $\mathcal{I}$ not containing iteration 1, the algorithmic spending rate is $O(\sqrt{T_{\mathcal{I}} J_{\mathcal{I}}} + J_{\mathcal{I}})$.

Our lower bound is as follows. We define a ***purge-based*** algorithm to be any algorithm where (1) IDs pay a 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 iteration. The algorithmic spending rate is $\Omega(\sqrt{T J} + J)$, where J is the good ID join rate and T is the algorithmic spending rate, over the iteration.*

C. Model Discussion

We note that A1-A4 admit a highly-dynamic system, since the number of good IDs that join or depart in any single round may be nearly linear in the system size at the beginning of the epoch; additionally, there are no constraints on the behavior of bad IDs. Also, there is nothing special about the constants used in A1-A4; 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 [64], and data collection on the Bitcoin network indicates a network of more than $5,000$ IDs over the past two years [14].

Finally, in the event that 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 [21]. In addition to our recent work [31], there is large body of literature on defenses (see surveys [7], [20], [34], [46], [51]). *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].

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 [27], [28], [48], [51]. However, this approach may fail if the adversary can monitor all of the channels.

Several results leverage social networks to yield Sybil resistance (see the survey [65]). 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 [9], [19], [58], [63], 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 [18], [57], 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, puzzles have the useful property that verifying a solution is 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 [62], require verification that a sufficient number of packets are 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 [2]. 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 [26], 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 [39], [54], [55] and scalable blockchains [29], [33], [44], [66]. When the number of bad participants is assumed to be a bounded minority, several adversarial fault-tolerant systems exist (see the survey [61]).

Byzantine Consensus. The Byzantine consensus problem [41] is described as follows. Each good ID has an initial input and the goal is for (1) all good IDs to decide on the same input; and (2) this input to equal the input 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 [15], [23], [26], [30]; trustworthy computing [16], [17], [40], [59]; P2P systems [52]; and databases [47], [56]. Establishing Byzantine consensus via a committee is a common approach (see [26], [37], [43]).

II. OUR ALGORITHM

In this section, we describe our algorithm, GMCOM, whose pseudocode is provided in Figure 1.

A. Preliminaries

GENID. To initialize our system, we make use of an algorithm created by Andrychowicz and Dziembowski [6], 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. GENID also selects a committee of logarithmic size that has a majority of good IDs. Finally, GENID 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. In such a committee

GMCOM

Key Variables

i: iteration number
$\mathcal{S}_i$: set of IDs at end of iteration i
$\mathcal{S}_{\text{cur}}$: 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 iteration 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 iteration.
   (b) $\mathcal{S}_i \leftarrow$ set of IDs returning difficulty 1 puzzle solutions within 1 round.
   (c) The committee selects a new committee of size $\Theta(\log n_0)$ from $\mathcal{S}_i$ and sends out this information via DIFFUSE.
   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.

the algorithm in [36] can perform Byzantine consensus in $O(1)$ expected and $O(\log n_0)$ worst case rounds. The committee uses Byzantine consensus to maintain values for key variables, decide when an epoch ends, generate and issue random seeds for puzzles, and create a new committee.

B. Overview of GMCOM

We describe our algorithm while referring to our pseudocode in Figure 1. We then elaborate on specific aspects of our algorithm in Sections II-C, II-E, and II-D.

The execution of GMCOM is broken into ***iterations***, each of which corresponds to running Steps 1 and 2 of the for-loop in the algorithm.

Step 1 of an iteration lasts for an epoch as defined in Section I. In particular, for any epoch $i \geq 1$, denote the set of IDs (good and bad) at the start of epoch i (equivalently, the end of iteration $i-1$) by $\boldsymbol{\mathcal{S}_{i-1}}$, and denote the current set of IDs (good or bad) by $\boldsymbol{\mathcal{S}}_{\text{cur}}$. Step 1 (epoch i) ends when $|\mathcal{S}_{\text{cur}} \otimes \mathcal{S}_{i-1}| \geq |\mathcal{S}_{i-1}|/3$.

In Step 1, each joining ID must solve a puzzle of difficulty $\max\left\{\lceil J_{\text{cur}}/\tilde{J}_{i-1}\rceil, 1\right\}$, where $\boldsymbol{J}_{\text{cur}}$ is the current join rate defined as the current number of join events in epoch i divided by current time elapsed in epoch i, and $\boldsymbol{\tilde{J}_{i-1}}$ is the estimate of the join rate for good IDs in the previous epoch (note $\tilde{J}_0 \leftarrow \infty$). We provide intuition for this cost in Section II-C.

The committee tracks ID joins and departures in order to calculate the symmetric difference. Tracking of departures can be done by having each ID issue a "heartbeat" message to the committee periodically to let them know the ID is still present in the system.

Step 2 starts when $|\mathcal{S}_{\text{cur}} \otimes \mathcal{S}_{i-1}| \geq |\mathcal{S}_{i-1}|/3$. A ***purge test*** is then initiated by the committee as follows. All IDs are immediately issued a 1-hard puzzle via DIFFUSE (Step 2(a)). Included in this communication from the committee is a random string r which must be incorporated into the puzzle solution. This randomness prevents a pre-computation attack by the adversary; see Section II-E for more details.

In Step 2(b), each ID must respond with a valid solution within 1 round, or else be removed from the system; the committee removes unresponsive or late IDs from its whitelist, which is maintained via Byzantine consensus amongst the committee members.

The current committee then selects $\Theta(\log n_0)$ IDs uniformly at random from $\mathcal{S}_i$ (Step 2(c)). This selection is done via a committee election protocol, for example [35], [36]. The committee uses DIFFUSE to inform $\mathcal{S}_i$ that the selected IDs are the new committee for iteration $i+1$. All messages from committee members are verified via public key digital signatures.

In Step 2(d) the current committee calculates $\tilde{J}_i = \left(|\mathcal{S}_{i-1} \otimes \mathcal{S}_i| - \alpha(|\mathcal{S}_{i-1}| + |\mathcal{S}_i|)\right)/\ell_i$, and gives this to the new committee. Intuitively, this is an estimate of the good ID join rate during epoch i; more discussion is given in Section II-D.

Finally, in Step 2(e), iteration i ends.

C. Intuition for Why GMCom is Asymmetric

We now give intuition for our entrance-cost function. In the absence of an attack, the entrance cost should be small. In this case, J_{cur} is always 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 A2), and these join events are roughly evenly-distributed over the epoch (Assumption A4). Consequently, each good ID will spend $O(1)$ to join.

In contrast, when there is a large attack, the entrance-cost function imposes 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.

Imagine that the adversary injects IDs at a rate of $(\tilde{J}_{i-1})^k$ for some $k \gg 1$. Then the entrance cost is $(\tilde{J}_{i-1})^{k-1}$. Thus, the spending rate for the adversary is the entrance cost times the adversarial join rate, which is $T = (\tilde{J}_{i-1})^{2k-1}$. Since the good ID join rate in epoch i is just $\Theta(\tilde{J}_{i-1})$ (by Assumption A2), the cost rate for the good IDs is approximately $(\tilde{J}_{i-1})^k$, which equals $\Theta(\sqrt{TJ})$, where $J = \tilde{J}_{i-1}$.

These two extreme cases provide intuition for the asymmetric cost guaranteed by GMCom. Interpolating between these cases, showing that $\tilde{J}_i$ is a good estimate for J_{i-1}, and incorporating multiple epochs and purge costs, is the subject of our analysis in Section III.

D. A Note on Estimating the Join Rate of Good IDs, $\tilde{J}_i$

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

E. How Puzzles Are Used

Although all puzzles are 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 an ***entrance puzzle***.

The solution to the puzzle is $\boldsymbol{K_v}||\boldsymbol{s}||\boldsymbol{\tau}$, where $\boldsymbol{K_v}$ is the public key of v, s is a nonce selected by v in order to solve the puzzle, and $\boldsymbol{\tau}$ is the timestamp of when the puzzle solution was generated. The value of τ in the solution to an entrance puzzle must be within some small margin of the current time which, in practice, would primarily depend on network latency.

Note that, for a bad ID, a 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 our invariants.

The second type of puzzle is a ***purge puzzle***, which limits the fraction of bad IDs in the system, and 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 $\Theta(\log n_0)$ bits is generated by the committee and included as part of the announcement. The string r must be appended to the inputs for all requested solutions in this round; that is, the input is $\boldsymbol{K_v}||\boldsymbol{s}||\boldsymbol{r}$. The string r ensures 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 this random string 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.

Using the public key in the puzzle construction 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? No, since 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.

TABLE I: A summary of our notation.

Used in Model and Algorithm	
Symbol	**Description**
α	Fraction of total computational power that the adversary controls.
n_0	Lower bound on 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.
ℓ_i	Length of epoch i.
J_i	(Number of good IDs joining in epoch i) / ℓ_i.
$\mathcal{S}_i$	Set of all IDs in the system at the end of iteration i. Equivalently, the set at the beginning of epoch $i+1$.
$\tilde{J}_i$	$(\lvert\mathcal{S}_{i-1} \otimes \mathcal{S}_i\rvert - \alpha(\lvert\mathcal{S}_{i-1}\rvert + \lvert\mathcal{S}_i\rvert))/\ell_i$.
$\mathcal{S}_{\text{cur}}$	Set of all IDs in the system at current time.

Used in Proofs	
Symbol	**Description**
G_i	Number of good IDs in the system at the end of iteration i.
B_i	Number of bad IDs in the system at the end of iteration i.
N_i	$G_i + B_i$.
g_i^a	Number of good IDs that arrive in iteration i.
b_i^a	Number of bad IDs that arrive in iteration i.
n_i^a	$g_i^a + b_i^a$.
g_i^d	Number of good IDs that depart in iteration i.
b_i^d	Number of bad IDs that depart in iteration i.
n_i^d	$g_i^d + b_i^d$.
$\mathcal{T}_i$	Total computational cost to the adversary in iteration i.

III. Upper Bounds

In this section, we begin by showing the correctness of GMCom, followed by proving 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. Maintaining the Population and Committee Invariants

We first prove that the population invariant always holds. For any iteration i, we let $\boldsymbol{B_i}$ and $\boldsymbol{G_i}$ respectively denote the number of bad and good IDs in the system at the end of epoch i, and let $\boldsymbol{N_i = B_i + G_i}$.

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

Proof. For $i = 0$, by the use of GENID for initializing the system (recall Section II-A), $B_0 < (3/10)N_0$. For $i > 0$, since $\alpha \leq 1/14$, the number of 1-hard puzzles the adversary can solve during the purge at the end of iteration i is less than $G_i/2$. Thus, for all $i > 0$, 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 iteration i. Similarly, let $\boldsymbol{n_i^d, g_i^d, b_i^d}$ denote the total, good, and bad IDs that depart over iteration i. We can now prove the population invariant.

Lemma 4. *The fraction of bad IDs is always less than $1/2$.*

Proof. Fix some iteration $i > 0$. By Step 2 of our algorithm, we always have that $|\mathcal{S}_{i-1} \otimes \mathcal{S}_{\text{cur}}| \leq |\mathcal{S}_{i-1}|/3$ 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 iteration. Thus, we pessimistically assume all additions of bad IDs and removals of good IDs come first. 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, $B_{i-1} < N_{i-1}/3$. Thus, 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 $b_i^a + g_i^d < N_{i-1}/3$. 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

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

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 $b_i^a + g_i^d < N_{i-1}/3$, and this does not depend on b_i^d. □

Next, we prove that GMCOM preserves the committee invariant for a number of iterations that is polynomial in n_0, say n_0^γ for any fixed positive constant $\boldsymbol{\gamma}$.

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.

Lemma 5. *Over a polynomial number of join and departure events, there is always an honest majority in the committee with probability $1 - O(1/n_0)$.*

Proof. For iteration $i = 0$, the committee invariant holds by the use of GENID to initialize the system (recall Section II-A; for details, see Lemma 6 of [6]).

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

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

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 Bounds [45]:

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

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

What about the number of good IDs in the committee over the iteration? 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 less than $|S_{i-1}|/3$. Since the probability that a departing good ID is in the committee equals the fraction of total good IDs that are in the committee (recall 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 upper bound the number of departures of good IDs from the committee using Chernoff Bounds [45]:

$$\begin{aligned} Pr\left(Y_g > (1+\delta') \frac{c \log |S_i|}{3}\right) \quad & \leq \quad \exp\left\{-\frac{\delta'^2 c}{9} \log |S_i|\right\} \\ & = \quad O\left(n_0^{-(\gamma+1)}\right) \end{aligned}$$

where the first step holds for any constant $0 < \delta' < 1$ and the last step holds for all $c \geq \frac{9(\gamma+1)}{\delta'^2}$.

Letting $\delta' = 1/5$, the following result holds: with probability $1 - O(n_0^{-(\gamma+1)})$, the minimum number of good IDs in the committee is greater than $(9/10)c\log|S_i| - (4/10)c\log|S_i|$ at any point during the epoch in iteration $i > 0$.

The formation of a new committee and generation of a random string can be performed w.h.p. by the existing committee using the Byzantine consensus algorithm in [36]. Given that no more than $\epsilon'/\log n_0$ good IDs depart per round, for a sufficiently small constant $\epsilon' > 0$, and that the probability that a departing good ID is in the committee equals the fraction of total good IDs that are in the committee (recall Section I-A), then w.h.p. the committee maintains a majority of good IDs for any iteration $i > 0$.

Finally, by a union bound over n_0^γ iterations, the committee invariant is maintained over n_0^γ iterations with probability $1 - O(1/n_0)$, which implies the claim. □

B. Bounds on Estimation of Good ID Join Rate

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

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

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

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 iteration 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 A3:

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

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

$$\tilde{J}_i \;\geq\; (1/3 - (7/3)\alpha)(J_i/2) \;\geq\; J_i/12$$

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

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

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

$$\begin{aligned}\tilde{J}_{i-1} &\leq 3J_{i-1} && \text{by Lemma 6} \\ &\leq 6J_i && \text{by Assumption A2 in Section I-A.}\end{aligned}$$

Similarly, we can obtain the lower bound:

$$\begin{aligned}\tilde{J}_{i-1} &\geq \frac{J_{i-1}}{12} && \text{by Lemma 6} \\ &\geq \frac{J_i}{24} && \text{by Assumption A2 in Section I-A}\end{aligned}$$

which completes the proof. □

C. Cost Analysis

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

We divide epoch i of iteration i into ***sub-epochs***. Sub-epoch $j \geq 1$ begins when $(j-1)/J_i$ time has elapsed in epoch i and ends when j/J_i time has elapsed. $\boldsymbol{\mathcal{T}_i^j}$ 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 $\boldsymbol{b_i^j}$ 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 $\boldsymbol{b_i}$ 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:

$$\max\left\{\frac{k/t_j}{\tilde{J}_{i-1}}, 1\right\} \geq \frac{k}{\tilde{J}_{i-1}\,t_j} \geq \frac{k}{6\,J_i\,t_j} \geq \frac{k}{6\,j}$$

where the second inequality follows by Lemma 7. Thus:

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

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

We use the following fact in the proofs of Lemma 10 and 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 iteration $i \geq 2$, the total entrance cost paid by the good IDs is $O\left(\sqrt{\mathcal{T}_i J_i \ell_i} + J_i \ell_i\right)$.*

Proof. Fix an iteration $i \geq 2$ and let $j \geq 1$. Let the j^{th} sub-epoch end at time $t_j = j/J_i$. By Assumption A4, 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 A4, the total entrance cost paid by the good IDs is at most:

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

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

Proof. Recall that the number of bad IDs that remain in the system at the end of iteration $i-1$ is at most $\alpha|\mathcal{S}_{i-1}|$ and that the number of bad IDs that enter in iteration i is b_i. Therefore, the number of join and leave events in iteration 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 A1, so the number of good IDs that join or depart in iteration i is at most $(5/2)J_i\ell_i$. Thus, the total join and leave events in iteration i is at most $2b_i + \alpha|\mathcal{S}_{i-1}| + (5/2)J_i\,\ell_i$.

By Step 2 of GMCom, $|\mathcal{S}_{i-1} \otimes \mathcal{S}_i| = |\mathcal{S}_{i-1}|/3$. Thus:

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

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 $|\mathcal{S}_{i-1}|$:

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

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

Lemma 12. *For any iteration $i \geq 2$, 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 iteration i is at most:

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

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. **Population and Committee Invariants.** Follows from Lemma 4 and Lemma 5.

Spending Rate. Let $\mathcal{I}$ be any subset of all the iterations during the lifetime of the system. By Lemmas 10 and 12, the total computational cost paid by good IDs in all iterations in $\mathcal{I}$ is:

$$\sum_{i\in\mathcal{I}} O\left(\sqrt{\mathcal{T}_i J_i\, \ell_i} + J_i\, \ell_i\right) = O\left(\sqrt{\sum_{i\in\mathcal{I}} \mathcal{T}_i \sum_{i\in\mathcal{I}} J_i \ell_i} + \sum_{i\in\mathcal{I}} J_i \ell_i\right)$$

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

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

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

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 iterations, but we consider more general iterations 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 iteration, 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 iteration varies only with the join rate during that iteration. GMCom's cost function has this property, since $\tilde{J}_{i-1}$ is fixed throughout iteration 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 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 $G = \Omega(\sqrt{T\,J} + J)$, where J is the good ID join rate, and T is the algorithmic spending rate, both taken over the iteration.*

Proof. Fix an iteration i. Let n be the number of IDs in the system at the start of iteration i. Let ρ be any adversarial join rate during this iteration. We show that the adversary can always force the good nodes' spending rate to be $\Omega(\sqrt{T\,J})$, where T is the spending rate of the adversary in iteration i, and J is the join rate of good IDs in iteration i. The adversarial strategy is for bad IDs to join uniformly at a rate of ρ during the iteration, and then depart right before the purge.

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

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

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

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

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 iteration. The adversarial join rate is ρ , so we have:

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

Rearranging we get:

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

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

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

On solving the above for ρ, we get:

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

By definition, the good ID join rate is 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:

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

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

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

Rearranging, we get:

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

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

$$\begin{aligned} G &= \Omega(\rho + J) \\ &= \Omega\left(\sqrt{T\,J} + J\right) \end{aligned}$$

where the last line follows from Equation 8. □

V. Experiments

We simulate GMCom to evaluate its performance with respect to the computational cost from solving puzzles. Given this goal, we do not model Byzantine consensus or committee formation. In all of our experiments, we assume a computational cost of k for solving a puzzle of difficulty k.

In Section V-B, we validate the asymptotic behavior of the asymmetric spending rate. In Section V-A, we compare GMCom with two PoW-based Sybil defenses.

Our simulation code [1] is written in MATLAB and was executed on a Mac machine with High Sierra (version 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 $O(\sqrt{T\,J}+J)$, where T is the adversary's computational cost for solving puzzles divided by the duration of an attack, and J is the average join rate of good IDs over the duration of the experiment. 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 $\boldsymbol{G}$.

We assume $\alpha = 1/14$, and T ranges over $[2, 2^{100}]$, where for each 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 first 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 when $G = T$ 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.

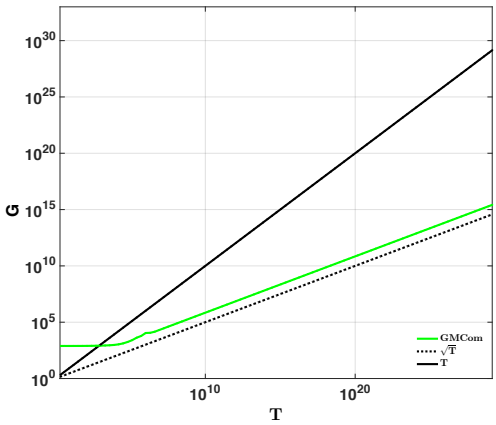

Fig. 2: Average cost for GMCOM vs. average cost to adversary.

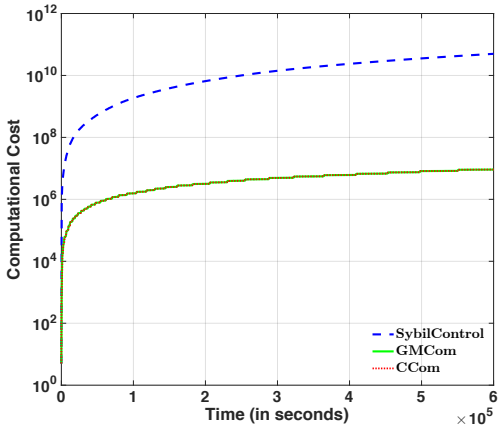

Fig. 3: Cumulative cost for algorithms in Scenario I.

B. Comparison with Existing PoW-Based Sybil Defenses

We implement and evaluate two contemporary PoW-based algorithms, CCOM [31] and SYBILCONTROL [42].

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

1) The Bitcoin Network: We simulate CCOM, SYBILCONTROL, and GMCOM on a real-world dataset for the Bitcoin network [50] consisting of roughly 7 days of join/departure-event timestamps (data obtained by personal correspondence with Till Neudecker [50]). 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. The overall cost of SYBILCONTROL is much higher than that of CCOM and GMCOM before the attack. 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 [50].

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 Monte-Carlo simulations, where each value is averaged over 20 separate simulations.

Figures 4(c) - (f) illustrate the cost to good IDs as the budget of the adversary per second varies. We note that at the largest value of $T = 2^{16}$, a purge occurs once per second in GMCOM, and an ID tests its neighbors every 5 seconds in SYBILCONTROL. Even with this high purge rate, the asymmetric property of GMCOM allows it to outperform SYBILCONTROL (and CCOM); note the logarithmic y-axis.

VI. CONCLUSION AND FUTURE WORK

We have presented an asymmetric Sybil defense using PoW. We have provided empirical evidence that our algorithm

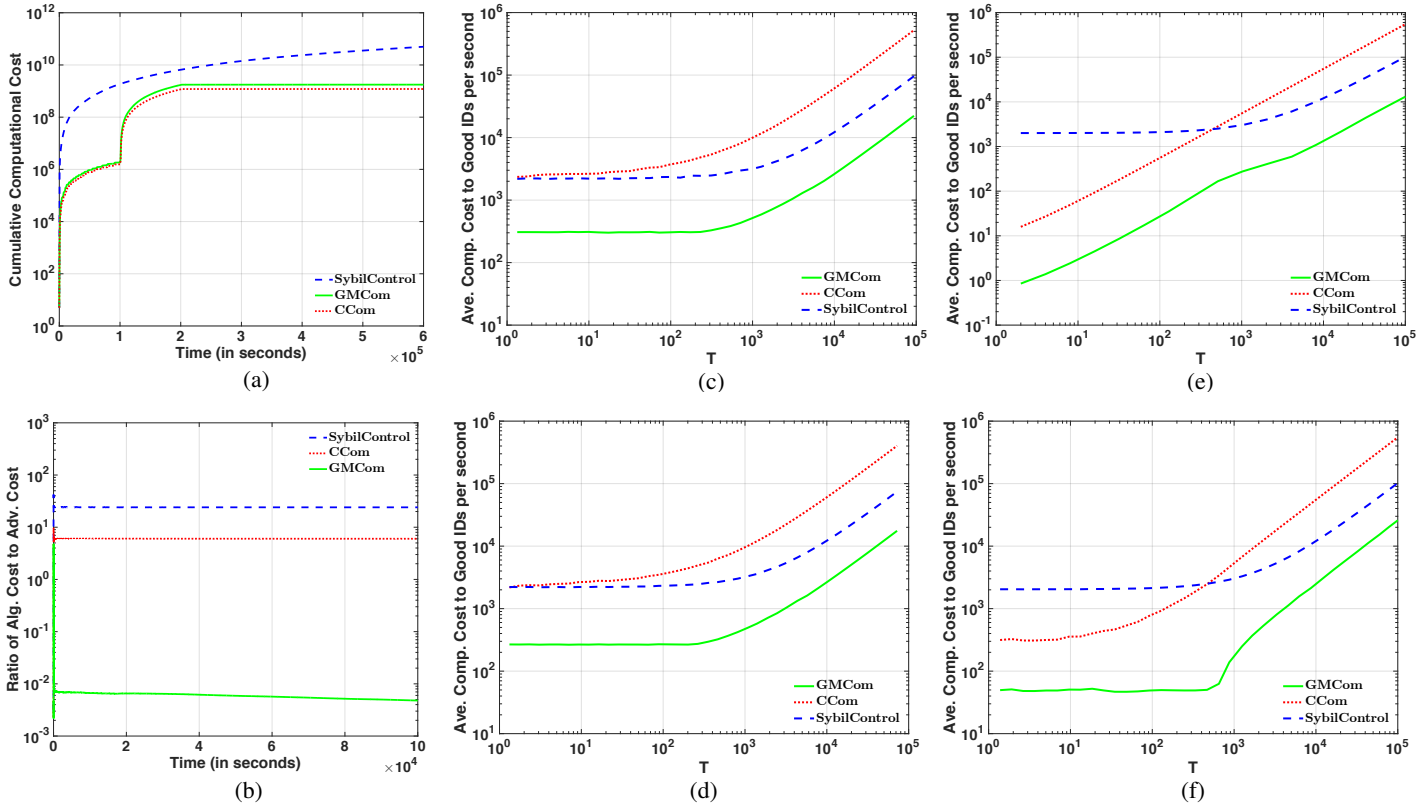

Fig. 4: Bitcoin experiments: (a) cumulative computational cost and (b) ratio of algorithmic cost to adversarial cost for Scenario II. Average computational cost to good IDs per second vs. adversarial computational cost per second for the following peer-to-peer networks: (c) BitTorrent Debian, (d) BitTorrent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

is as efficient as state-of-the-art PoW-based 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] Source code is available at https://dikshagupta.blog/research/.

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

[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. PoW-Based Distributed Cryptography with No Trusted Setup. In *Annual Cryptology Conference*, pages 379–399. Springer, 2015.

[7] 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.

[8] 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.

[9] 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.

[10] 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, 1993.

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

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

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

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

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] T. Economist. The Magic of Mining. 2015. www.economist.com/news/business/21638124-minting-digital-currency-has-become-big-ruthlessly-competitive-business-magic.

[23] 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.

[24] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *Proceedings of the* 14^{th} *ACM Conference on Computer and Communications Security*, pages 375–388, 2007.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

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

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

[34] 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.

[35] B. Kapron, D. Kempe, V. King, J. Saia, and V. Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 1038–1047, 2008.

[36] J. Katz and C.-Y. Koo. On expected constant-round protocols for byzantine agreement. *J. Comput. Syst. Sci.*, 75(2):91–112, Feb. 2009.

[37] 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.

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

[39] 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 *Proceedings of the* 25^{th} *USENIX Security Symposium (USENIX)*, pages 279–296, 2016.

[40] 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.

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

[42] 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.

[43] 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.

[44] 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)*, pages 31–42, 2016.

[45] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017.

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

[47] 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.

[48] D. Mónica, L. Leitao, 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.

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

[50] 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.

[51] 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.

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

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

[54] R. Pass and E. Shi. Hybrid Consensus: Efficient Consensus in the Permissionless Model. *IACR Cryptology ePrint Archive*, page 917, 2016.

[55] 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, 2018.

[56] 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.

[57] 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.

[58] M. Sherr, M. Blaze, and B. T. Loo. Veracity: Practical Secure Network Coordinates via Vote-based Agreements. In *Proceedings of the USENIX Annual Technical Conference*, pages 13 –13, 2009.

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

[60] A. Technica. Mining Bitcoins Takes Power, But is it an Environmental Disaster? 2013. http://arstechnica.com/business/2013/04/mining-bitcoins-takes-power-but-is-it-an-environmental-disaster/.

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

[62] 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.

[63] 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.

[64] 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.

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

[66] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. V. Renesse. REM: Resource-Efficient Mining for Blockchains. In *Proceedings of the* 26^{th} *USENIX Security Symposium*, pages 1427–1444, 2017.