Bankrupting Sybil Despite Churn

Diksha Gupta
School of Computing
National University of Singapore
Singapore, Singapore
dcsdg@nus.edu.sg

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
Mississippi, USA
myoung@cse.msstate.edu

***Abstract*—A Sybil attack occurs when an adversary pretends to be multiple identities (IDs). Limiting the number of Sybil (bad) IDs to a minority is critical to the use of well-established tools for tolerating malicious behavior, such as Byzantine agreement and secure multiparty computation.**

A popular technique for enforcing a Sybil minority is resource burning: verifiable consumption of a network resource, such as computational power, bandwidth, or memory. Unfortunately, typical defenses based on resource burning require non-Sybil (good) IDs to consume at least as many resources as the adversary. Additionally, they have a high cost, even when the system membership is relatively stable.

Here, we present a new Sybil defense, ERGO, that guarantees (1) there is always a minority of Sybil IDs; and (2) when the system is under significant attack, the good IDs consume asymptotically less than the bad. In particular, for churn rate that can vary exponentially, the resource burning rate of ERGO is $O(\sqrt{TJ} + J)$, where T is the resource burning rate of the adversary, and J is the join rate of good IDs.

We empirically evaluate ERGO alongside prior Sybil defenses. Unlike other Sybil defense, ERGO can be combined with machine learning techniques for identifying Sybil IDs, in a way that maintains its theoretical guarantees. Based on our experiments comparing ERGO with two state-of-the-art Sybil defenses, we show that ERGO improves by up to 2 orders of magnitude without machine learning, and up to 3 orders of magnitude using machine learning.

***Keywords*-Sybil attack; resource burning; security;**

I. INTRODUCTION

A ***Sybil attack*** occurs when a single adversary pretends to be multiple identities (***IDs***) [27]. A classic defense is resource burning, whereby IDs are periodically required to consume local resources in a verifiable manner [44]. A well-known example of resource burning is proof-of-work (PoW) [28], but several other methods exist (see Section II).

Unfortunately, a major drawback of resource burning is that resources are always consumed, even when the system is not under attack. This non-stop consumption translates into substantial energy and monetary costs [58], [59].

Prior work shows it is sometimes possible for good IDs to consume fewer total resources than the adversary [42]. Unfortunately, this work fails to hold in settings where the rate at which system membership changes—often referred to as the ***churn*** rate— is high. Many systems vulnerable to Sybil attacks have high churn [29], [79], [84].

Thus, a key question is: *Can we design a Sybil defense where good IDs spend less than the attacker despite churn?*

A. Our Contributions

We demonstrate such a defense, **ERGO**. Informally, our model of churn is as follows (cf. Section II-A). *Epoch* boundaries occur when the membership of good IDs changes by a constant fraction. Churn due to bad IDs is arbitrary, while churn due to good IDs is specified by two *a priori unknown* parameters: $\boldsymbol{\alpha}, \boldsymbol{\beta}$. First, the good join rate between two consecutive epochs differs by at most an α factor. Second, the number of good IDs that join or depart during any duration of ℓ seconds within an epoch differs by at most a β-factor from the expected number over that duration. Thus, α characterizes how the rate at which good IDs join, denoted by $\boldsymbol{\rho}$, changes over epochs; and β characterizes the burstiness of good ID arrivals and departures within an epoch.

ERGO ensures the fraction of bad IDs is always less than $1/6$; this constant is arbitrary and can be decreased by reducing $\boldsymbol{\kappa}$, the fraction of the system resources the adversary is assumed to control. Let the ***good spend rate*** be the total resource burning cost for all good IDs per second. Similarly, let $\boldsymbol{T}$ denote the ***adversary's spend rate*** and let $\boldsymbol{J}$ be the ***join rate of good IDs*** over the system lifetime. All of our theorems hold with probability of error that is $o(1/n_0)$ over a number of ID good and bad ID joins and departures polynomial in n_0, where $\boldsymbol{n_0}$ is a lower bound on the number of good at any time in the system. In the following, $\kappa = 1/18$ for ease of analysis, larger values can be tolerated. Additionally, the fraction of bad IDs can be held smaller than $1/6$ by reducing κ.

Theorem 1. *For $\kappa \leq 1/18$,* ERGO *ensures that the fraction of bad IDs in the system is always less than $1/6$ and has good spend rate $O\left(\alpha^6\beta^4\left(\sqrt{T(J+1)}+J\right)\right)$.*

ERGO makes critical use of a second algorithm, **GOODJEST**, that may be of independent interest. GOODJEST estimates the good join rate assuming the fraction of bad IDs is always less than $1/6$.

Theorem 2. *Assume the fraction of bad IDs is always less than* $1/6$*. Fix any epoch. Let* ρ *be the good join rate during that epoch. Then, if* $\tilde{\mathtt{J}}$ *is the estimate from* GOODJEST *at any time during that epoch:*

$$1/(418\alpha^4\beta^3)\rho \leq \tilde{\mathtt{J}} \leq 267\alpha^4\beta^5\rho.$$

This holds no matter how the adversary injects bad IDs. Based on our experiments on multiple networks, GOODJEST always provides an estimate within a factor of 10 of the true good join rate, and often much closer (cf. Section VII-B).

We validate our theoretical results by comparing ERGO against prior PoW defenses using real-world data from several networks (Section VII-A). We find that ERGO performs up to 2 orders of magnitude better than previous defenses, according to our simulations. Using insights from these first experiments, we engineer and evaluate several heuristics aimed at further improving the performance of ERGO. Our best heuristic performs up to 3 orders of magnitude better than previous algorithms for large-scale attacks.

II. MODEL AND PROBLEM

We now describe a general network model. The system consists of virtual ***identifiers (IDs)***, where each ID is either ***good*** if it obeys protocol, or ***bad*** if it is controlled by the Sybil adversary (or just ***adversary***).

Resource-Burning Challenges. IDs can construct resource-burning challenges of varying hardness, whose solutions cannot be stolen or pre-computed; some examples are discussed in Section III. A ***k-hard challenge*** for any integer $k \geq 1$ imposes a resource cost of k on the challenge solver. Our results are agnostic to the type of challenges employed, either those discussed above or new resource-burning schemes available for future use.

Coordination. To simplify our presentation, we assume that there is a single server running our algorithms. However, in Section VIII, we show how the server can be replaced with a small committee, thus allowing our algorithms to execute in decentralized settings.

A ***round*** is the amount of time it takes to solve a 1-hard challenge plus time for communication between the server and corresponding ID for issuing the challenge and returning a solution. As is common in the literature, we assume that good IDs have clocks that are closely synchronized. Intuitively, if there is message delay or clock drift, then a challenger cannot accurately measure the response time for the ID solving the challenge; see [57] for further discussion. Techniques for synchronizing on the order of milliseconds are known and suffice for our purposes [61].

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 can send messages to any ID at will, and can read the messages sent by good IDs before sending its own. It knows when good IDs join and depart, but it does not know the private bits of any good ID.

The adversary is resource-bounded: in any single round where all IDs are solving challenges, the adversary can solve a κ-fraction of the challenges; this assumption is common [5], [34], [68].

Joins and Departures. Every join and departure event is assumed to occur at a unique point in time. In practice, this means that the events are serialized by the server or committee.

Whenever the adversary decides to cause a good ID departure event, the departing good ID is selected independently and uniformly at random from the set of good IDs in the system. Departing good IDs announce their departure. In practice, each good ID can issue "heartbeat messages" to the server that indicate they are still alive; the absence of a heartbeat message is interpreted as a departure by the corresponding ID. Thus, even departures by bad IDs are detectable.

Every joining ID is treated as a new ID. We ensure every joining ID is given a unique name by concatenating a join-event counter to the name chosen by the ID. As in [30], [39], [47], [78], we assume that every joining ID knows at least one good ID in order to be bootstrapped into the system.

We define $\boldsymbol{n_0}$ ($n_0 \geq 4$) to be the minimum number of good IDs in the system at any time. We define the ***system lifetime*** to be the duration over which n_0^γ joins and departures occur, for any fixed constant $\boldsymbol{\gamma} > 0$.

A. Epochs, Smoothness, and Churn

Our model of *good* churn is quite general (cf. [43]); we make no assumptions about the bad churn. Time is divided into ***epochs*** whose boundaries occur when the symmetric difference between the sets of good IDs at the start and the end of the epoch exceeds $3/4$ times the number of good IDs at the start. Epochs are important to our model and analysis; however, our approach does *not* assume knowledge of when epochs begin or end.

Good churn is specified by two *a priori unknown* parameters: $\boldsymbol{\alpha}, \boldsymbol{\beta}$. First, the good join rate between two consecutive epochs differs by at most an α factor. Second, the number of good IDs that join or depart during ℓ consecutive seconds within an epoch differs by at most a β factor from ℓ times the good join rate of the epoch. Thus, α characterizes how the good join rate changes over epochs; and β characterizes the burstiness of good ID arrivals and departures within an epoch.

For any time x, let $\boldsymbol{G(x)}$ be the set of good IDs, and $\boldsymbol{S(x)}$ be the set of all IDs at time x. Also, define $A \triangle B$ to denote the symmetric difference between any two sets A and B, i.e. $A \triangle B = (A - B) \cup (B - A)$.

Definition 3. *For all $i \geq 1$, epoch i begins at time $t = 0$ if $i = 1$, or at time t when epoch $i-1$ ends otherwise. Epoch i ends at the smallest $t' > t$ such that $|G(t') \triangle G(t)| \geq (3/4)|G(t)|$.*

Let $\boldsymbol{\rho_i}$ be the join rate of good IDs (i.e., ***good join rate***) in epoch i; that is, the number of good IDs that join in epoch i divided by the number of seconds in epoch i. The bound on β below is necessary to obtain a lower bound on the number of good IDs joining in an interval (See Lemma 8).

Definition 4. *For any $\alpha \geq 1$ and $1 \leq \beta \leq \sqrt{\frac{5}{80} n_0 - 1}$, and any epoch $i > 1$, we define the following.*

leftmargin=10pt

- **α-smoothness***: $(1/\alpha)\rho_{i-1} \leq \rho_i \leq \alpha\, \rho_{i-1}$.*
- **β-smoothness***: For any duration of ℓ seconds in the epoch, the number of good IDs that join is at least $\lfloor \ell \rho_i / \beta \rfloor$ and at most $\lceil \beta \ell \rho_i \rceil$. Also, the number of good IDs that depart during this duration is at most $\lceil \beta \ell \rho_i \rceil$.*

Varying Churn Rate. The parameter, α captures *any* possible change in the good join rate between consecutive epochs, since there always exists an α that satisfies the definition. Thus, the good join rate may change rapidly. For example, suppose that say, $\alpha = 2$. In this case, the good join rate may increase *exponentially* from epoch to epoch. Similarly, the good join rate may decrease exponentially when $\alpha = 2$. The parameter β ensures there can be possibly large deviations within an epoch from the average good join rate over the entire epoch.

Our problem: DEFID In the well-studied GENID problem [3], [5], [6], [46], [49], there is some initial set of good and bad IDs in a permissionless system. All good IDs must decide on a set of S such that: all good IDs are in S; and at most a $O(\kappa)$-fraction of the IDs in S are bad.

The **DEFID *(DEFend ID)*** problem generalizes GENID to handle churn. Specifically, bad IDs join and depart in a worst-case manner, and good IDs join in a α,β-smooth manner for unknown α and β. Our goal is to ensure that, at any time t, all IDs know a set S_t and that (1) all good IDs are in S_t; and (2) at most a $O(\kappa)$-fraction of IDs in S_t are bad.

DEFID presents novel challenges. The fraction of bad IDs increases whenever a bad ID joins or a good ID departs. Since bad and good IDs cannot be differentiated *a priori*, the desired bound on the fraction of bad IDs may be violated via churn. Naively executing a solution to GENID after every join and departure event would prevent this, but is expensive.

III. RELATED WORK

Churn. A common assumption in related work is that the number of good IDs is *fixed* at a sufficiently large value or can vary by at most a constant factor [12]–[16], [30], [74], [75], [87]. In this setting, several results by Augustine et al. [8]–[12] address robust distributed computation, but with the added challenge that the system membership can change rapidly. Guerraoui et al. [39] address a challenging setting where the system size can vary polynomially as a function of some initial quantity of good IDs. We address the same challenge here. However, we differ from these past works in that our algorithmic resource costs are tuned to both the amount of actual churn and the amount spent by an attacker.

Sybil Attacks. There is significant prior work on Sybil attacks [27]. For example, see surveys [48], [64],and additional work documenting real-world Sybil attacks [69], [83], [86].

Several results leverage social networks for Sybil defense [63], [85], [88]. However, social-network information may not be available in many settings. Another approach is to use network measurements to verify the uniqueness of IDs [33], [54], [76], but these techniques rely on accurate measurements of latency, signal strength, or round-trip times, and this may not always be possible. Containment strategies are explored in overlays [24], [75], but these results do not ensure a bound on the fraction of bad IDs.

Resource Burning. Many resource burning schemes for Sybil defense exist. *Computational puzzles* consume CPU cycles [5], [53], [68]. *Proof of Space-Time*, requires allocation of storage capacity [67]. *Proof of useful-work* consumes CPU cycles to solve challenges applicable to real-world scientific or engineering problems [18], [77].

A *completely automated public Turing test to tell computers and humans apart (CAPTCHA)* is a resource-burning tool where the resource is human effort [66], [81]. CAPTCHAs of tunable hardness have been proposed [17], as have CAPTCHAs that channel human effort into practical problems such as deciphering scanned words or detecting spam [82].

In a wireless network with multiple communication channels, Sybil attacks can be mitigated via *radio-resource testing* if the adversary cannot listen to all channels simultaneously [37], [38], [65]; the resource here is listening capacity.

Finally, we note that Proof of Stake [4], [34], [50] is *not* a resource burning technique. It requires that the "stake" of each ID to be a globally known quantity and thus is likely to remain relevant primarily for cryptocurrencies. Moreover, even in that domain, it is controversial [23].

Guaranteed Spend Rate. In [40] and [42], Gupta et al. proposed two algorithms CCOM and GMCOM that ensure that the fraction of bad IDs is always small, with respective good spend rates of $O(T + J)$ and $O(J + \sqrt{T(J+1)})$. Unfortunately, the second result only holds in the case where (1) churn is sufficiently small; and (2) there is a fixed constant amount of time that separates all join events by good IDs (i.e., non-bursty arrivals). ERGO does not require these assumptions. Finally, outside of the Sybil attack,

ENTIRE BY RATE OF GOOD (ERGO)

$S(0) \leftarrow$ set of IDs that returned a valid solution to 1-hard RB-challenge.

$\tilde{\mathtt{J}}$ is maintained by running GOODJEST in parallel to the following code. $\tau \leftarrow$ time at system initialization; τ' is the current time.

For each iteration, do: leftmargin=14pt

1. Each joining ID is assigned a RB-challenge of hardness 1 plus the number of IDs that have joined in the last $1/\tilde{\mathtt{J}}$ seconds of the current iteration.
2. When number of joining and departing IDs in this iteration exceeds $|S(\tau)|/11$, perform a purge:leftmargin=15pt
   (a) Issue all IDs a 1-hard RB-challenge.
   (b) $\mathcal{S}(\tau) \leftarrow$ IDs solving this RB-challenge in 1 round.
   (c) $\tau \leftarrow \tau'$

Figure 1: Pseudocode for ERGO.

several prior works address network security challenges with results that are parameterized by the adversary's cost [2], [19], [20], [25], [26], [35], [36], [52], [89].

IV. ERGO

Figure 1 gives pseudocode for ERGO. The server initializes system membership with all IDs that solve a 1-hard RB-challenge. Then, execution occurs over disjoint periods of time called ***iterations***. Each iteration consists of Steps 1 and 2.

In Step 1, each joining ID must solve an RB-challenge of hardness 1 plus the number of IDs that joined within the last $1/\tilde{\mathtt{J}}$ seconds, where $\tilde{\mathtt{J}}$ is the current good join rate estimate obtained from GOODJEST. We call the hardness of this RB-challenge the *entrance cost*; intuition for its value is in Section IV-A.

Step 1 lasts until the number of IDs that join and depart in the iteration is at least $1/11$ times the number of IDs at the start of the iteration. In Step 2, the server performs a ***purge*** by resetting system membership to all IDs that solve a 1-hard RB-challenge within 1 round. Note that the server maintains a membership list of all IDs in the system, which is a common technique used in prior works addressing security in permissionless systems.

A. Entrance Cost Intuition

To gain intuition, fix an iteration, assume that $\beta = \Theta(1)$, and let $\mathtt{J}$ be the good join rate during the iteration. Then, in the absence of attack, all entrance costs are $O(1)$ since $O(1)$ good IDs join on average during $1/\tilde{\mathtt{J}} \approx 1/\mathtt{J}$ seconds. If there is a large attack, the adversary pays more than ERGO. For example, if the ratio of bad ID joins to good ID joins is x, then the number of bad IDs joining in any $1/\tilde{\mathtt{J}}$ seconds is about x, so the adversary pays entrance costs of $\Theta(x^2)$

GOOD JOIN ESTIMATE (GOODJEST)

In the following, t' is the current time and $S(x)$ is the set of IDs in the system at time x.

$t \leftarrow$ time at system initialization.
$\tilde{\mathtt{J}} \leftarrow |S(t)|$ divided by time required for initialization.
Repeat forever: whenever $|S(t') \triangle S(t)| \geq \frac{5}{8}|S(t')|$, do leftmargin=15pt

1. $\tilde{\mathtt{J}} \leftarrow |S(t')|/(t' - t)$.
2. $t \leftarrow t'$.

Figure 2: Pseudocode for GOODJEST.

during a time when the good IDs pay entrance costs of $O(x)$, which is square root of what the adversary pays.

More specifically, during a large attack, the adversary's spend rate is $T = \Theta(\xi \mathtt{J}_\mathtt{a})$, where ξ is the average entrance cost, and $\mathtt{J}_\mathtt{a}$ is the join rate for all IDs. Then, the good spend rate due to entrance costs is $\Theta(\xi \mathtt{J})$, and the good spend rate due to purges is $\Theta(\mathtt{J}_\mathtt{a})$. When $\xi = \mathtt{J}_\mathtt{a}/\mathtt{J}$, these two costs are balanced, and the good spend rate due to the entrance costs and purge costs is within a constant factor of:

$$\xi \mathtt{J} + \mathtt{J}_\mathtt{a} \leq 2\mathtt{J}_\mathtt{a} = 2\sqrt{(\mathtt{J}_\mathtt{a})^2} = 2\sqrt{\mathtt{J}_\mathtt{a} \xi \mathtt{J}} = 2\sqrt{\mathtt{J}T}$$

where the first step holds by our setting of ξ, the third step since $\mathtt{J}_\mathtt{a} = \xi \mathtt{J}$, and the final step since $T = \xi \mathtt{J}_\mathtt{a}$.

As a side note, our entrance cost approximates the ratio of the total join rate over the good join rate, which motivates the name ENTIRE BY RATE OF GOOD; ERGO is also the Greek word for work.

Technical Difficulties. While the above gives intuition for our analysis, challenges remain. First, how do we get an estimate of the good join rate when we don't know anything about which IDs are good or bad, when epochs begin or end, or the values of α and β? Solving this problem is a key technical difficulty, addressed by our algorithm GOODJEST, which we describe and analyze in Section VI-A.

Second, how can the above intuition for analyzing entrance costs generalize when the good join rate is changing, possibly within each iteration of ERGO? To handle this, we first show that our estimate of the good join rate updates at least once every $O(1)$ epochs, so it is never too stale. Next, we make use of Cauchy-Schwartz to upper-bound ERGO's total cost based on the bounds for each iteration.

V. GOODJEST

GOODJEST provides an estimate, $\tilde{\mathtt{J}}$, of the good join rate, when there is at most a constant fraction of bad IDs; for simplicity, we assume this fraction is less than $1/6$, but larger constants can be tolerated.

Initially, GOODJEST sets $\tilde{\mathtt{J}}$ equal to the number of IDs at system initialization divided by the total time taken for initialization, where initialization consists of the server issuing a 1-hard RB-challenge to all nodes. In Section VIII,

we show how to decentralize this algorithm. The value t is set to the system start time. Throughout the protocol, t will equal the last time that $\tilde{\mathsf{J}}$ was updated, and t' will be the current time.

The pseudocode is presented in Figure 2. There are two aspects that must be addressed in designing GOODJEST. First, at what points in time should $\tilde{\mathsf{J}}$ be updated? This occurs whenever the system membership has changed by a constant factor with respect to the current system size. In particular, $\tilde{\mathsf{J}}$ is updated when $|S(t') \triangle S(t)| \geq \frac{5}{8}|S(t')|$ holds true. Since join and departure events are serialized, this is equivalent to the property that $|S(t') \triangle S(t)| = \lceil \frac{5}{8}|S(t')| \rceil$. We refer to $(t, t']$ as an ***interval***. The execution of GOODJEST divides time into consecutive, disjoint intervals.

Second, how is $\tilde{\mathsf{J}}$ updated? This is done by setting $\tilde{\mathsf{J}}$ to the current system size divided by the amount of time since the last update to $\tilde{\mathsf{J}}$. In particular, we set $\tilde{\mathsf{J}} \leftarrow |S(t')|/(t' - t)$.

VI. ANALYSIS

A. Analysis Overview of GOODJEST

Why does GOODJEST provide a close estimate of the good join rate? Recall that GOODJEST divides time into intervals. We say that an interval ***intersects*** an epoch if there is a point in time belonging to both the interval and the epoch. We now sketch the analysis; complete proofs are in [43]. The proofs of the next two lemmas follow easily from definitions.

Lemma 5. *An interval intersects at most two epochs.*

Lemma 6. $|S(t')| \geq \frac{6}{13}|S(t)|$.

For the remainder of this section, fix an interval that starts at time $\boldsymbol{t}$ and ends at time $\boldsymbol{t'}$. Let $\boldsymbol{a}$ be the number of good IDs that have joined during the interval. All lemmas hold with high probability in n_0.

The next lemma is one of the more technically challenging in our analysis. Recall that $S(\tau)$ is the set of all IDs in the system at time τ. In order to upper bound the number of joining good IDs, we need to first upper bound the number of new, good IDs that depart, where an ID is new if it has joined in the current interval. The key technical difficulty is establishing this bound with high probability. To do so, we compute the expected number of departing new, good IDs, and then use a stochastic dominance argument and Chernoff bounds to show tight concentration around this expectation.

Lemma 7. $a \leq 23|S(t)| + 4$.

Proof: Note that:

$$\left\lceil \frac{5}{8}|S(t')| \right\rceil = |S(t') \triangle S(t)| \geq |G(t') - G(t)|$$

where the first step holds by the definition of an interval and the fact that all join and leave events occur at unique times. The next step holds since sets of good and bad IDs are disjoint. Thus, we have:

$$|G(t') - G(t)| \leq \left\lceil \frac{5}{8}|S(t')| \right\rceil \leq \frac{3}{4}|G(t')| + 1$$

In the above, the second step holds since the fraction of bad IDs is always less than $1/6$. Hence, $\frac{|G(t')|}{|S(t')|} > \frac{5}{6}$ implies that $|S(t')| < \frac{6}{5}|G(t')|$. Then:

$$\left\lceil \frac{5}{8}|S(t')| \right\rceil \leq \frac{5}{8}|S(t')| + 1 < \frac{3}{4}|G(t')| + 1$$

This gives our first key inequality:

$$|G(t') - G(t)| < \frac{3}{4}|G(t')| + 1 \tag{1}$$

Let d be the number of good IDs that have departed in the interval. Let the random variable X be the number of IDs in $G(t)$ that have departed during the interval. Note that X is stochastic since, when a good ID departs at any time $t' \geq t$, the probability that it is an ID from G(t) equals $|G(t) \cap G(t')/|G(t')|$. By Equation 1, $E(X) \geq \frac{9}{40}d$, for $|G(t')|/40 \geq 1$, or $n_0 \geq 40$. Additionally, X stochastically dominates a simpler random variable that counts the number of successes when there are d independent trials, each succeeding with probability $\frac{9}{40}$. Hence, by Chernoff bounds, when $d \geq |G(t)|$, $X \geq \frac{1}{5}d$, with probability of failure that is $O(e^{-cn_0})$, for some constant $c > 0$. For any fixed λ, this probability is at most $n_0^{-\lambda-1}$ for n_0 sufficiently large. Hence, by a union bound, $X \geq \frac{1}{5}d$ over all intervals, with probability of failure at most $1/n_0$.

Clearly, $X \leq |G(t)|$. So by the above, we have that, with high probability, $\frac{1}{5}d \leq |G(t)|$, which gives:

$$d \leq 5|G(t)| \tag{2}$$

Since the number of new good IDs in $S(t')$ is at least $a - d$, then $|G(t') - G(t)| \geq a - d$. Thus:

$$\begin{aligned} a &\leq |G(t') - G(t)| + d \\ &\leq \left(\frac{3}{4}|G(t')| + 1\right) + 5|G(t)| \\ &\leq \frac{3}{4}\left(|G(t)| + a\right) + 1 + 5|G(t)| \\ &\leq \frac{23}{4}|G(t)| + \frac{3}{4}a + 1 \end{aligned}$$

In the above, the second step follows by applying inequalities 1 and 2, and the third step by noting that $|G(t')| \leq |G(t)| + a$. Finally, the lemma follows by isolating a in the last inequality, to get $a \leq 23|G(t)| + 4 \leq 23|S(t)| + 4$. ■

Lemma 8. $a \geq \frac{|S(t')|}{12(1+\beta^2)} - 2 \geq 8$.

Proof: Let d be the number of good IDs that have departed in the interval. We start by proving that:

$$d \leq \beta^2(a + 2) + 2. \tag{3}$$

By Lemma 5, an interval intersects at most two epochs. If two epochs are intersected, let ρ, ρ' be the good join rates

over the two epochs intersected, and ℓ, ℓ' be the lengths of the intersection. If a single epoch is intersected, let ρ and ρ' both equal the good join rate over that epoch, and let ℓ, ℓ' both be half the length of the intersection of the interval and the epoch. Then, in every case, from $\beta-$smoothness, we have:

$$a \geq \left\lfloor \frac{\rho\ell}{\beta} \right\rfloor + \left\lfloor \frac{\rho'\ell'}{\beta} \right\rfloor \geq \frac{\rho\ell + \rho'\ell'}{\beta} - 2$$

For which:

$$\rho\ell + \rho'\ell' \leq \beta(a+2) \tag{4}$$

We can bound the number of departures using $\beta-$smoothness:

$$d \leq \lceil \beta\rho\ell \rceil + \lceil \beta\rho'\ell' \rceil \leq \beta(\rho\ell + \rho'\ell') + 2 \leq \beta^2(a+2) + 2$$

where the last step follows from Inequality 4, and this yields Equation 3. Next, note that:

$$\begin{aligned} |G(t') \triangle G_t| &= |G(t') - G(t)| + |G(t) - G(t')| \\ &\leq a + d \\ &\leq a + \beta^2(a+2) + 2 \\ &\leq (1+\beta^2)(a+2) \end{aligned} \tag{5}$$

where the second to last step follows from Equation 3.

Since the sets of good and bad IDs are disjoint, we have $|S(t') \triangle S(t)| = |G(t') \triangle G_t| + |B(t') \triangle B(t)|$. Rearranging, we get:

$$\begin{aligned} |G(t') \triangle G_t| &= |S(t') \triangle S(t)| - |B(t') \triangle B(t)| \\ &\geq \frac{5}{8}|S(t')| - |B(t') \triangle B(t)| \\ &= \frac{5}{8}|S(t')| - (|B(t') - B(t)| + |B(t) - B(t')|) \\ &\geq \frac{5}{8}|S(t')| - \frac{|S(t')|}{6} - \frac{|S(t)|}{6} \\ &\geq \frac{5}{8}|S(t')| - \frac{|S(t')|}{6} - \frac{13}{6}\left(\frac{|S(t')|}{6}\right) \\ &\geq \frac{|S(t')|}{12} \end{aligned} \tag{6}$$

The second step follows from the definition of an interval. The third step follows from the definition of symmetric difference. The fourth step follows by the assumption that the fraction of bad IDs is always less than $1/6$, so $|B(t') - B(t)| \leq \frac{|S(t')|}{6}$ and $|B(t) - B(t')| \leq \frac{|S(t)|}{6}$. The fifth step follows from Lemma 6.

Combining Inequality 5 and Inequality 6, we get:

$$(1+\beta^2)(a+2) \geq \frac{|S(t')|}{12}$$

On isolating a in the above, we get:

$$a \geq \left(\frac{1}{12(1+\beta^2)}\right)|S(t')| - 2$$

Substituting $\beta \leq \sqrt{\frac{n_0}{120} - 1}$ from Definition 4, we get:

$$a \geq \left(\frac{1}{12(1 + (\frac{n_0}{120} - 1))}\right)|S(t')| - 2 \geq \left(\frac{10}{n_0}\right)n_0 - 2 = 8$$

where the second step holds since $|S(t')| \geq n_0$. ■

We bound $\tilde{\mathtt{J}}$ with respect to the good join rate over the interval.

Lemma 9. *Let $\tilde{\mathtt{J}}$ be the estimated join rate at the end of the interval and $\mathtt{J}$ be the good join rate during the interval. Then, $(1/100)\mathtt{J} \leq \tilde{\mathtt{J}} \leq 30\beta^2\mathtt{J}$.*

Proof: At the end of the interval, GOODJEST sets $\tilde{\mathtt{J}}$ equal to:

$$\frac{|S(t')|}{t'-t} \geq \frac{6}{13}\left(\frac{|S(t)|}{t'-t}\right) \geq \frac{6}{13}\left(\frac{(a-4)/23}{t'-t}\right) \geq \frac{1}{50}\left(\frac{a - a/2}{t'-t}\right) \geq \frac{\mathtt{J}}{100}$$

The second step follows from Lemma 6, the third step from Lemma 7 using $a \leq 23|S(t)| + 4$, and the fourth step from Lemma 8 using $a \geq 8$. Similarly, $\tilde{\mathtt{J}}$ equals:

$$\frac{|S(t')|}{t'-t} \leq \frac{12(1+\beta^2)(a+2)}{(t'-t)} \leq 12(1+\beta^2)\left(\frac{5a}{4(t'-t)}\right) \leq 30\beta^2\mathtt{J}$$

The second and third steps follow from Lemma 8 using $a \geq \frac{|S(t')|}{12(1+\beta^2)} - 2$ and $a \geq 8$; the last step holds since $\beta \geq 1$. ■

The next lemma follows from Lemma 5 and the bound $a \geq 8$ from Lemma 8.

Lemma 10. *Consider an epoch that intersects any interval. Suppose ρ is the good join rate over the epoch, and $\mathtt{J}$ is the good join rate over the interval. Then, $\frac{4}{5\alpha\beta}\rho \leq \mathtt{J} \leq \frac{8}{3}\alpha\beta\rho$.*

The next lemma makes use of Lemma 5, and the bounds in Lemmas 9 and 10.

Lemma 11. *Let $\tilde{\mathtt{J}}$ be the estimated join rate at the end of the interval and $\mathtt{J}$ be the join rate during the* next *interval. Then, $1/\left(334\alpha^3\beta^2\right)\mathtt{J} \leq \tilde{\mathtt{J}} \leq 100\alpha^3\beta^4\mathtt{J}$.*

Now we give the proof of Theorem 2.

Proof: Fix a time step τ in the interval. Let ρ_τ be the good join rate over the epoch containing τ. Combining the bounds on $\tilde{\mathtt{J}}$ from Lemma 11 with those from Lemma 10,we get:

$$\left(\frac{4}{1670\alpha^4\beta^3}\right)\rho_\tau \leq \tilde{\mathtt{J}} \leq 100\alpha^3\beta^4\left(\frac{8}{3}\alpha\beta\right)\rho_\tau$$

Simplifying this equation yields the result. ■

B. Analysis of ERGO

The proof that ERGO ensures less than a $1/6$ fraction of bad IDs follows directly from the number of joins and departures allowed by ERGO between purges, see [43] for the proof.

Here, we sketch the bound on the good spend rate. To begin, we partition every interval of length ℓ into $\lceil \ell\tilde{\mathtt{J}} \rceil$ ***sub-intervals*** of length at most $1/\tilde{\mathtt{J}}$, where $\tilde{\mathtt{J}}$ is the estimate of

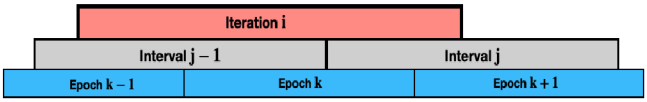

Figure 3: *Epochs* derive from our model of churn (Section II-A). *Intervals* derive from GOODJEST, specifically the times at which it sets the variable $\tilde{\mathtt{J}}$ (Figure 2, Step 1). *Iterations* derive from ERGO, specifically the duration between purges (Figure 1, Step 2).

the good join rate used in the interval. Then we have the following lemmas.

Lemma 12. *Fix a sub-interval j. Let $\mathcal{T}_j$ be the total spending of the adversary in sub-interval j. Then, the number of bad IDs that join in this sub-interval is at most $\sqrt{2\mathcal{T}_j}$.*

Proof: Let b_j be the number of bad IDs joining in the sub-interval j. Then, pessimistically assuming all bad IDs join before any good IDs in a sub-interval, we get:

$$\mathcal{T}_j \geq \sum_{i=1}^{b_j} i \geq \frac{b_j^2}{2}$$

Solving the above for b_j, we obtain the result. ■

Lemma 13. *An iteration intersects at most two intervals.*

Proof: We prove this by contradiction. Assume an iteration starts at time t_0 and intersects three or more intervals. Then, there will be at least one interval that is completely contained within the iteration. Let the first such interval start at time $t_1 \geq t_0$ and end at time $t_2 > t_1$. Let $n^a(n^d)$ be the number of IDs that join (depart) during this interval. Then:

$$n^a + n^d \geq |S(t_1) \triangle S(t_2)| \geq \frac{5}{8}|S(t_2)| \geq \frac{5}{8}\left(\frac{10}{11}|S(t_0)|\right)$$
$$= \frac{25}{44}|S(t_0)|$$

The second step follows from the definition of an interval. The third step holds since during an iteration, at most $|S(t_0)|/11$ IDs can depart, and so the system size at time t_2 is at least $\frac{10}{11}|S(t_0)|$. But the number of joins and departures during the iteration is at most $|S(t_0)|/11$ by the definition of an iteration. This gives the contradiction. ■

Lemma 14. *Fix an iteration. Let $\mathcal{L}$ be the length, and $\mathcal{J}$ be the good join rate in this iteration. Then, the number of sub-intervals in the iteration is at most $100\alpha^3\beta^6(\mathcal{J}\mathcal{L}+10)$.*

Proof: From Lemma 13, an iteration intersects at most two intervals. For $i \in \{1, 2\}$, let t_i denote time at which the i^{th} interval intersects the iteration for the first time; $\mathtt{J}_i$ be the good join rate in the i^{th} overlapping interval and $\tilde{\mathtt{J}}_i$ be the estimated good join rate set at the end of interval i. If there is only one interval intersected, let $\mathtt{J}_1 = \mathtt{J}_2$, $\tilde{\mathtt{J}}_1 = \tilde{\mathtt{J}}_2$ and $t_1 = t_2$.

By Lemma 5, an interval intersects at most two epochs.

So, let ρ_1 and ρ_2 be the join rate of good IDs over the two epochs that intersect with interval 1, and let ℓ_1 and ℓ_2, respectively be the lengths of their intersection, with $\ell_2 = 0$ if there is only one such epoch. Similarly, let ρ_3 and ρ_4 be the join rate of good IDs over the two epochs that intersect with interval 2, and let ℓ_3 and ℓ_4, respectively be the lengths of their intersection, with $\ell_4 = 0$ if there is only one such epoch. Then, from the β-smoothness property, we have:

$$\mathcal{J}\mathcal{L} \geq \sum_{k=1}^{4}\left\lfloor\frac{\rho_k\ell_k}{\beta}\right\rfloor \geq \sum_{k=1}^{4}\left(\frac{\rho_k\ell_k}{\beta} - 1\right) = \frac{1}{\beta}\sum_{k=1}^{4}\rho_k\ell_k - 4 \quad (7)$$

Next, let t_0 denote the time at the start of the iteration. Then, the number of sub-intervals in the iteration is:

$$\sum_{i=1}^{2}\lceil(t_i - t_{i-1})\tilde{\mathtt{J}}_i\rceil \leq 100\alpha^3\beta^4\sum_{i=1}^{2}((t_i - t_{i-1})\mathtt{J}_i + 1)$$
$$\leq 100\alpha^3\beta^4\left(2 + \sum_{k=1}^{4}\lceil\beta\rho_k\ell_k\rceil\right)$$
$$\leq 100\alpha^3\beta^5\left(2 + \sum_{k=1}^{4}\rho_k\ell_k + 4\right)$$
$$\leq 100\alpha^3\beta^6(\mathcal{J}\mathcal{L} + 10)$$

In the above, the first step follows from Lemma 11; the second step follows from β-smoothness and Lemma 5; the third step holds since $\beta \geq 1$; and the last step from inequality 7 by isolating the value of $\sum_{k=1}^{4}\rho_k\ell_k$, since $\beta \geq 1$. ■

Lemma 15. *The number of good IDs that join over any sub-interval is at most $418\alpha^4\beta^4 + 1$.*

Proof: Let $\tilde{\mathtt{J}}$ be the estimate of the good join rate in the interval containing the sub-interval, and let ρ be the good join rate over the epoch that contains the sub-interval. Then, by β-smoothness, the number of good IDs that join over the sub-interval is at most:

$$\left\lceil\beta\rho\left(\frac{1}{\tilde{\mathtt{J}}}\right)\right\rceil \leq \beta\rho\left(418\alpha^4\beta^3\left(\frac{1}{\rho}\right)\right) + 1$$
$$\leq 418\alpha^4\beta^4 + 1$$

In the above, the first step follows from Theorem 2. ■

Lemma 16. *Suppose that u and v are x-dimensional vectors in Euclidean space. For all $x \geq 1$:*

$$\sum_{j=1}^{x}\sqrt{u_j v_j} \leq \sqrt{\sum_{j=1}^{x}u_j\sum_{j=1}^{x}v_j}$$

Proof: Using the Cauchy-Schwarz inequality [45], we have:

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

The result follows by taking the square-root of both sides. ■

Lemma 17. *Fix an iteration. Let $\mathcal{L}$ be the length of this iteration, $\mathcal{J}$ be the join rate of good IDs in the iteration, and $\mathcal{T}$ be the total resource cost to the adversary during the iteration. Then, the total entrance cost to good IDs during the iteration is:*

$$O\left(\alpha^{11/2}\beta^7\sqrt{(\mathcal{J}\mathcal{L}+1)\mathcal{T}}+\alpha^{11}\beta^{14}\mathcal{J}\mathcal{L}\right).$$

Proof: Fix a sub-interval j of the iteration. Let g_j (b_j) be the number of good (bad) IDs that join in sub-interval j, and $\mathcal{T}_j$ be the resource cost to the adversary in sub-interval j. Pessimistically assuming all good IDs enter at the end of the sub-interval, the total entrance cost to good IDs in sub-interval j is at most:

$$\begin{aligned}\sum_{k=1}^{g_j}(b_j+k) &\le g_j\left(\sqrt{2\mathcal{T}_j}+g_j\right)\\ &\le (418\alpha^4\beta^4+1)\left(\sqrt{2\mathcal{T}_j}+418\alpha^4\beta^4+1\right)\end{aligned}$$

The first step follows from Lemma 12, and the second step follows from Lemma 15.

Suppose the iteration consists of t sub-intervals. Then, the total entrance cost to the good IDs in the iteration is:

$$\begin{aligned}&\sum_{j=1}^{t}\left((418\alpha^4\beta^4+1)\left(\sqrt{2\mathcal{T}_j}+(418\alpha^4\beta^4+1)\right)\right)\\ &\quad\le (418\alpha^4\beta^4+1)\sqrt{2t\mathcal{T}}+(418\alpha^4\beta^4+1)^2t\\ &\quad= O\left(\alpha^{11/2}\beta^7\sqrt{(\mathcal{J}\mathcal{L}+1)\mathcal{T}}+\alpha^{11}\beta^{14}\mathcal{J}\mathcal{L}\right)\end{aligned}$$

The first step follows from Lemma 16 and by noting that $\sum_{j=1}^{t}\mathcal{T}_j=\mathcal{T}$. The second step follows from using Lemma 14 to upperbound t. ■

Lemma 18. *Fix an iteration. For this iteration, let $\mathcal{L}$ be the length, $\mathcal{D}$ be the rate of departure, $\mathcal{J}$ be the join rate of good IDs, and $\mathcal{T}$ be the total RB-cost to the adversary. Then, the total spending for good IDs in this iteration is:*

$$O\left(\mathcal{D}\mathcal{L}+\alpha^{11/2}\beta^7\sqrt{(\mathcal{J}\mathcal{L}+1)\mathcal{T}}+\alpha^{11}\beta^{14}\mathcal{J}\mathcal{L}\right).$$

Proof:

Let S be the set of IDs at the beginning of iteration. For the iteration, let t be the number of sub-intervals; g and b be the number of good and bad IDs that join, and d be the total number of IDs that depart. For any sub-interval j of the iteration, let $\mathcal{T}_j$ be the total RB-cost to the adversary in that sub-interval.

Each good ID solves a 1-hard RB-challenge during purges. Hence the cost due to purges is at most the number of good IDs at the end of the iteration, which is at most:

$$\begin{aligned}\frac{12}{11}|S| &\le \frac{12}{11}\left(11\left(d+b+g\right)\right)\\ &\le 12\left(D\mathcal{L}+\sum_{j=1}^{t}\sqrt{2\mathcal{T}_j}+\mathcal{J}\mathcal{L}\right)\\ &\le 12\left(D\mathcal{L}+\sqrt{2t\sum_{j=1}^{t}\mathcal{T}_j}+\mathcal{J}\mathcal{L}\right)\\ &\le 12\left(D\mathcal{L}+\sqrt{200\alpha^3\beta^6(\mathcal{J}\mathcal{L}+10)\mathcal{T}}+\mathcal{J}\mathcal{L}\right) \quad (8)\end{aligned}$$

In the above, the first step follows since over an iteration the number of good IDs in the system can increase by at most $|S|/11$. The second step follows since the number of ID joins and deletions in an iteration, i.e. $d+b+g$, is at least $|S|/11$ (Step 2 of ERGO). The third step follows by upper bounding b using Lemma 12 to bound the number of bad IDs joining over all sub-intervals; and noting that $g=\mathcal{J}\mathcal{L}$ and $b=\mathcal{D}\mathcal{L}$. The fourth step follows from Lemma 16. The last step follows from Lemma 14 and substituting $\sum_{j=1}^{t}\mathcal{T}_j=\mathcal{T}$. Combining Equation 8 with the cost from Lemma 17 yields the result. ■

Consider a long-lived system which undergoes an attack over some limited number of consecutive iterations. A resource bound over the period of attack, rather than over the lifetime of the system, is stronger, and may be of additional value to practitioners. Thus, we first provide this type of guarantee in Lemma 19; Theorem 1 then becomes a simple corollary of this lemma, when considered over all iterations.

For the following lemma, let $\boldsymbol{\mathcal{I}}$ be a subset of contiguous iterations containing all iterations numbered between x and y inclusive, for any x and y, $1\le x\le y$. Let $\boldsymbol{\delta(\mathcal{I})}$ be $|S_x-S_y|$; and let $\boldsymbol{\Delta(\mathcal{I})}$ be $\delta(\mathcal{I})$ divided by the length of $\mathcal{I}$. We note that in the proof of Theorem 1, $\Delta(\mathcal{I})$ will be 0. Let $\boldsymbol{T_{\mathcal{I}}}$ be the adversarial spend rates over $\mathcal{I}$; and let $\boldsymbol{J_{\mathcal{I}}}$ be the good join rate over $\mathcal{I}$. Then we have the following lemma.

Lemma 19. *For any subset of contiguous iterations, $\mathcal{I}$, starting after iteration 1, the good spend rate over $\mathcal{I}$ is:*

$$O\left(\Delta(\mathcal{I})+\alpha^{11/2}\beta^7\sqrt{(J_{\mathcal{I}}+1)T_{\mathcal{I}}}+\alpha^{11}\beta^{14}J_{\mathcal{I}}\right).$$

Proof: For all iterations $i\in\mathcal{I}$, let $\mathcal{L}_i$ be the length of iteration i, $\mathcal{J}_i$ be the good join rate in iteration i, $\mathcal{D}_i$ be the good departure rate in iteration i, and T_i be the adversarial resource spend rate in iteration i. Then, by Lemma 18, for some constant c, we have the total spending of the good IDs over all iterations in $\mathcal{I}$ is at most:

$$\sum_{i\in\mathcal{I}}c\left(D_i\mathcal{L}_i+\alpha^{11/2}\beta^7\sqrt{(\mathcal{J}_i\mathcal{L}_i+1)T_i\mathcal{L}_i}+\alpha^{11}\beta^{14}\mathcal{J}_i\mathcal{L}_i\right)$$

Dividing this by $\sum_{i\in\mathcal{I}} \mathcal{L}_i$ and using Lemma 16, we get:

$$\frac{c\sum_{i\in\mathcal{I}} D_i\mathcal{L}_i}{\sum_{i\in\mathcal{I}}\mathcal{L}_i} + c\alpha^{11/2}\beta^7\sqrt{\left(\frac{\sum_{i\in\mathcal{I}}(\mathcal{J}_i\mathcal{L}_i+1)}{\sum_{i\in\mathcal{I}}\mathcal{L}_i}\right)\frac{\sum_{i\in\mathcal{I}} T_i\mathcal{L}_i}{\sum_{i\in\mathcal{I}}\mathcal{L}_i}}$$
$$+ c\alpha^{11}\beta^{14}\frac{\sum_{i\in\mathcal{I}}\mathcal{J}_i\mathcal{L}_i}{\sum_{i\in\mathcal{I}}\mathcal{L}_i}$$
$$= O\left(\Delta(\mathcal{I}) + \alpha^{11/2}\beta^7\sqrt{(J_\mathcal{I}+1)T_\mathcal{I}} + \alpha^{11}\beta^{14}J_\mathcal{I}\right)$$

which yields the result. ■

Proof of Theorem 1: The resource cost bound follows immediately from Lemma 19 by noting that $\Delta(\mathcal{I}) = 0$ when $\mathcal{I}$ is all iterations, since the system is initially empty. Then, Lemma 12 from [43] completes the proof, by showing that the fraction of bad is always less than $1/6$. ■

VII. Experiments

We now report on several empirical contributions. First, in Section VII-A, we propose and implement numerous heuristics for ERGO. Second, we measure the resource burning cost for ERGO, as a function of the adversarial cost, and compare it against prior results. Third, in Section VII-B, we evaluate the performance of the GOODJEST algorithm by measuring the approximation factor for the join rate of good IDs. All our experiments were written in MATLAB.

We use churn data from the following networks:

• **Bitcoin.** This dataset records the join and departure events of IDs in the Bitcoin network, timestamped to the second, over roughly 7 days [70].

• **BitTorrent.** This dataset simulates the join and departure events for the BitTorrent network to obtain a RedHat ISO image. We use the Weibull distribution with shape and scale parameters of 0.59 and 41.0, respectively, from [79].

• **Ethereum.** This dataset simulates join and departure events of IDs for the Ethereum network. Based on a study in [51], we use the Weibull distribution with shape parameter of 0.52 and scale parameter of 9.8.

• **Gnutella.** This dataset simulates join and departure events for the Gnutella network. Based on a study in [73], we use an exponential distribution with mean of 2.3 hours for session time, and Poisson distribution with mean of 1 ID per second for the arrival rate.

A. Evaluating ERGO

ERGO may incorporate other tools in order to improve its performance, while preserving the guarantees of Theorem 1. For example, several recent works have explored the possibility of identifying bad IDs based on the network topology [31], [62]. We employ the machine-learning result, SybilFuse, which correctly classifies an ID as good or bad with probability 0.98 based on the empirical results from Gao et al. [31] (specifically Section IV-B, last paragraph). ERGO functions as before with the modification that SybilFuse is used to diagnose whether a joining ID is good or bad; in the latter case, the ID is refused entry. Several other heuristics are explored in the full version of this work [43]. However, the above heuristic delivers the best performance gains.

We compare the performance of ERGO against four resource burning based Sybil defense algorithms: CCOM [41], SYBILCONTROL [53], REMP (a name that uses the authors' initials) [73] and ERGO-SF, summarized below.

• **CCOM.** CCOM is the same as ERGO except that the hardness of RB-challenge assigned to joining IDs is always set to 1.

• **SYBILCONTROL.** Each ID solves a RB-challenge to join. Additionally, each ID tests its neighbors with a RB-challenge every 0.5 seconds, removing from its list of neighbors those IDs that fail to provide a solution within a fixed time period. These tests are not coordinated between IDs.

• **REMP.** Each ID solves a RB-challenge to join. Additionally, each ID must solve RB-challenges every $\boldsymbol{W}$ seconds. We use Equation (4) from [73] to compute the value of spend rate per ID as $\frac{L}{W} = \frac{n}{N_{attacker}} = \frac{T_{max}}{\kappa N}$, where L is the cost to an ID per W seconds, n is the number of IDs that the adversary can add to the system and $N_{attacker}$ is the total number of attackers in the system. The total good spend rate is:

$$\mathcal{A}_{REMP} = (1-\kappa)N \times \frac{L}{W} = \frac{(1-\kappa)T_{max}}{\kappa} \qquad (9)$$

to guarantee that the fraction of bad IDs is less than half.

• **ERGO-SF.** This is ERGO using SybilFuse (SF) with accuracy parameter as 0.98.

Setup. For all algorithms, we measure the spend-rate, which is based solely on the cost of solving RB-challenges. We assume a cost of k for solving a k hard RB-challenge. We set $\kappa = 1/18$, and let T range over $[2^0, 2^{20}]$, where for each value of T, the system is simulated for $10,000$ seconds. We assume that the adversary only solves RB-challenges to add IDs to the system. For REMP, we consider $T_{max} = 10^7$ to ensure correctness for all values of T considered.

Results. Figure 4 illustrates our results; we omit error bars since they are negligible. The x-axis is the adversarial spend rate, T; and the y-axis is the good spend rate, $\mathcal{A}$.

We cut off the plot of SYBILCONTROLwhen the algorithm can no longer ensure that the fraction of bad IDs is less than $1/6$. We also note that REMP-10^7 only ensures a minority of bad IDs for up to $T = 10^7$.

ERGO always has a spend rate as low as the other algorithms for $T \geq 100$, and significantly less than the other algorithms for large T, with improvements that grow to about 2 orders of magnitude. Our heuristic improves further, allowing ERGO to outperform for all $T \geq 0$. This is illustrated by ERGO-SF, which reduces costs significantly,

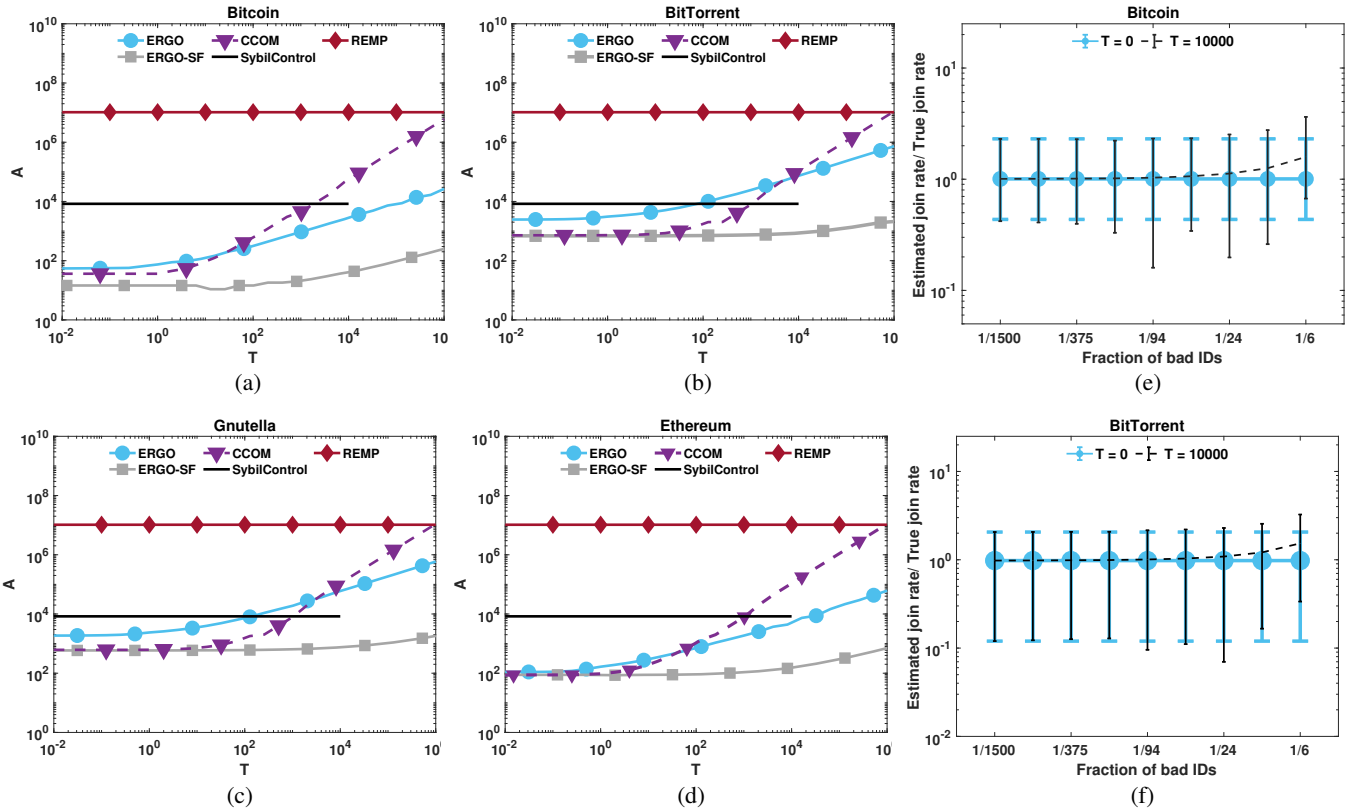

Figure 4: Plots (a) – (d) illustrate the good spend rate (A) versus adversarial spend rate (T). Plots (e) and (f) depict the ratio of GOODJEST estimated to the true join rate for good IDs versus fraction of bad IDs.

yielding improvements of up to three orders of magnitude during the most significant attack tested. The spend rate for ERGO is linear in $\sqrt{T}$, agreeing with our theoretical analysis. We emphasize that the benefits of ERGO are consistent over four disparate networks.

B. *Evaluating* GOODJEST

For the Bitcoin network, the system initially consists of 9212 IDs, and the join and departure events are based off the dataset from Neudecker et al. [71]. For the Bitcoin network, the system is initialized with 10,000 IDs, and join and departure events are simulated over 100,000 time steps. Results for Ethereum and Gnutella are presented in [43].

In our simulations, all joins and departures from the data sets are assumed to be good IDs. The fraction of bad IDs varies over the values $\{1/1500, 1/375, 1/94,\ 1/24, 1/6\}$. Furthermore, the adversary injects additional bad IDs at a constant rate that can be afforded when $T = 10,000$. For every interval, we measure the ratio of the estimate from GOODJEST to the actual good join rate.

We report our results in Figure 4. These plots demonstrate the robustness of GOODJEST. When $T = 0$, our estimate is always within range $(0.08, 1.2)$ of the actual good join rate. Moreover, even when $T = 10,000$, our estimate is always within range $(0.08, 4)$ of the actual good join rate.

VIII. DECENTRALIZATION

When there is no server, we can run our algorithms using a ***committee***: an $O(\log n_0)$ sized subset of IDs with a good majority. This committee takes over the responsibilities of the server, which means the committee runs GOODJEST and ERGO in a robust, distributed fashion. Below, we discuss the necessary model and algorithmic modifications.

A. *Model Modifications*

All communication among good IDs uses a broadcast primitive, DIFFUSE, which allows a good ID to send a value to all other good IDs within a known and bounded amount of time, despite an adversary. Such a primitive is a standard assumption in PoW schemes [22], [32], [34], [55]; see [60] for empirical justification. The committee uses DIFFUSE to issue RB-challenges; other IDs use DIFFUSE to answer challenges.

Here, a ***round*** is the amount of time it takes to solve a 1-hard RB-challenge *plus* the time for communication between the committee and corresponding ID in order to issue the challenge and returning a solution. In any round, up to a constant fraction of the good IDs may depart. This is necessary to ensure that not too many good IDs leave the committee before they can be replaced.

B. Committee and System Initialization

GENID. To initialize our system, we require a solution to GENID (recall Section II-A). GENID guarantees that at initialization, (1) all good IDs agree on the same set of IDs; and (2) at most a κ-fraction of IDs in that set are bad. Additionally, GENID ensures that all good IDs agree on a committee of logarithmic size with a majority of good IDs. There are several algorithms that solve GENID in our model, as defined in Section II [3], [5], [46], [49]. The algorithm in [3], makes use of computational challenges, and runs in expected $O(1)$ rounds, and, in expectation, requires each good ID to send $O(n)$ bits, and solve $O(1)$ 1-hard RB-challenges.

C. Use and Maintenance of Committee

The committee uses ***State Machine Replication (SMR)*** (see [1], [7], [21], [56]) to agree on an ordering of network events so as to execute GOODJEST and ERGO in parallel. To run SMR, the following invariant must be maintained.

Committee Invariant: *There always exists a committee known to all good IDs. This committee has size $\Theta(\log n_0)$, and a majority of good IDs.*

To maintain this invariant, a new committee is elected by the old committee at the end of each iteration. In particular, at the end of iteration i, the old committee selects a committee of size $C \log |S_i|$, where $C > 1$ is a sufficiently large constant. This committee selection process can be accomplished in our model via classic secure multiparty computation protocols; for example, see Rabin and Ben-Or [72]. Note that subsequent results can accomplish the same task more efficiently, but require cryptographic assumptions. For example, Awerbuch and Scheideler [14] describe an algorithm specifically for random number generation that can be used by the existing committee to select new committee members.

The algorithms of [1] and [72] work in our model (Section II), assuming at most a $1/6$ fraction of Byzantine IDs. With the modifications above we are able to maintain the committee invariant. See [43] for proofs.

IX. CONCLUSION

ERGO is a new Sybil-defense that efficiently employs resource-burning to limit Sybil IDs, despite high churn. Our experiments show that ERGO significantly decreases resource costs when compared to other Sybil defenses.

Many open problems remain. First, can we apply the results in this paper to build and maintain a Sybil-resistant distributed hash table (DHT) [80]? To the best of our knowledge, there is no such result that ensures the good IDs pay a cost that is a slowly growing function of both the good churn rate and the cost paid by an attacker. Second, can we improve the costs in this paper? In [40], there is a lower bound that asymptotically matches when α and β are both constants. However, this lower bound only holds for a certain class of algorithms. Can we show a lower bound for arbitrary algorithms and any values of α and β?

Acknowledgements. This work is supported by the National Science Foundation grants CNS 1816076 and 1816250.

REFERENCES

[1] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and M. Yin. Sync HotStuff: Simple and Practical Synchronous State Machine Replication. *IACR Cryptology ePrint Archive*, 2019.

[2] A. Aggarwal, V. Dani, T. Hayes, and J. Saia. Secure One-Way Interactive Communication. In 15^{th} *International Conference on Distributed Computing and Networking (ICDCN)*, 2017.

[3] A. Aggarwal, M. Movahedi, J. Saia, and M. Zamani. Bootstrapping Public Blockchains Without a Trusted Setup. In *Proc. of 2019 ACM Symposium on Principles of Distributed Computing*, 2019.

[4] 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/.

[5] M. Andrychowicz and S. Dziembowski. PoW-based distributed cryptography with no trusted setup. In *Proc. of Annual Cryptology Conference*. Springer, 2015.

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

[7] H. Attiya and J. Welch. *Distributed computing: Fundamentals, Simulations, and Advanced Topics*, volume 19. John Wiley & Sons, 2004.

[8] J. Augustine, A. R. Molla, E. Morsy, G. Pandurangan, P. Robinson, and E. Upfal. Storage and Search in Dynamic Peer-to-peer Networks. In 25^{th} *Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2013.

[9] J. Augustine, G. Pandurangan, and P. Robinson. Fast Byzantine Agreement in Dynamic Networks. In *ACM Symposium on Principles of Distributed Computing (PODC)*, 2013.

[10] J. Augustine, G. Pandurangan, and P. Robinson. Fast Byzantine Leader Election in Dynamic Networks. In *Intl. Symposium on Distributed Computing (DISC)*. Springer, 2015.

[11] J. Augustine, G. Pandurangan, P. Robinson, S. Roche, and E. Upfal. Enabling Robust and Efficient Distributed Computation in Dynamic Peer-to-Peer Networks. In *Proc. of IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 350–369, 2015.

[12] J. Augustine, G. Pandurangan, P. Robinson, and E. Upfal. Towards Robust and Efficient Computation in Dynamic Peer-to-peer Networks. In *Proc. of Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2012.

[13] B. Awerbuch and C. Scheideler. Group Spreading: A Protocol for Provably Secure Distributed Name Service. In 31^{st} *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2004.

[14] B. Awerbuch and C. Scheideler. Robust Random Number Generation for Peer-to-peer Systems. In *10th Intl. Conference On Principles of Distributed Systems (OPODIS)*, 2006.

[15] B. Awerbuch and C. Scheideler. Towards a Scalable and Robust DHT. In 18^{th} *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2006.

[16] B. Awerbuch and C. Scheideler. Towards Scalable and Robust Overlay Networks. In 6^{th} *Intl. Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

[17] H. S. Baird, M. A. Moll, and S.-Y. Wang. ScatterType: A legible but hard-to-segment CAPTCHA. In 8^{th} *International Conference on Document Analysis and Recognition*, 2005.

[18] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan. Proofs of Work from Worst-Case Assumptions. In *Annual International Cryptology Conference (CRYPTO)*. Springer, 2018.

[19] M. A. Bender, J. T. Fineman, S. Gilbert, and M. Young. How to Scale Exponential Backoff: Constant Throughput, Polylog Access Attempts, and Robustness. In 27^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2016.

[20] M. A. Bender, J. T. Fineman, M. Movahedi, J. Saia, V. Dani, S. Gilbert, S. Pettie, and M. Young. Resource-Competitive Algorithms. *SIGACT News*, 46(3), Sept. 2015.

[21] A. Bessani, J. Sousa, and E. E. Alchieri. State Machine Replication for the Masses with BFT-SMART. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 355–362. IEEE, 2014.

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

[23] CoinDesk. Vulnerable? Ethereum's Casper Tech Takes Criticism at Curacao Event. www.coindesk.com/fundamentally-vulnerable-ethereums-casper-tech-takes-criticism-curacao, 2018.

[24] G. Danezis, C. Lesniewski-Laas, M. F. Kaashoek, and R. Anderson. Sybil-Resistant DHT Routing. In 10^{th} *European Symposium On Research In Computer Security*, 2005.

[25] V. Dani, T. Hayes, M. Movahedi, J. Saia, and M. Young. Interactive Communication with Unknown Noise Rate. *Information and Computation*, 2017.

[26] V. Dani, M. Movahedi, J. Saia, and M. Young. Interactive Communication with Unknown Noise Rate. In *Colloquium on Automata, Languages, and Programming (ICALP)*, 2015.

[27] J. Douceur. The Sybil Attack. In *Proc. of Second International Peer-to-Peer Symposium (IPTPS)*, 2002.

[28] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *Proc. of* 12^{th} *Annual International Cryptology Conference on Advances in Cryptology*, 1993.

[29] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson. Profiling a Million User DHT. In 7^{th} *ACM SIGCOMM Conference on Internet Measurement*, 2007.

[30] A. Fiat, J. Saia, and M. Young. Making Chord Robust to Byzantine Attacks. In 13^{th} *European Symposium on Algorithms (ESA)*, 2005.

[31] P. Gao, B. Wang, N. Z. Gong, S. R. Kulkarni, K. Thomas, and P. Mittal. Sybilfuse: Combining Local Attributes with Global Structure to Perform Robust Sybil Detection. In *2018 IEEE Conference on Communications and Network Security (CNS)*, 2018. https://www.princeton.edu/~pmittal/publications/sybilfuse-cns18.pdf.

[32] J. Garay, A. Kiayias, and N. Leonardos. The Bitcoin Backbone Protocol: Analysis and Applications. In 34^{th} *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 2015.

[33] S. Gil, S. Kumar, M. Mazumder, D. Katabi, and D. Rus. Guaranteeing Spoof-Resilient Multi-Robot Networks. In *Robotics: Science and Systems*, 2015.

[34] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In 26^{th} *Symposium on Operating Systems Principles (SOSP)*, 2017.

[35] S. Gilbert, V. King, S. Pettie, E. Porat, J. Saia, and M. Young. (Near) Optimal Resource-Competitive Broadcast with Jamming. In 26^{th} *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014.

[36] S. Gilbert, V. King, J. Saia, and M. Young. Resource-Competitive Analysis: A New Perspective on Attack-Resistant Distributed Computing. In 8^{th} *ACM International Workshop on Foundations of Mobile Computing*, 2012.

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

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

[39] R. Guerraoui, F. Huc, and A.-M. Kermarrec. Highly Dynamic Distributed Computing with Byzantine Failures. In *Symposium on Principles of Distributed Computing (PODC)*, 2013.

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

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

[42] D. Gupta, J. Saia, and M. Young. Peace Through Superior Puzzling: An Asymmetric Sybil Defense. In *33rd IEEE Intl. Parallel and Distributed Processing Symposium*, 2019.

[43] D. Gupta, J. Saia, and M. Young. Bankrupting Sybil Despite Churn, 2020. https://arxiv.org/abs/2010.06834.

[44] D. Gupta, J. Saia, and M. Young. Invited paper: Resource burning for permissionless systems. In 27^{th} *International Conference on Structural Information and Communication Complexity (SIROCCO)*, 2020.

[45] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge University Press, 2012.

[46] R. Hou, I. Jahja, L. Luu, P. Saxena, and H. Yu. Randomized View Reconciliation in Permissionless Distributed Systems. *IEEE Intl. Conference on Computer Communications (INFOCOM)*, 2018.

[47] M. O. Jaiyeola, K. Patron, J. Saia, M. Young, and Q. M. Zhou. Tiny Groups Tackle Byzantine Adversaries. In *IEEE Intl. Parallel and Distributed Processing Symposium*, 2018.

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

[49] J. Katz, A. Miller, and E. Shi. Pseudonymous Secure Computation from Time-Lock Puzzles. *Citeseer*, 2014.

[50] A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In J. Katz and H. Shacham, editors, 37*th Annual International Cryptology Conference (CRYPTO)*, 2017.

[51] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey. Measuring Ethereum Network Peers. In *Internet Measurement Conference*, 2018.

[52] V. King, J. Saia, and M. Young. Conflict on a Communication Channel. In *Proc. of* 30^{th} *Symposium on Principles of Distributed Computing (PODC)*, 2011.

[53] F. Li, P. Mittal, M. Caesar, and N. Borisov. SybilControl: Practical Sybil Defense with Computational Puzzles. In 7^{th} *ACM Workshop on Scalable Trusted Computing*, 2012.

[54] 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), 2015.

[55] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A Secure Sharding Protocol For Open Blockchains. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2016.

[56] N. A. Lynch. *Distributed algorithms*. Elsevier, 1996.

[57] D. Malkhi. The BFT Lens: Hot-Stuff and Casper. https://dahliamalkhi.wordpress.com/2018/03/13/casper-in-the-lens-of-bft/, 2018.

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

[59] The Economist. Why Bitcoin Uses So Much Energy. www.economist.com/the-economist-explains/2018/07/09/why-bitcoin-uses-so-much-energy, 2018.

[60] A. Miller, J. Litton, A. Pachulski, N. Spring, N. Gupta, D. Levin, and B. Bhattacharjee. Discovering Bitcoin's Public Topology and Influential Nodes, 2015. http://cs.umd.edu/projects/coinscope/coinscope.pdf.

[61] D. L. Mills. Improved Algorithms for Synchronizing Computer Network Clocks. *IEEE/ACM Transactions on Networking*, 3(3), 1995.

[62] S. Misra, A. S. M. Tayeen, and W. Xu. SybilExposer: An Effective Scheme to Detect Sybil Communities in Online Social Networks. In *IEEE Intl. Conference on Communications (ICC)*, 2016.

[63] A. Mohaisen and S. Hollenbeck. Improving Social Network-Based Sybil Defenses by Rewiring and Augmenting Social Graphs. In 14^{th} *Intl. Workshop on Information Security Applications (WISA)*, 2014.

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

[65] D. Mónica, L. Leitao, L. Rodrigues, and C. Ribeiro. On the Use of Radio Resource Tests in Wireless Ad-Hoc Networks. In 3^{rd} *Workshop on Recent Advances on Intrusion-Tolerant Systems*, 2009.

[66] M. Moradi and M. Keyvanpour. CAPTCHA and its Alternatives: A Review. *Security and Communication Networks*, 8(12), 2015.

[67] T. Moran and I. Orlov. Simple Proofs of Space-Time and Rational Proofs of Storage. In *Annual Intl. Cryptology Conference*. Springer, 2019.

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

[69] T. Neudecker. Bitcoin Cash (BCH) Sybil Nodes on the Bitcoin Peer-to-Peer Network, 2017. http://dsn.tm.kit.edu/publications/files/332/bch_sybil.pdf.

[70] T. Neudecker. Personal correspondence, 2019.

[71] T. Neudecker, P. Andelfinger, and H. Hartenstein. Timing Analysis for Inferring the Topology of the Bitcoin Peer-to-Peer Network. In *Proc. of 13th IEEE International Conference on Advanced and Trusted Computing (ATC)*, July 2016.

[72] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In 21^{st} *annual ACM symposium on Theory of computing*, 1989.

[73] H. Rowaihy, W. Enck, P. McDaniel, and T. La Porta. Limiting Sybil attacks in Structured P2P Networks. In *IEEE Intl. Conference on Computer Communications (INFOCOM)*, 2007.

[74] C. Scheideler. How to Spread Adversarial Nodes? Rotate! In 37^{th} *Symposium on Theory of Computing (STOC)*, 2005.

[75] C. Scheideler and S. Schmid. A Distributed and Oblivious Heap. In *36th International Colloquium on Automata, Languages and Programming*, ICALP '09, 2009.

[76] M. Sherr, M. Blaze, and B. T. Loo. Veracity: Practical Secure Network Coordinates via Vote-based Agreements. In *Proc. of USENIX Annual Technical Conference*, 2009.

[77] A. Shoker. Sustainable Blockchain through Proof of Exercise. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2017.

[78] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2001.

[79] D. Stutzbach and R. Rejaie. Understanding Churn in Peer-to-peer Networks. In 6^{th} *ACM Conference on Internet Measurement (IMC)*, 2006.

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

[81] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *International conference on the theory and applications of cryptographic techniques*. Springer, 2003.

[82] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895), 2008.

[83] L. Wang and J. Kangasharju. Real-World Sybil Attacks in BitTorrent Mainline DHT. In *Proc. of IEEE Global Communications Conference (GLOBECOM)*, 2012.

[84] L. Wang and J. Kangasharju. Measuring Large-Scale Distributed Systems: Case of BitTorrent Mainline DHT. In *IEEE 13th Intl. Conference on Peer-to-Peer Computing*, 2013.

[85] W. Wei, F. Xu, C. C. Tan, and Q. Li. SybilDefender: A Defense Mechanism for Sybil Attacks in Large Social Networks. *IEEE Transactions on Parallel & Distributed Systems*, 24(12), 2013.

[86] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering Social Network Sybils in the Wild. In *Conference on Internet Measurement Conference (IMC)*, 2011.

[87] M. Young, A. Kate, I. Goldberg, and M. Karsten. Towards Practical Communication in Byzantine-Resistant DHTs. *IEEE/ACM Transactions on Networking*, 21(1), Feb. 2013.

[88] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. *Proc. of 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 36, Aug. 2006.

[89] M. Zamani, J. Saia, and J. Crandall. TorBricks: Blocking-Resistant Tor Bridge Distribution. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*. Springer, 2017.