Bankrupting Sybil Despite Churn

Diksha Gupta[1], Jared Saia[2], and Maxwell Young[3]

[1]IBM Research Singapore, 9 Changi Business Park Central 1, Singapore,486048, gupta.diksha.1991@gmail.com
[2]Department of Computer Science, University of New Mexico, Albuquerque, NM, USA, 87131, saia@cs.unm.edu
[3]Department of Computer Science and Engineering, Mississippi State University, MS, USA, 39762, myoung@cse.msstate.edu

Abstract

A Sybil attack occurs when an adversary controls multiple identifiers (IDs) in a system. 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: the 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 resource burning 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 bad IDs; and (2) when the system is under significant attack, the good IDs consume asymptotically less resources than the bad. In particular, for churn rate that can vary exponentially, the resource burning rate for good IDs under 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 show this resource burning rate is asymptotically optimal for a large class of algorithms.

We empirically evaluate ERGO alongside prior Sybil defenses. Additionally, we show that ERGO can be combined with machine learning techniques for classifying Sybil IDs, while preserving its theoretical guarantees. Based on our experiments comparing ERGO with two previous Sybil defenses, ERGO improves on the amount of resource burning relative to the adversary by up to 2 orders of magnitude without machine learning, and up to 3 orders of magnitude using machine learning.

1 Introduction

A ***Sybil attack*** occurs when an adversary controls multiple identifiers (***IDs***) in the system [3]. In practice, these attacks may be motivated by financial gain [4, 5, 6] and their mitigation has attracted significant attention from the research community [7].

A classic defense is resource burning (RB), whereby IDs are periodically required to consume local resources in a verifiable manner [8]. A well-known example of resource burning is proof-of-work (PoW) [9], but several other methods exist (see Section 2).

Unfortunately, current resource burning methods always consume resources, even when the system is not under attack. This non-stop consumption translates to substantial monetary costs [10, 11].

Prior work shows it is sometimes possible for non-adversarial (***good***) IDs to consume fewer total resources than the adversary [2]. Unfortunately, this work fails to hold when the rate at which system membership changes—often referred to as the ***churn rate***— is high. Many systems vulnerable to Sybil attacks have significant churn [12, 13, 14].

1.1 Bankrupting Sybil Despite Churn

Given this state of affairs, a natural question is: *Despite churn, can we design a Sybil defense where the cost to good IDs scales slowly with the cost to the adversary?*

A key contribution of our work is answering this question in the affirmative. Informally, we design and analyze a new defense where the rate of RB performed in total by the good IDs:

1. depends only on the churn rate of good IDs, when there is no attack;
2. grows *asymptotically* slower than that of the adversary during an attack.

In Section 3, we provide a formal statement of these guarantees, but their implications are worth highlighting here. Regarding (1), in the absence of an attack, the overhead from our defense is low. Specifically, each good ID incurs only a constant amount of RB to join the system and then a constant amount each time the system size increases or decreases by a constant factor (details are given in Section 7). Therefore, the total amount of RB performed by good IDs grows slowly, despite large changes in the system size.

Regarding (2), given that RB ultimately translates into a monetary cost, our defense places the adversary at an economic disadvantage. A Sybil attack is now provably more expensive given our defense, which is a deterrent against malicious behavior.

1.2 A Roadmap

The remainder of this manuscript is structured as follows. In Section 2, we introduce our model and formal problem. With these preliminaries in place, we give a formal statement of our main results in Section 3, along with highlighting the novelty of our approach, along with new material not present in preliminary conference versions of our work. To provide additional context for our results, we discuss prior related work in Section 6.

The specifics of our defense are given in Section 7 and 8, along with a discussion providing intuition for our design decisions. An upper-bound analysis is presented in Section 9, and followed up by experimental results in Section 10 that illuminate the performance of our defense. In Section 11, we provide a lower bound to show that our result is asymptotically optimal from a large class of natural defenses.

A discussion of how to fully decentralize our defense is given in Section 12. Finally, in Section 13, we conclude with several promising avenues for future research.

2 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 the protocol, or ***bad*** if it is controlled by the Sybil adversary (or just ***adversary***). The amount of resource burning that each good ID can perform is identical; that is, the population of good IDs is homogeneous.

Resource-Burning Challenges. IDs can construct ***resource-burning (RB) challenges*** of varying hardness, whose solutions cannot be stolen or pre-computed; some examples are discussed in Section 6. To specify hardness, a ***k-hard RB 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. The server—or the committee in our decentralized setting—learns when an ID joins or departs the system. Later, in Section 12, 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 the 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 significant message delay or clock drift, then a challenger cannot accurately measure the response time for the ID solving the challenge; see [15] for further discussion. Techniques for synchronizing on the order of milliseconds are known and suffice for our purposes [16]. We assume that no more than an ϵ-fraction of the good IDs depart in any round, for some small positive constant $\epsilon < 1/12$.

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, can read the messages sent by good IDs before sending its own. The adversary sets the timing of join and departure events by good and bad IDs, and makes these timing choices adaptively over time. Furthermore, the adversary can select which bad ID departs. However, the adversary cannot select which specific good ID departs, but rather the departing ID is selected uniformly at random (u.a.r.) from the good IDs currently in the system.

By being able to scheduling the departure times of good IDs, we capture a strong adversary. In terms of motivation, one may imagine an attacker who has been monitoring the system for a long period of time and, therefore, possesses statistics on when good IDs join and depart over different times of the day. Such an attacker might not know *exactly* when good IDs are going to join/depart, but it could predict this behavior to a degree, and our pessimistic model accounts for this. We also note that considering an adversary without this power does not appear to make it easier to obtain theoretical guarantees on security and performance.

The adversary is resource-bounded: in any single round where all IDs are solving challenges, the adversary can solve a $\boldsymbol{\kappa}$-fraction of the challenges; this assumption is common [17, 18, 19].

2.1 ABC Model of Churn

We now describe our model of churn, which we call the ***ABC model*** of churn, based on two parameters α (A) and β (B) of churn (C). This churn model is particular relevant for defending against Sybil attacks (see Section 5.1 for further discussion).

2.1.1 Joins and Departures

For simplicity, in our model, every join and departure event is assumed to occur at a unique point in time. In practice, because of the granularity of clocks, it may seem as if events are occurring at the same time, and this can be handled by having the server or committee order these events; all of our results hold even if this ordering is dictated by the adversary.

Whenever the adversary decides to cause a good ID departure event, the adversary does not get to select which good ID departs. Rather, 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. These messages are sent periodically, and their absence is interpreted as a departure by the corresponding ID. We note that a bad ID that fails to issue heartbeat messages will be treated by the server as having departed from the system. 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. We assume that every joining ID initially knows one good ID in the system. This follows if every joining ID knows a set of IDs with a good majority, since the majority of good IDs can suggest a single good one; further it is needed to avoid eclipse attacks [20, 5]. Thus it is a standard assumption in peer-to-peer networks [21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]; cryptocurrency systems like Ethereum, which uses the structured Kademlia peer-to-peer network [32, 33], and Bitcoin, which uses an ad-hoc peer-to-peer network [17, 5]; and for decentralized blockchains that use peer-to-peer overlays [34].

We define $\boldsymbol{n_0}$ to be the minimum number of good IDs in the system at any time. The ***system lifetime*** is defined over n_0^γ joins and departures, for any fixed constant $\boldsymbol{\gamma} > 0$. Our results for GoodJEst and the costs for Ergo hold over this system lifetime (see Section 3). A system designer may choose a value of γ in order to have the guarantees provided by our results.

To illustrate, imagine a system where there are always at least 6000 good IDs participating. If a system designer wishes the guarantees of Ergo to hold over 100 billion join and departure events, then $\gamma = 3$ will suffice. For dynamic systems, such guarantees are common (for examples, see [30, 29, 22]).

2.1.2 Epochs, Smoothness, and Churn

Here, we give the details for our model of churn. Later, in Section 5, we argue that our model of *good* churn is quite general, and we provide intuition for why. We also highlight that we make no assumptions about the bad churn.

Let $A \triangle B$ denote the symmetric difference between any two sets A and B, i.e. $A \triangle B = (A - B) \cup (B - A)$. Time is partitioned 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 1/2 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.

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. We now formally define the notion of "smoothness" for α and β, which allows our model to capture a churn rate that varies widely over time.

Definition 1. *For any $\alpha \geq 1$ and $\beta \geq 1$, and any epoch $i > 1$, we define the following.*

- **$\boldsymbol{\alpha}$-smoothness***:* $(1/\alpha)\rho_{i-1} \leq \rho_i \leq \alpha\, \rho_{i-1}$.
- **$\boldsymbol{\beta}$-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 $\lceil \beta\ell\rho_i \rceil$.*

Our guarantees require that $n_0 \geq \max\{6000, (720(\gamma+1))^{4/3}, (41\beta)^2\}$. This lower bound on n_0 arises in our analysis and we discuss this in Section 9.3.

Varying Churn Rate. In Section 5, we motivate our new churn model, and we provide intuition with examples aided by illustrations. Here, we succinctly highlight the roles of α and β.

The parameter α can capture *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, say $\alpha = 2$. In this case, the good join rate may increase *exponentially* from epoch to epoch. Similarly, the good join rate may also decrease exponentially. The parameter β ensures there can be possibly large deviations within an epoch from the average good join rate over the entire epoch.

Many other churn models have been described in the research literature. In Section 4, we compare our model of churn against several popular prior models.

2.2 The DefID Problem

In the well-studied GenID problem [35, 36, 18, 37, 38], there is some initial set of good and bad IDs in a permissionless system. The problem GenID requires the IDs to all *generate* an initial set of IDs with a bounded fraction of bad. In particular, all good IDs must decide on a set of S such that: all good IDs are in S; and a $O(\kappa)$-fraction of the IDs in S are bad.

We introduce the **DefID *(DEFend ID)*** problem, which generalizes GenID to handle churn. Specifically, ID join and departure events follow our ABC model of churn. Our goal is to ensure that, at any time t, all good IDs know a set $S(t)$ such that (1) all good IDs are in $S(t)$; and (2) a $O(\kappa)$-fraction of IDs in $S(t)$ are bad.

2.2.1 Why is DefID harder than GenID?

DefID is strictly harder than GenID. Now, 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 $O(\kappa)$ bound on bad IDs may be violated after some amount of churn. A naive approach to solve DefID would be to run a solution to GenID after every join and departure event. But this is expensive. Specifically, it requires a resource-burning cost that is linear in the current system size for *every* ID join and departure event, using the best known bound for GenID [38]. Thus, it is necessary to define this new DefID problem in order to be able to design an efficient Sybil defense in the presence of churn.

3 Our Results

Recall that κ is the fraction of the resource that is controlled by the adversary. Let the ***good spend rate*** be the total resource burning cost for all good IDs per second. Similarly, let $\boldsymbol{T}$ be the ***adversary's spend rate*** and let $\boldsymbol{J}$ be the ***join rate of good IDs*** over the system lifetime.

The first theorem solves the DefID problem using our algorithm **Ergo**, whose specification is given in Section 7.

Theorem 1. *For $\kappa \leq 1/18$,* Ergo *ensures that the fraction of bad IDs in the system is always less than $3\kappa \leq 1/6$. Additionally, with probability of error that is $o(1/n_0)$ over the*

system lifetime, the good spend rate is

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

A strict upper bound of 1/6 enables solutions to problems such as Byzantine agreement and secure multiparty computation [39, 40].

To complement our upper bound, we provide a lower bound in Theorem 11.1 that shows that this result is asymptotically optimal for a large class of algorithms.

To complement the upper bound in Theorem 1, we show in Section 11 that when α and β are $\Theta(1)$, ERGO's resource burning rate is asymptotically optimal over a large class of algorithms. We also show in Section 12 how to remove reliance on a single server so that ERGO can be executed in a decentralized fashion, while providing similar guarantees (see Theorem 4).

ERGO makes use of a second algorithm, **GOODJEST**, that may be of independent interest. GOODJEST correctly estimates the good join rate, provided that the fraction of bad IDs is always less than 1/6. **Since ERGO always ensures that the fraction of bad IDs is less than 1/6**, it is able to make use of GOODJEST. As addressed in Sections 7.1 and 9.2, ERGO uses GOODJEST to achieve the good spend rate of Theorem 1. The main properties of GOODJEST are as follows.

Theorem 2. *Assume the fraction of bad IDs is always less than* 1/6*. Then with probability of error that is* $o(1/n_0)$ *over the system lifetime the following holds. 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:*

$$1/(88\alpha^4\beta^3)\rho \leq \tilde{\mathtt{J}} \leq 1867\alpha^4\beta^5\rho.$$

This theorem 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 10.2).

We validate our theoretical results by comparing ERGO against prior RB-based defenses using real-world data from several networks (Section 10.1). In terms of the amount of RB performed relative to the adversary, 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 has ERGO leveraging a prior machine-learning method [41], and we observe an improvement by up to 3 orders of magnitude in comparison to previous algorithms for large-scale attacks.

3.1 A Note on Incentives

In this paper, we do not explore the challenging problem of how to *incentivize* good IDs to solve challenges. This is an issue that concerns resource-burning techniques in general (see Section 6 for other examples of resource-burning), and there are a variety of ways in which

good IDs might be motivated to solve puzzles. Even though it is beyond the scope of this paper, we do *sketch* some ideas of how incentivization could happen, based on techniques used in cryptocurrency systems. For clarity of presentation, we defer this discussion to Section 13.

3.2 New Contributions

This manuscript contains results previously published in the proceedings of the *41st IEEE International Conference on Distributed Computing Systems* (ICDCS'21) [1] and in the proceedings of the *33rd IEEE International Parallel and Distributed Processing Symposium* (IPDPS'19) [2]. In particular, the majority of our results presented here appeared recently at ICDCS'21, while our lower-bound result appeared at IPDPS'19.

This manuscript has been substantially revised and expanded. Specifically, we highlight the following new material:

- Full proofs of all our results, including proofs of several lemmas that were not included in the conference proceedings due to space constraints (Section 9).
- Exposition giving intuition for the guarantees provided by GoodJEst (Section 8.1 and Figure 6).
- A complete presentation of our approach for decentralizing Ergo. We have included full proofs and added additional exposition (Section 12).
- Additional empirical results for our main algorithm, Ergo, that defends against the Sybil attack. We design and evaluate several heuristics aimed at further lowering the cost of Ergo, while still retaining its guarantees (Section 10.3 and Figure 10).
- Additional empirical results for the algorithm, GoodJEst. We examine the performance of GoodJEst over new data sets for the Ethereum and Gnutella systems (Section 10.2 and Figure 9).
- Additional discussion of several pertinent issues related to our results: incentives for good IDs to solve puzzles (Section 3.1), additional discussion of classification approaches for defending against the Sybil attack (Section 6), and future directions of research (Section 13).

3.3 Technical Contributions and Novelty of Approach

The technical contributions of our results include:

- **Formal Problem Definition.** To the best of our knowledge, this paper is the first to formally define the Sybil defense problem with churn. Our model of churn (Section 2.1.2) is quite general, simple to describe and mathematically tractable. Our DefID problem (Section 2.2) formally defines what is needed to prevent Sybil attacks in the presence of churn.

- **Spending Less than the Attacker.** Before the results in this paper, it was unknown whether a Sybil defense algorithm could spend asymptotically less than an attacker. In this paper, we describe and analyze the first Sybil defense algorithm that achieves this result. The fact that such a result is even possible is surprising, and perhaps a useful contribution to the general science of distributed security.

Achieving these results requires solving many technical issues. Below, we summarize some of the novel techniques we have developed in order to address these problems.

We note in the following that, initially, we assume a single server maintains the system membership information (recall Section 2); in Section 12 we describe how a committee can maintain system membership information.

- **Estimate and Set.** We introduce a new, general approach for the design of algorithms to secure a system in the presence of churn. Our technique consists of two parts. First, *estimating good activity*: here, estimating the rate at which good IDs join the system. Second, *setting costs using this estimate*: here, costs to join the system are set carefully based on this estimate.

- **Estimating the Good Join Rate.** A major technical challenge is estimating the good join rate (Sections 8 and 9.1). How can we achieve this estimate when we do not know, a priori, whether joining IDs are good or bad? Our solution is to estimate the "flow" of new, good IDs into the network during a certain time period, by bounding the flow of bad IDs in; and the flow of new, good IDs out. Technical tools we use to achieve these bounds are: (1) defining the time period for measurement using symmetric difference of membership sets, in order to achieve both upper and lower bounds on the flows; and (2) deriving careful stochastic, concentration results to bound the flow of good IDs out, with high probability. Section 8.1 gives more intuition on our approach, which we believe again may be of independent interest.

- **Setting Costs based on Good Join Rate.** We use our estimate of the good join rate to set the costs to enter the system. Whenever the system membership changes by a constant fraction, we also charge all IDs a cost of 1, in order to purge the system of a possible excess of bad IDs. Setting the entrance cost too high means that good IDs pay too much when there is no attack. Setting it too low means that bad IDs can cheaply join and increase costs in the future due to more frequent purges. We can resolve this tension mathematically by setting the entrance costs to the ratio of the join rate of all IDs over the (estimated) join rate of good IDs. Section 7.1 gives more intuition about our approach for setting costs, and Section 9.2 gives our formal analysis, which makes use of bounds on our estimate of the good join rate, and also makes critical use of the Cauchy-Schwartz inequality.

4 Churn Models

As we highlighted in above in Section 3.3, our churn model is an important contribution. Here, we summarize other popular churn models in the literature (Sections 4.1 and 4.2), and

then we motivate and discuss our new model of churn (Section 5).

4.1 Dynamic Network Model with Churn (DNC)

The *Dynamic Network Model with Churn* (DNC) model was first proposed by Augustine, Pandurangan, Robinson, and Upfal in 2012 [42]. Then, subsequent papers described algorithms in this model to solve Byzantine agreement [42, 43]; leader election [44]; maintain a Distributed Hash Table (DHT) [45]; maintain a bounded-degree expander topology [46] and solve storage and retrieval problems [47]. The survey paper [48] gives an overview of the DNC model and results.

For completeness, we now describe aspects of the DNC model that are relevant to this paper. DNC assumes that the network size is fixed. An adversary is assumed to control what nodes join and leave and at what time, and the adversary has complete knowledge of the algorithm and unlimited computational power. The maximum node churn rate is parameterized: in any round, up to $C(n)$ nodes can be replaced by new nodes. Typically, algorithmic results in this model can handle $C(n) = \Theta(n)$ for an adversary that is oblivious: unaware of the past random choices of the algorithm; and $C(n) = \Theta(\sqrt{n})$ for an adversary that knows the random choices of the algorithm. Finally, we note that the DNC model also considers edge churn in the network topology; we omit discussion of this aspect of the model since it is not relevant to this paper.

Byzantine IDs occur in the DNC model; typically with the assumption that for some positive ϵ, there are $O(n^{1/2-\epsilon})$ Byzantine IDs in every round. The adversary controlling the Byzantine IDs also controls churn. This is similar to the adversary in our own ABC model.

Results in the DNC generally hold for all but a $1/\text{polylog}(n)$ fraction of the good IDs. For example, in the leader election results, all but a $1/\text{polylog}(n)$ fraction of the IDs agree on the correct leader [44]; in Byzantine agreement [44, 42, 43] all but a $1/\text{polylog}(n)$ fraction of the IDs agree on a correct output bit. These types of results are necessary given that the DNC model (1) assumes connectivity in sparse networks with edge churn; and (2) allows targeted deletions of good IDs by the adversary.

4.1.1 Differences between the DNC and ABC model

We now discuss differences between the DNC model and the ABC model. First, while the ABC model allows the adversary to schedule when good ID deletions will occur, it does not allow the adversary to target specific good ID. In particular, in the ABC model, when a good ID deletion event occurs, the good ID that is deleted is selected uniformly at random from the set of all good IDs.

Byzantine IDs are another key difference. In the DNC model, in every round, by assumption, for some positive ϵ, there are at most $O(n^{1/2-\epsilon})$ Byzantine IDs in the system. In contrast, in the ABC model, there is no such assumption. Instead, the ABC model assumes that the adversary controls a constant fraction of the RB resource, and must use this fact to try to constrain the fraction of Byzantine IDs.

4.2 Churn Models without Byzantine IDs

Several other models have been proposed for churn which, unlike the DNC and ABC models, do not consider Byzantine IDs. We now discuss these other results.

First, work by Ko, Hoque and Indranil, from 2008, describes two churn models, *TRAIN* and *CROWD* [49]. In both models, the system size is fixed, and each ID join event occurs in parallel with an ID deletion event. Additionally, in both models, the churn rate, or number of processes joining per unit time, is assumed to be fixed over the entire lifetime of the system. In the TRAIN model, there is some fixed, positive integer K and the join and leave events can only occur at times that are integer multiples of K. In the more challenging CROWD model, join and leave events can happen at any time. In contrast, the ABC model allows for system size increase and decrease over time, and these changes are not restricted to specific multiples of events.

We note that the DNC model (recall Section 4.1) is more general than both the TRAIN and CROWD models since it allows up to a certain number of join and deletion events to occur in a single round, but does not require that exactly a K number of events occur. Thus, it does not require that the churn rate stay fixed throughout the lifetime of the system, only that there is some upper bound on the churn rate.

Second, in 2004, Aguilera proposed several churn models allowing for infinitely many IDs [50]. Four key models are proposed in this paper, based on: whether or not (1) the system has a finite number of IDs; and (2) whether or not each run has a finite number of IDs. For example, if the system has infinite IDs, then for every integer N, there are runs with more than N IDs seen throughout the run. IDs are assumed to communicate via shared memory, and the paper describes how to use shared memory to implement counters, atomic snapshots, group membership, and mutual exclusion in several of their proposed churn models. These aspects of the model differ substantially from the ABC model because of the problems for which they are designed. In particular, join and departure event timing is not parameterized as it is in the ABC model, and so resource costs for their results are not given as a function of model parameters such as α and β.

Third, in 2002, Liben-Nowell, Balakrishnan and Karger [51] defined the notion of *half-life* as follows. Consider a system with N IDs at time t. The time elapsed until another N IDs join is the doubling time from time t. The time until $N/2$ IDs that are present in the system at time t depart is the halving time from time t. The minimum of the doubling time and the halving time is the half-life from time t, and the *half-life of the system* is the minimum half-life over all t.

The half-life as defined in this way is closely related to our notion of an epoch, but there are technical differences, which we now describe. There is always at least one epoch in every half-life. To see this, first note that after $N/2$ additions or $N/2$ deletions, the symmetric difference has changed by at least $N/2$, which satisfies the criteria to end an epoch. However, there may be multiple epochs in one half-life, since IDs added over one epoch may be deleted in subsequent epochs, thereby avoiding the criteria needed for the end of a half-life.

Fourth, the distribution of ID ***session times*** can be used to characterize churn; that is, the times for which IDs remain in the system. Smaller session times are indicative of higher churn, and vice-versa. Real-world measurements can inform the distribution of

session times, although these findings are specific to the system in question. For example, a measurement study of the peer-to-peer (P2P) system Gnutella found that session time was distributed exponentially with a mean exceeding two hours, whereas the distribution for the P2P system Kazaa is heavy-tailed, with an average session time of roughly a few minutes [52]. Another example involves the peer-to-peer system KAD [12] and the Bitcoin network [53], where session times for both are well-fit by a Weibull distribution; however, the parameters of these distributions are very different. Therefore, even within the P2P domain, the session-time distribution—and, thus, the characterization of churn—can differ significantly between specific systems. By comparison, the ABC model does not incorporate a particular session-time distribution; rather, churn is defined more generally via the two parameters α and β.

5 Motivating α, β-Churn

Here, we argue that the Sybil attack necessitates a new model of churn. Then, we motivate our notion of α, β-churn, illustrating how it captures the key aspects of a Sybil attack, while remaining mathematical tractable.

5.1 Sybil Attack Demands New Churn Models

During a Sybil attack, the system size may increase as the adversary injects large numbers of IDs. Additionally, a Sybil adversary can also decrease the system size by removing IDs it controls. In all cases, an algorithm that depends on static system sizes will be fragile when faced with a Sybil adversary. Thus, models that assume a static system size—such as DNC, TRAIN, and CROWD—seem inappropriate for addressing the Sybil attack.

5.2 Some Gentle Intuition for the α, β-Churn (ABC) Model

If the system size is not fixed, how should we model churn? First, it seems clear that there should be no constraints on the timing of events for the bad IDs, since these are controlled by an adversary.

So, how should we constrain the timing of *good* ID join and departure events? A simple idea is to assume a fixed rate for good events, and to assume that this rate never changes. This is similar to some models in [49]. However, this is unrealistic in many settings. For example, a system might experience an unusually high good join rate during certain times of day or the week, or during unpredictable events, such as a sudden spike in popularity in some system service. Additionally, there may be periods of time during which systems size grows or shrinks *non-linearly*. These phenomena cannot be captured by a fixed event rate.

A more sophisticated approach is to allow for some change in the good event rate. For example, the good join rate could itself change according to some separate rate. But what should be this new change rate be?

One idea is that the rate of *change* could depend on the good event rate. For example, if the good event rate is initially 1 ID per millisecond, then the change rate could be by a

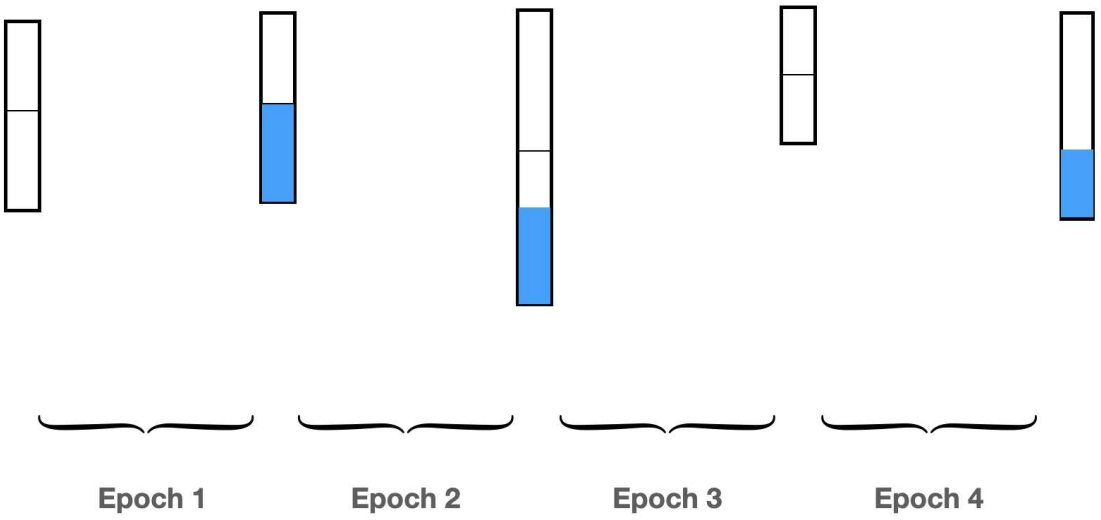

Figure 1: Illustration of four epochs. Time proceeds from left to right, and each rectangle illustrates the state of the system at the beginning of an epoch, with half of the rectangle delineated by the thin horizontal line. The blue rectangles indicate the new IDs that have been added during the epoch; see the discussion at the end of Section 5.3.

factor of 2 every *millisecond*. But this seems too fast - it allows the rate to double with each new event.

An alternative would be to have the change rate set to a factor of 2, say every $1,000$ events ($1,000$ milliseconds). But setting the denominator to some absolute number of events is artificial: in systems with large populations, we might not expect much change over $1,000$ events, but in systems with small populations, we might expect significant change over $1,000$ events.

5.3 The epoch

To resolve this issue, the fundamental time period we use for defining rate of change is the *epoch*. As defined previously, an epoch is the amount of time over which the symmetric difference of the system population changes by half the initial size of the system (see Section 2.1.2). Recall that this definition is closely related to the "half-life" as defined in [51].

By letting the event rate change by a fixed amount in each *epoch*, we obtain a churn model that generalizes to both small and large systems.

We now provide some intuition about the notion of an epoch. Figure 1 illustrates four example epochs. In this figure, time moves from left to right, and each rectangle illustrates the state of the system at the beginning of an epoch, with half of the rectangle delineated by the thin horizontal line.

The first epoch ends with half of the IDs changed; these IDs are illustrated with the blue bar. Notice that the system size has not changed at the end of this first epoch. The second epoch ends when new IDs total half of the system size at the start of the epoch. Notice

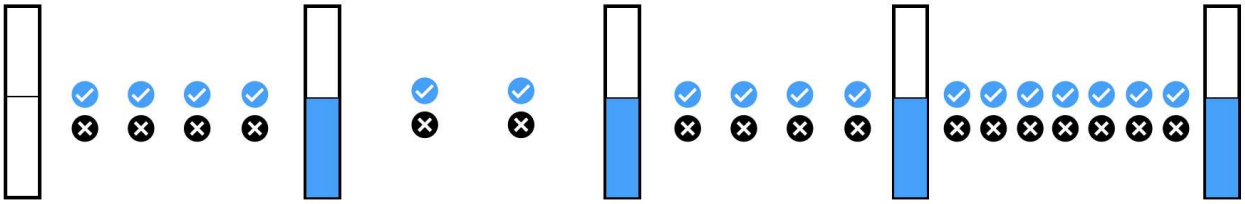

Figure 2: Example for $\alpha = 2$, $\beta = 1$. Blue checks represent good ID join events; black x's represent good ID deletion events. In this example, for simplicity, the system size does not change, so the number of good join events equals the number of good deletion events. Since $\alpha = 2$, the number of events can change by a factor of 2 from epoch to epoch. Since $\beta = 1$, all these events are uniformly distributed in time.

that the system size has grown by 50%. The third epoch ends when departing IDs total half of the system size at the start of the epoch. Notice that the system size has decreased by half. The fourth epoch ends when new IDs total half the size at the epoch start. Notice throughout that the system size is multiplied by some value in the range $[.5, 1.5]$ during the course of any epoch.

5.4 α-smoothness

We now describe how the value of α can allow for the good event rate to change across epochs. Recall from Section 2.1.2 that in each epoch i, there is a good event rate, ρ_i. Additionally, recall the α-smoothness criteria:

- ***α-smoothness***: $(1/\alpha)\rho_{i-1} \leq \rho_i \leq \alpha\,\rho_{i-1}$.

This ensures that the good event rate changes by no more than an α-factor from epoch to epoch. For example, when $\alpha = 2$, if the good event rate is 1 event per millisecond in epoch $i - 1$, then in epoch i, the good event rate will be in the range from 1 event per two milliseconds up to 2 events per millisecond.

Figure 2 illustrates an example for $\alpha = 2$. In this figure, the checks are good ID additions and the x's are good ID deletions. Again, the system state at the beginning of epochs is represented by a rectangle; the blue half of the rectangle illustrates the half of the system IDs that are new in comparison to the system population at the start of the epoch. For simplicity, this figure illustrates a case where the system size does not change.

All epoch lengths are the same: 1 hour. Finally, in order to focus solely on α, we have set $\beta = 1$. This means that all events are spread out evenly over the duration of the epoch. To keep things simple, in the figure, the epochs all have the same length. This can happen even when the value of ρ changes, because deletion events can either be for IDs that have been added during the epoch, or from IDs that were around at the start of the epoch. In the former case, the ID that joined and then departed does not hasten the end of the epoch.

Since $\alpha = 2$, it is possible for the ρ_i values vary by multiplicative factors of 2 from one epoch to another. In the figure, $\rho_1 = 4$, $\rho_2 = 2$, $\rho_3 = 4$ and $\rho_4 = 7$.

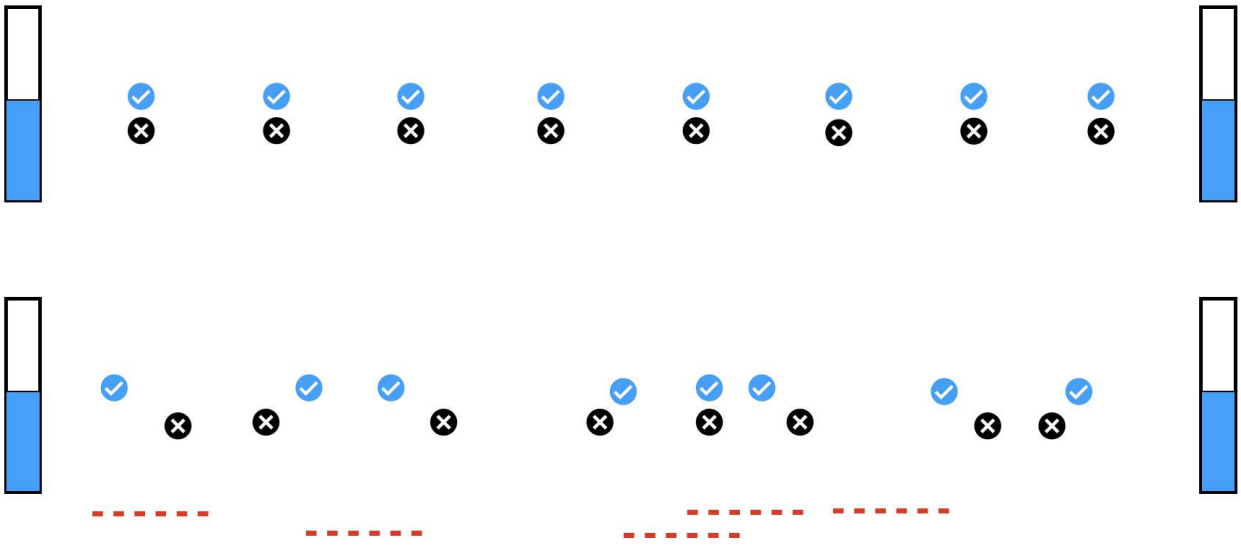

Figure 3: Single Epoch with Different β values. The dashed red lines at the bottom represent "durations" of time equal to the epoch length divided by the number of good join events. Top: $\beta = 1$; each duration overlaps exactly 1 join event and at most 1 deletion event; events are spread evenly. Bottom: $\beta = 2$; each duration overlaps between $\lfloor 1/2 \rfloor = 0$ and $\lceil 2 \rceil = 2$ join events, and at most 2 deletion events; events may clump.

A key observation is that, even for a small value such as $\alpha = 2$, the good join rate can increase (or decrease), *exponentially* over multiple epochs. In particular, over x epochs the event rate can decrease by a factor of 2^{-x} or increase by 2^x.

5.5 β-smoothness

In the previous section, good events were evenly spread over time in each epoch. Next, we show how this can change by discussing the final part of our model: β-smoothness. Recall the definition from Section 2.1.2:

- ***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 $\lceil \beta\ell\rho_i \rceil$.

Figure 3 illustrates how different values of β effect the spacing of good events over a single epoch lasting 1 hour. Both the top and bottom sub-figures illustrate epochs with the same ρ value: $\rho = 8$ events per hour. In the top figure, $\beta = 1$ and in the bottom figure, $\beta = 2$.

The red dashed lines at the bottom of the figure are a few example "durations" of length $\ell = 3600/8 = 450$ seconds. In the top epoch, β-smoothness for $\beta = 1$ requires that within each such duration, there must be at least $\lfloor \ell\rho \rfloor = 1$, and at most $\lceil \ell\rho \rceil = 1$ good join event during this red duration. Additionally, there must also be at most $\lceil \ell\rho \rceil = 1$ good deletion. Note that these bounds hold for each duration in the top subfigure.

In the bottom epoch, $\beta = 2$, so β-smoothness requires that within each duration, there must be at least $\lfloor (1/2)\ell\rho \rfloor = 0$, and at most $\lceil 2\ell\rho \rceil = 2$ ID join events during this red duration. Additionally, there must also be at most $\lceil \ell\rho \rceil = 2$ good deletions. Note that these bounds hold for each duration in the bottom subfigure.

In sum, larger values of β, such as in the bottom epoch, allow for more "clumping" of the good events. Smaller values of β require the events to be more evenly spread. Finally, we note that for simplicity, our duration lengths in Figure 3—the red dashed lines—are all the same length. However, the β-smoothness criteria holds for any duration length that is completely contained in the epoch.

6 Related Work

A preliminary version of our results appeared in [1, 2]. Specifically, the problem definition for DefID, model of churn, upper-bound analysis, and experimental results appeared recently in [1], while our lower-bound result (Section 11) appeared in [2]. Our presentation here includes new material as detailed previously in Section 3.2.

Sybil Attacks. There is significant prior work on Sybil attacks [3]. For example, see surveys [7, 54], [55, 7, 54] and additional work documenting real-world Sybil attacks [6, 56, 57].

Several results leverage social networks for Sybil defense [58, 59, 60], including recent work using machine learning to *classify* likely Sybil IDs, such as SybilExpose [61] and **SybilFuse** [41]. Since social-network data may not always be available, in this paper, we focus on Sybil defense without it. We also note that, by themselves, classification methods do not solve DefID. In particular, a classifier that is wrong with even a small probability, say 10^{-6}, still allows the adversary to obtain a bad majority, over a large number of attempted join events.

That said, in Section 10, we do show that Ergo combines well with classification algorithms like SybilFuse. In particular, SybilFuse significantly reduces costs for Ergo, when social network data is available to use SybilFuse (Section 10, Heuristic 4). Additionally, Ergo enables a classification algorithm like SybilFuse to be leveraged to create a full-fledged Sybil defense algorithm that can withstand significant, long-term attack.

Other Sybil defenses use network measurements [62, 63, 64]. These defenses rely on accurate measurements of latency, signal strength, or round-trip times to try to detect Sybil IDs. Again, since such data may not always be available, Ergo does not rely on its use. But we expect that, when this type of data is available, these results could also be used to reduce costs for Ergo.

Finally, Danezis et al. [65] and Scheideler and Schmid [66] describe containment strategies for Sybil attacks in overlay networks. However, these results focus on isolating older IDs from newer bad IDs during a Sybil attack, and so do not ensure that the fraction of bad IDs in the network is always bounded.

Resource Burning. Many resource burning schemes for Sybil defense exist. *Computational puzzles* consume CPU cycles [17, 67, 18]. *Proof of Space-Time*, requires allocation of storage

capacity [68]. *Proof of useful-work* consumes CPU cycles to solve challenges applicable to real-world scientific or engineering problems [69, 70].

A *completely automated public Turing test to tell computers and humans apart (CAPTCHA)* is a resource - burning tool where the resource is human effort [71, 72]. CAPTCHAs of tunable hardness have been proposed [73], as have CAPTCHAs that channel human effort into practical problems such as deciphering scanned words or detecting spam [74].

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 [75, 76, 77]; the resource here is listening capacity.

Finally, we note that Proof of Stake [19, 78, 79] 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 [80].

Guaranteed Spend Rate. In [81] and [2], 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 holds for 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.

We also note that, outside of the Sybil attack, several prior works address network security challenges with results that are parameterized by the adversary's cost [82, 83, 84, 85, 86, 87, 88, 89, 90]. Such results are referred to as *resource-competitive*, and many examples are provided in the survey by Bender et al. [91].

7 ERGO

We begin by walking through the execution of Ergo and providing intuition for its design. Recall from Section 2 that we are presenting Ergo with coordination provided by a server. Therefore, the server executes the pseudocode for Ergo presented in Figure 4. 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***, where an iteration consists of Steps 1 and 2 in the pseudocode.

In Step 1, each ID that wishes to join the system must solve an RB challenge of hardness 1 plus the number of IDs that joined within the last $1/\tilde{\mathsf{J}}$ seconds, where $\tilde{\mathsf{J}}$ 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 7.1. Once the ID returns a valid solution, the server adds the ID to a membership set that it maintains and the ID is considered to have successfully joined the system.

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; we note that the value 1/11 is not special, as discussed later in Section 9.3. 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. To do this,

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:

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 as follows:
   (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 4: Pseudocode for ERGO.

the server issues a 1-hard RB challenge to all IDs and then removes from the membership set those IDs that fail to respond with a valid solution within 1 round. Note, again, that the server maintains a membership set of all IDs in the system.

7.1 Intuition for Entrance Cost

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, consider the case where the ratio of bad ID joins to good ID joins is x and these bad ID join events are evenly spread out over time. Thus, the number of bad IDs joining in any $1/\tilde{\mathtt{J}}$ seconds is about x. For each good join event, the adversary pays an entrance cost which is at least $1 + 2 + 3 + ... + x = \Theta(x^2)$. In contrast, the good ID that joins in this time pays at most $x + 1 = O(x)$; recall that, in the worst case, this good ID joins after the bad IDs. Therefore, over this interval of $1/\tilde{\mathtt{J}}$ seconds, the good ID pays an entrance cost that is asymptotically the square root of what the adversary pays.

A challenge we face in analyzing ERGO is establishing this flavor of result more generally. Nonetheless, we can extend this reasoning a bit further to demonstrate more intuition for why Theorem 1 is plausible. During a large attack, the adversary's spend rate is $T = \Theta(\xi \mathtt{J}_\mathtt{a})$, where $\boldsymbol{\xi}$ 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:

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

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

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 do not 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 9.1.

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. This implies that the entrance costs—which make use of the good join rate—yield an advantage over the adversary, as sketched above. Finally, we make use of Cauchy-Schwartz to upper-bound ERGO's total cost based on the bounds for each iteration, thus completing the upper-bound analysis.

7.2 Intuition for Purging

The purpose of purging is to ensure that the fraction of bad IDs in the system is less than 1/6 at all times. When a 1-hard RB challenge is issued to all IDs, the adversary can only solve a k-fraction within a round and thus keeps at most a κ-fraction of its bad IDs in the system. Our main result for ERGO result holds for $\kappa \leq 1/18$, and so immediately after each purge the fraction of bad IDs in the system is at most 1/18. During the iteration, the fraction of bad IDs can increase, but the iteration always ends before this fraction can be reach 1/6. This reasoning is formalized in Lemma 9.

8 GOODJEST

GOODJEST provides an estimate, $\tilde{\mathtt{J}}$, of the good join rate, when there is at most a constant fraction of bad IDs. We require that the fraction of bad IDs is less than 1/6.

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 12, 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 $\tilde{\mathtt{J}}$ was updated, and t' will be the current time.

The pseudocode is presented in Figure 5. There are two aspects that must be addressed in designing GOODJEST. First, at what points in time should $\tilde{\mathtt{J}}$ be updated? This occurs

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 $|S(t') \triangle S(t)| \geq \frac{5}{12}|S(t')|$, do:

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

Figure 5: Pseudocode for GOODJEST.

whenever the system membership has changed by a constant factor with respect to the current system size. In particular, $\tilde{\mathtt{J}}$ is updated when $|S(t') \triangle S(t)| \geq \frac{5}{12}|S(t')|$ holds true; we note that the value 5/12 is not special, as discussed later in Section 9.3. Since join and departure events are ordered, this is equivalent to the property that $|S(t') \triangle S(t)| = \lceil \frac{5}{12}|S(t')| \rceil$. We refer to $(t, t']$ as an ***interval***. The execution of GOODJEST divides time into consecutive, disjoint intervals.

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

8.1 Intuition for GOODJEST

We provide intuition, using Figure 6, for how GOODJEST estimates the rate at which good IDs join the system, despite the fact that we do not know a priori which IDs are good and which are bad. The full, formal analysis is in Section 9.1.

Fix some interval, and let $S(t)$ and $S(t')$ be the set of good IDs at the beginning and end of the interval, respectively; $s = |S(t)|$; and a be the number of good IDs that join during the interval. We provide intuition for why a is always $\Theta(s)$, and thus, why dividing s by the interval length yields an estimate of the good join rate. For simplicity, we only consider here the case where the system size is fixed, but the IDs change; our intuitive reasoning only provides results in expectation. Our full proof handles changing system size and gives results with high probability (see Section 9.1). Symmetric difference is critical; for example, if we delineated intervals simply by the raw number of joins and deletions, this would allow us to obtain that $a = O(s)$, but not that $a = \Omega(s)$. We now sketch our result.

Figure 6 (left) shows why the number of good IDs that join is greater than a constant times the system size at the start of the interval, i.e. why $a = \Omega(s)$. In the figure, the old IDs are grey and new IDs are blue, where an ID is called *old* if it was in the system at the beginning of the interval, and is *new* otherwise. The good IDs are the stick figures and the bad IDs are the horned faces. The interval only ends when a 5/24 fraction of new IDs have joined. This is true because, since the system size stays constant, the number of departing IDs equals the number of joining IDs, thus, 5/24 joining and 5/24 departing IDs yields the total 5/12-fraction change required to ends an interval. Note that the fraction of new, good IDs must be at least $5/24 - 1/6 = 1/24$ since the fraction of new, bad IDs is at most 1/6.

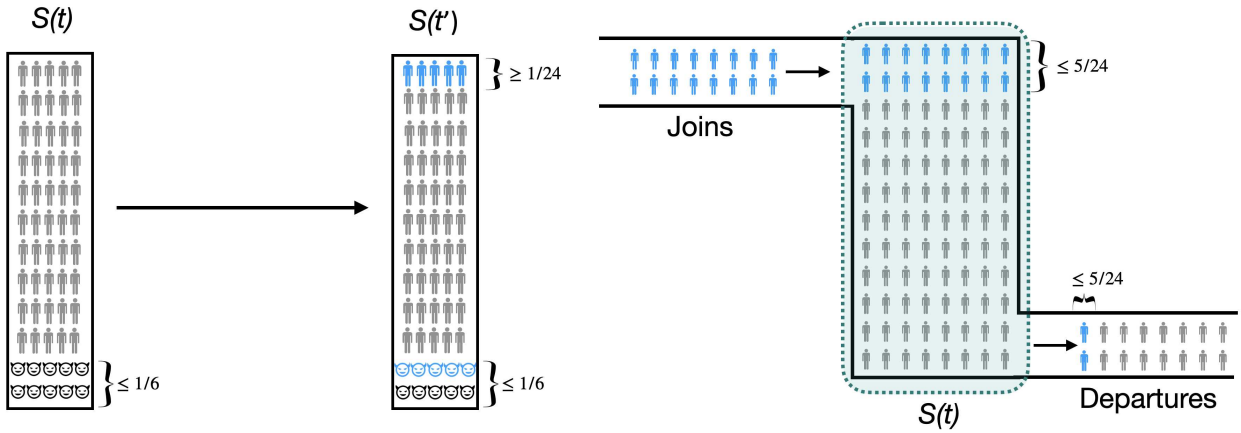

Figure 6: Example used to give intuition for GoodJEst, as discussed in Section 8.1. Blue and gray indicate a new and old ID, respectively. Stick figures and horned-faces indicate good and bad IDs, respectively.

This shows that $a \geq (1/24)s$.

In Figure 6 (left), the system size is 60. The number of bad IDs (old and new) is 10, and the fraction is thus exactly 1/6. The total number of good new IDs is 5, giving a fraction that is $5/60 \geq 1/24$.

Figure 6 (right) shows why the number of good IDs that join is smaller than a constant times s, i.e. why $a = O(s)$. This fact does not follow trivially: if each new good ID immediately departs, this would increase a, but would not alter the symmetric difference. For simplicity, our figure focuses only on the good IDs, in order to more simply illustrate the interplay between arrivals and departures.

The figure illustrates the following. **Fact 1:** the fraction of new good IDs in the system is always at most 5/24 during the interval, otherwise the interval would end. To see this, first note that GoodJEst ends an interval when the symmetric difference is more than a 5/12 fraction of the system size at the beginning of the interval. Then, note that if the fraction of new good IDs is ever more than a 5/24 fraction of the system size, the symmetric difference will exceed a 5/12 fraction of the beginning system size.

Next, we have the following. **Fact 2:** the fraction of new IDs in the set of departing good IDs is at most 5/24 in expectation. To see this, first recall that when a good departure occurs, the departing ID is selected uniformly at random from the set of good IDs currently in the system (See Section 2). Thus, a departing good ID is new with probability at most equal to the fraction of new, good IDs in the system.

From these two facts, we note the following. In expectation, the fraction of good IDs that join and are deleted is at least $a - (5/24)a$, by Fact 2. Second, the fraction of good new IDs in the system at any point is always no more than $(5/24)s$, by Fact 1. Putting these together, we get that in expectation, $a - (5/24)a \leq (5/24)s$; solving for a, we get that $a \leq (5/19)s$.

The above intuition is formalized in Lemmas 3 and 4 of Section 9.1. The analysis in

Notation	Definition
$A \triangle B$	The symmetric difference between sets A and B.
ρ_i	The good join rate in epoch i.
α	The good join rate between two consecutive epochs differs by at most an α-factor.
β	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.
κ	In any single round where all IDs are solving challenges, the adversary can solve a κ-fraction of the challenges.
ϵ	In any single round, at most an ϵ-fraction of good IDs may depart, for $\epsilon < 1/12$.
n_0	The minimum number of good IDs in the system at any time.
γ	The system lifetime is defined over n_0^γ joins and departures, for any fixed constant $\gamma > 0$.
T	The adversary's spend rate over the system lifetime.
J	The join rate of good IDs over the system lifetime.
J^B	The join rate of bad IDs over the system lifetime (specific to the lower bound argument in Section 11).

Table 1: Table of commonly used notation.

Section 9.1 solves the following remaining problems: providing sharp concentration bounds; handling changing system sizes; and using the value of a to obtain the good join rate, even when an interval intersects multiple epochs.

9 Analysis

In this section, we provide full proofs of our results. We start with the analysis of GoodJEst (Section 9.1). Next, we prove the correctness properties for Ergo, and prove the spending rate upper-bounds for Ergo, when using the estimate provided by GoodJEst (Section 9.2). For ease of reference, we collect our commonly used notation in Table 1. Finally, we conclude this section with a discussion of our choice of values for several parameters and constants used in our analysis and algorithm design (Section 9.3).

9.1 Analysis 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.

Lemma 1. *An interval intersects at most two epochs.*

Proof. Assume that some interval starts at time t_0 and intersects at least three epochs; we will derive a contradiction. Given this assumption, there must be at least one epoch entirely contained within the interval. Consider the earliest such epoch, and let it start at time $t_1 \geq t_0$ and end at time $t_2 > t_1$. Observe that:

$$\begin{aligned} |S(t_2) \triangle S(t_0)| &\geq |G(t_2) \triangle G(t_1)| \\ &\geq \left(\frac{1}{2}\right) |G(t_2)| \\ &\geq \left(\frac{1}{2}\right)\left(\frac{5}{6}\right) |S(t_2)| \\ &= \left(\frac{5}{12}\right) |S(t_2)| \end{aligned}$$

In the above, step one holds since $|S(t_2) \triangle S(t_0)| \geq |S(t_2) \triangle S(t_1)|$; step two holds by the definition of an epoch; and the second to last step holds given that the fraction of bad IDs is always less than 1/6.

But the above inequalities show that $|S(t_2) \triangle S(t_0)| \geq \frac{5}{12}|S(t_2)|$. Therefore, the interval ends by time t_2, and there can be no third epoch intersecting the interval; this contradiction completes the argument. □

At this point, it is useful to foreshadow the relationship between epochs (recall Section 2.1.2), intervals (used by GoodJEst), and iterations (used in Ergo). Lemma 1 establishes the "translation" between the first two. Later, in Section 9.2 (Lemma 11), we prove the translation between intervals and and iterations. Figure 7 depicts the relationship between epochs, intervals, and iterations.

Why is this translation necessary? In this section, we will show that GoodJEst achieves an estimate of the good join rate that is parameterized by factors of α and β. These factors—which impact the accuracy of the estimate provided byGoodJEst—arise because an interval may overlap more than one epoch. When we go from intervals to iterations, we will similarly accrue additional factors of α and β in our analysis for the good spend rate under Ergo, since an iteration can overlap more than one interval (see the analysis in Section 9.2 starting with Lemma 12).

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.

Lemma 2. *Assume that* $n_0 \geq 120$. *Then,* $|S(t')| \geq \frac{7}{10}|S(t)|$.

Figure 7: *Epochs* derive from our model of churn (Section 2.1.2). *Intervals* derive from GoodJEst, specifically the times at which it sets the variable $\tilde{\mathtt{J}}$ (Figure 5, Step 1). *Iterations* derive from Ergo, specifically the duration between purges (Figure 4, Step 2).

Proof. By the definition of an interval, we know that:

$$|S(t') \triangle S(t)| \leq \left\lceil \frac{5}{12}|S(t')| \right\rceil \leq \frac{5}{12}|S(t')| + 1$$

Note that $|S(t') \triangle S(t)| \geq |S(t) - S(t')|$, which implies:

$$|S(t) - S(t')| \leq \frac{5}{12}|S(t')| + 1 \tag{1}$$

Moreover, $|S(t) - S(t')| \geq |S(t)| - |S(t')|$. Rearranging, we get:

$$\begin{aligned} |S(t')| &\geq |S(t)| - |S(t) - S(t')| \\ &\geq |S(t)| - \left(\frac{5}{12}|S(t')| + 1\right) \end{aligned}$$

where the second step follows from Inequality 1. Finally, isolating $|S(t')|$ in the last inequality, we get:

$$|S(t')| \geq \frac{12}{17}\left(|S(t)| - 1\right) \geq \frac{12}{17}\left(\frac{119}{120}|S(t)|\right) \geq \left(\frac{7}{10}\right)|S(t)|$$

where the second inequality holds so long as $n_0 \geq 120$. □

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 3. *Assume that* $n_0 \geq \max\{6000, (720(\gamma+1))^{4/3}\}$. *Then, for any interval,* $a \leq 11|S(t)| + 2$.

Proof. Note that:

$$\left\lceil \frac{5}{12}|S(t')| \right\rceil = |S(t') \triangle S(t)| \geq |G(t') \triangle G(t)| \geq |G(t') - G(t)|$$

where the first step holds by the definition of an interval. Thus, we have:

$$|G(t') - G(t)| \leq \left\lceil \frac{5}{12}|S(t')| \right\rceil < \left(\frac{1}{2}\right)|G(t')| + 1$$

The last inequality holds given that the fraction of bad IDs is always less than 1/6, so $\frac{|G(t')|}{|S(t')|} > \frac{5}{6}$ implies that $|S(t')| < \frac{6}{5}|G(t')|$. This gives our first key inequality:

$$|G(t') - G(t)| < \left(\frac{1}{2}\right)|G(t')| + 1 \tag{2}$$

We can bound the probability that a good ID deleted at time $\tau \geq t$ is from the set $G(\tau) - G(t)$ by dividing both sides of Equation 2 by $G(\tau)$ to get

$$\begin{aligned} \frac{|G(\tau) - G(t)|}{|G(\tau)|} &\leq \frac{1}{2} + \frac{1}{|G(\tau)|} \\ &\leq 31/40 \end{aligned}$$

where the final inequality holds assuming that $1/|G(\tau)| \leq 11/40$, which holds when $n_0 \geq 5$. From this, we know that the probability that a good ID deleted at time τ is from $G(t)$ is at least $1 - 31/40 = 9/40$.

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. It follows that $E(X) \geq \frac{9}{40}d$, for $n_0 \geq 5$. 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 a standard Chernoff bound [92], we have:

$$\begin{aligned} Pr(X < (1-\delta)(9/40)d) &\leq e^{-\delta^2(9/40)d/2} \\ &= e^{-(1/81)(9/40)d/2} \\ &= e^{-d/720} \end{aligned}$$

where the second inequality follows from setting $\delta = 1/9$. Therefore, when $d \geq |G(t)| \geq n_0$, it follows that:

$$Pr(X < (1/5)d) \leq e^{-d/720}.$$

Therefore, $X \geq \frac{1}{5}d$, with probability of failure at most $e^{-n_0/720}$. This probability of failure is at most $1/n_0^{\gamma+1}$ for $n_0 \geq 720(\gamma+1)\ln n_0$. To derive a sufficient lower bound on n_0, note that

$$n_0/\ln n_0 \geq n_0/n_0^{1/4} = n^{4/3} \geq 720(\gamma+1)$$

where the first inequality holds for $n_0 \geq 6000$, and the last inequality holds so long as $n_0 \geq (720(\gamma+1))^{4/3} \approx 6454(\gamma+1)^{4/3}$. By a union bound, $X \geq \frac{1}{5}d$ over all intervals during the lifetime of the system, 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:

$$d \leq 5|G(t)| \tag{3}$$

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

$$\begin{aligned} a &\leq |G(t') - G(t)| + d \\ &\leq \left(\frac{1}{2}|G(t')| + 1\right) + 5|G(t)| \\ &\leq \frac{1}{2}\left(|G(t)| + a\right) + 1 + 5|G(t)| \\ &\leq \left(\frac{11}{2}\right)|G(t)| + \frac{a}{2} + 1 \end{aligned}$$

In the above, the second step follows by applying inequalities 2 and 3, 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 $a \leq 11|G(t)| + 2 \leq 11|S(t)| + 2$. □

Lemma 4. *Assume that* $n_0 \geq \max\{1681, (41\beta)^2\}$. *Then,* $a \geq \dfrac{|S(t')|}{84(1+\beta^2)} - 2 \geq 8$.

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

$$d \leq \beta^2(a+2) + 2. \tag{4}$$

By Lemma 1, 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:

$$\rho\ell + \rho'\ell' \leq \beta(a+2) \tag{5}$$

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 5, and this yields Equation 4. Next, note that:

$$\begin{aligned} |G(t') \triangle G(t)| &= |G(t') - G(t)| + |G(t) - G(t')| \\ &\leq a + d \\ &\leq a + \beta^2(a+2) + 2 \\ &\leq (1+\beta^2)(a+2) \end{aligned} \tag{6}$$

where the second to last step follows from Equation 4.

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:

$$\begin{aligned}
|G(t') \triangle G_t| &= |S(t') \triangle S(t)| - |B(t') \triangle B(t)| \\
&\geq \left(\frac{5}{12}\right) |S(t')| - |B(t') \triangle B(t)| \\
&= \left(\frac{5}{12}\right) |S(t')| - (|B(t') - B(t)| + |B(t) - B(t')|) \\
&\geq \left(\frac{5}{12}\right) |S(t')| - \frac{|S(t')|}{6} - \frac{|S(t)|}{6} \\
&\geq \left(\frac{5}{12}\right) |S(t')| - \frac{|S(t')|}{6} - \frac{10}{7}\left(\frac{|S(t')|}{6}\right) \\
&\geq \frac{|S(t')|}{84} \qquad (7)
\end{aligned}$$

The second step follows from the definition of an interval. The third step by definition of symmetric difference. The fourth step follows by the fact 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 2.

Finally, combining Inequality 6 and Inequality 7, we get:

$$(1 + \beta^2)(a + 2) \geq \frac{|S(t')|}{84}$$

On isolating a in the above, we get:

$$a \geq \left(\frac{1}{84(1 + \beta^2)}\right) |S(t')| - 2$$

Since $n_0 \geq (41\beta)^2$, we have that $\beta^2 \leq n_0/1681$. So, we get:

$$\begin{aligned}
a &\geq \left(\frac{1}{84(1 + \frac{n_0}{1681})}\right) |S(t')| - 2 \\
&\geq \left(\frac{1}{84(\frac{2n_0}{1681})}\right) |S(t')| - 2 \\
&\geq \left(\frac{10}{n_0}\right) |S(t')| - 2 \\
&\geq \left(\frac{10}{n_0}\right) |n_0| - 2 \\
&\geq 8
\end{aligned}$$

where the second line follows for $n_0 \geq 1681$ and the last line follows since $|S(t')| \geq n_0$. □

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

Lemma 5. *Let $\tilde{\mathtt{J}}$ be the estimated join rate at the end of any interval and $\mathtt{J}$ be the good join rate during the interval. Then:*

$$\mathtt{J}/21 \leq \tilde{\mathtt{J}} \leq 210\beta^2\mathtt{J}$$

Proof. At the end of the interval, GoodJEst sets the estimate of the good join rate as:

$$\begin{aligned}
\tilde{\mathtt{J}} &= \frac{|S(t')|}{t'-t} \\
&\geq \frac{7}{10}\left(\frac{|S(t)|}{t'-t}\right) \\
&\geq \frac{7}{10}\left(\frac{(a-2)/11}{t'-t}\right) \\
&\geq \frac{7}{110}\left(\frac{a-(a/4)}{t'-t}\right) \\
&\geq \frac{21}{440}\left(\frac{a}{t'-t}\right) \\
&\geq \frac{\mathtt{J}}{21}
\end{aligned}$$

The second step follows from Lemma 2, the third step from Lemma 3 using $a \leq 11|S(t)|+2$, and the fourth step from Lemma 4 using $a \geq 8$. Similarly:

$$\begin{aligned}
\tilde{\mathtt{J}} &= \frac{|S(t')|}{t'-t} \\
&\leq \frac{84(1+\beta^2)(a+2)}{(t'-t)} \\
&\leq 84(1+\beta^2)\left(\frac{5a}{4(t'-t)}\right) \\
&\leq 210\beta^2\mathtt{J}
\end{aligned}$$

The second and third steps follow from Lemma 4 using $a \geq \frac{|S(t')|}{84(1+\beta^2)} - 2$ and $a \geq 8$; the last step holds since $\beta \geq 1$ implies that $1+\beta^2 \geq 2\beta^2$. □

Lemma 6. *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:*

$$\frac{4}{5\alpha\beta}\rho \leq \mathtt{J} \leq \frac{8}{3}\alpha\beta\rho$$

Proof. Fix an interval that starts at time t and ends at time t'. By Lemma 1, we know the interval intersects at most two epochs. Let ρ and ρ' be the join rate of good IDs over the two epochs intersecting the interval over say lengths ℓ and ℓ', respectively. If only a single epoch is intersected, let ρ' and ℓ' both be 0. Then, by β-smoothness (Definition 1), we have:

$$
\begin{aligned}
J &\geq \frac{1}{t'-t}\left(\left\lfloor\frac{\rho\ell}{\beta}\right\rfloor+\left\lfloor\frac{\rho'\ell'}{\beta}\right\rfloor\right)\\
&\geq \frac{1}{t'-t}\left(\frac{\rho\ell}{\beta}+\frac{\rho'\ell'}{\beta}-2\right)\\
&\geq \frac{1}{\beta}\left(\rho\left(\frac{\ell}{t'-t}\right)+\rho'\left(\frac{\ell'}{t'-t}\right)\right)-\frac{2}{t'-t}\\
&\geq \frac{1}{\beta}\left(\rho\left(\frac{\ell}{t'-t}\right)+\left(\frac{\rho}{\alpha}\right)\frac{\ell'}{t'-t}\right)-\frac{2}{\ell}\\
&\geq \frac{\rho}{\beta\alpha}\left(\frac{\ell+\ell'}{t'-t}\right)-\frac{(J(t'-t)/4)}{t'-t}\\
&\geq \frac{\rho}{\alpha\beta}-\frac{J}{4}
\end{aligned}
$$

The fourth step follows from α-smoothness (Definition 1), and the fifth step follows from Lemma 4, specifically that $a \geq 8$. Isolating J in the last inequality, we get the lower bound.

Next, we prove the upper bound. Using β-smoothness:

$$
\begin{aligned}
J &\leq \frac{1}{t'-t}\left(\lceil\beta\rho\ell\rceil+\lceil\beta\rho'\ell'\rceil\right)\\
&\leq \beta\left(\rho\left(\frac{\ell}{t'-t}\right)+\alpha\rho\left(\frac{\ell'}{t'-t}\right)\right)+\frac{2}{t'-t}\\
&\leq \beta\left(\rho\left(\frac{\ell}{t'-t}\right)+\alpha\rho\left(\frac{\ell'}{t'-t}\right)\right)+\frac{(J(t'-t)/4)}{(t'-t)}\\
&\leq (1+\alpha)\beta\rho+\frac{J}{4}
\end{aligned}
$$

The second inequality follows from the α-smoothness, and the third inequality follows from Lemma 4, specifically that $a \geq 8$. To see the last step, note that since $\alpha \geq 1$, $1+\alpha \leq 2\alpha$, so we have:

$$J \leq (2\alpha)\beta\rho+\frac{J}{4}.$$

Isolating J in the last inequality, we obtain the upper bound. □

The next lemma makes use of Lemmas 1, 5 and 6.

Lemma 7. *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:*

$$1/\left(70\alpha^3\beta^2\right)\mathtt{J} \leq \tilde{\mathtt{J}} \leq 700\alpha^3\beta^4\mathtt{J}$$

Proof. By Lemma 1, intervals i and $i-1$ will intersect at most 4 epochs. Let ρ_1 and ρ_2 be the join rate of good IDs over the two epochs intersecting interval $i-1$. If there is only one epoch intersected, let $\rho_2 = \rho_1$. Let ρ_3 be the good join rate over the first epoch that interval i intersects.

Lower Bound. Applying Lemma 6 to interval $i-1$, we have:

$$J_{i-1} \geq \frac{4}{5\alpha\beta}\rho_2 \tag{8}$$

Applying Lemma 6 to interval i, we have:

$$\begin{aligned}
J_i &\leq \frac{4}{3}(1+\alpha)\,\beta\rho_3 \\
&\leq \frac{4}{3}(1+\alpha)\,\beta\alpha\rho_2 \\
&\leq \frac{4}{3}(1+\alpha)\alpha\beta\left(\frac{5\alpha\beta}{4}J_{i-1}\right) \\
&= \frac{5}{3}(1+\alpha)\alpha^2\beta^2 J_{i-1} \\
&\leq \frac{10}{3}\alpha^3\beta^2 J_{i-1} \\
&\leq \frac{10}{3}\alpha^3\beta^2\left(21\tilde{\mathtt{J}}_{i-1}\right) \\
&\leq 70\alpha^3\beta^2\tilde{\mathtt{J}}_{i-1}
\end{aligned}$$

The second step follows from α-smoothness, the third step follows by isolating ρ_2 in Inequality 8, the fifth step holds since $\alpha \geq 1$ from α-smoothness, the sixth step follows from Lemma 5, since $J_{i-1} \leq 21\tilde{\mathtt{J}}_{i-1}$.

Finally, isolating $\tilde{\mathtt{J}}_{i-1}$ in the last step of the above equation, we obtain the lower bound of our lemma statement.

Upper Bound. Similarly, by Lemma 6, the good-ID join rate in interval $i-1$ is:

$$J_{i-1} \leq \frac{4}{3}(1+\alpha)\,\beta\rho_2 \tag{9}$$

and for interval i is:

$$\begin{aligned}
J_i &\geq \frac{4}{5\alpha\beta}\rho_3 \\
&\geq \frac{4}{5\beta\alpha^2}\rho_2 \\
&\geq \frac{4}{5\beta\alpha^2}\left(\frac{3}{4(1+\alpha)\beta}J_{i-1}\right) \\
&\geq \frac{3}{10\alpha^3\beta^2}J_{i-1} \\
&\geq \frac{\tilde{\mathtt{J}}_{i-1}}{700\alpha^3\beta^4}.
\end{aligned}$$

The second step follows from the α-smoothness, the third step follows from inequality 9, and the fourth step holds since $\alpha \geq 1$ from α-smoothness. The fifth step follows from Lemma 5, since: $J_{i-1} \geq \tilde{\mathtt{J}}_{i-1}/(210\beta^2)$.

Finally, isolating $\tilde{\mathtt{J}}_{i-1}$ in the last step of the above equation, we obtain the lower bound. □

Now we give the proof of Theorem 2. We note that this theorem holds with high probability in n_0 for any interval, and also, by a union bound, it holds for the entire lifetime of the system over all intervals.

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 7 with those from Lemma 6,we get:

$$\left(\frac{4}{5\alpha\beta}\right)\left(\frac{1}{70\alpha^3\beta^2}\right)\rho_\tau \leq \tilde{\mathtt{J}} \leq 700\alpha^3\beta^4\left(\frac{8}{3}\alpha\beta\right)\rho_\tau$$

Simplifying this equation yields the result. □

9.2 Analysis of ERGO

For any iteration i, let $\boldsymbol{B_i}$ and $\boldsymbol{G_i}$ respectively denote the number of bad and good IDs in the system at the end of iteration i, and we let $\boldsymbol{N_i = B_i + G_i}$.

Lemma 8. *For all iterations $i \geq 0$:*

$$B_i \leq N_i\kappa/(1-\epsilon).$$

Proof. Recall that the server issues all IDs a 1-difficult challenge prior to iteration 0. This ensures the fraction of bad IDs at the end of iteration 0 is at most κ. Thus, our lemma statement holds for $i = 0$. Next, note that in Step 2 ending each iteration $i > 0$, the server issues a 1-difficult challenge to all IDs. Let N_i^s be the number of IDs in the system at the start of the purge. Recall from Section 2 that at most an $\epsilon-$fraction of good IDs can depart in a round. Thus, $N_i \geq N_i^s - \epsilon N_i^s = (1-\epsilon)N_i^s$.

Since the adversary holds at most a κ fraction of the resource, the number of bad IDs in the system at the end of iteration i is at most $\kappa N_i^s \leq \kappa N_i/(1-\epsilon)$. □

Lemma 9. *The fraction of bad IDs is always less than 3κ, provided $\kappa \leq 1/18$.*

Proof. Fix some iteration $i > 0$. Let n_i^a, b_i^a denote the total, and bad IDs that arrive over iteration i. Let n_i^d, g_i^d denote the total, and good IDs that depart over iteration i.

Recall that at the end of an iteration $n_i^a + n_i^d \geq N_{i-1}/11$. Thus, at any point during the iteration, we have:

$$b_i^a + g_i^d \quad \leq \quad n_i^a + n_i^d \leq N_{i-1}/11 \tag{10}$$

We are interested in the maximum value of the ratio of bad IDs to the total IDs at any point during the iteration. Thus, we pessimistically assume all additions of bad IDs and

removals of good IDs come first in the iteration. This leads us to examine the maximum value of the ratio over the iteration:

$$\frac{B_{i-1} + b_i^a}{N_{i-1} + b_i^a - g_i^d} \leq \frac{N_{i-1}\kappa/(1-\epsilon) + b_i^a}{N_{i-1} + b_i^a - g_i^d}$$

where the above inequality follows from Lemma 8. By Equation 10, we have $g_i^d \leq N_{i-1}/11 - b_i^a$. Thus, we have:

$$\begin{aligned}
\frac{N_{i-1}\kappa/(1-\epsilon) + b_i^a}{N_{i-1} + b_i^a - g_i^d} &\leq \frac{N_{i-1}\kappa/(1-\epsilon) + b_i^a}{N_{i-1} + b_i^a - (N_{i-1}/11 - b_i^a)} \\
&= 3\kappa \left(\frac{N_{i-1}/(3(1-\epsilon)) + b_i^a/(3\kappa)}{10N_{i-1}/11 + 2b_i^a} \right) \\
&\leq 3\kappa \left(\frac{N_{i-1}/(3(1-\epsilon))}{(10/11)N_{i-1}} + \frac{b_i^a/(3\kappa)}{(10/11)N_{i-1}} \right) \\
&\leq 3\kappa \left(\frac{11}{30(1-\epsilon)} + \frac{(1/(33\kappa))N_{i-1}}{(10/11)N_{i-1}} \right) \\
&< 3\kappa \left(\frac{11}{30(11/12)} + 3/5 \right) \\
&= 3\kappa
\end{aligned}$$

The fourth step follows since $b_i^a \leq N_{i-1}/11$ by Equation 10 and the fifth step holds for $\epsilon < 1/12$. □

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

Lemma 10. *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 11. *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:

$$\begin{aligned} n^a + n^d &\geq |S(t_1) \triangle S(t_2)| \\ &\geq \frac{5}{12}|S(t_2)| \\ &\geq \frac{5}{12}\left(\frac{10}{11}|S(t_0)|\right) \\ &= \frac{25}{66}|S(t_0)| \end{aligned}$$

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, since $25/66 > 1/11$. □

Lemma 12. *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:*

$$700\alpha^3\beta^6(\mathcal{J}\mathcal{L} + 10)$$

Proof. From Lemma 11, 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 1, 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:

$$\begin{aligned} \mathcal{J}\mathcal{L} &\geq \sum_{k=1}^{4} \left\lfloor \frac{\rho_k \ell_k}{\beta} \right\rfloor \\ &\geq \sum_{k=1}^{4} \left(\frac{\rho_k \ell_k}{\beta} - 1 \right) \\ &= \frac{1}{\beta} \sum_{k=1}^{4} \rho_k \ell_k - 4 \end{aligned} \tag{11}$$

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

$$\begin{aligned}\sum_{i=1}^{2}\lceil(t_i - t_{i-1})\tilde{\mathtt{J}}_i\rceil &\leq 700\alpha^3\beta^4\sum_{i=1}^{2}((t_i - t_{i-1})\mathtt{J}_i + 1)\\ &\leq 700\alpha^3\beta^4\left(2 + \sum_{k=1}^{4}\lceil\beta\rho_k\ell_k\rceil\right)\\ &\leq 700\alpha^3\beta^5\left(2 + \left(\sum_{k=1}^{4}\rho_k\ell_k\right) + 4\right)\\ &\leq 700\alpha^3\beta^6(\mathcal{J}\mathcal{L} + 10)\end{aligned}$$

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

Lemma 13. *The number of good IDs that join over any sub-interval is at most* $88\alpha^4\beta^4 + 1$.

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:

$$\begin{aligned}\left\lceil\beta\rho\left(\frac{1}{\tilde{\mathtt{J}}}\right)\right\rceil &\leq \beta\rho\left(88\alpha^4\beta^3\left(\frac{1}{\rho}\right)\right) + 1\\ &\leq 88\alpha^4\beta^4 + 1\end{aligned}$$

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

Lemma 14. *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 [93]:

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

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

Lemma 15. *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:

$$\begin{aligned} & \sum_{k=1}^{g_j} (b_j + k) \\ \leq \; & g_j \left(\sqrt{2\mathcal{T}_j} + g_j \right) \\ \leq \; & (88\alpha^4\beta^4 + 1) \left(\sqrt{2\mathcal{T}_j} + 88\alpha^4\beta^4 + 1 \right) \end{aligned}$$

The first step follows from Lemma 10, and the second step follows from Lemma 13.

Let t be the number of sub-intervals in the iteration. Then, the total entrance cost to the good IDs in the iteration is:

$$\begin{aligned} & \sum_{j=1}^{t} \left(\left(88\alpha^4\beta^4 + 1\right) \left(\sqrt{2\mathcal{T}_j} + (88\alpha^4\beta^4 + 1) \right) \right) \\ = \; & \left(88\alpha^4\beta^4 + 1\right) \sum_{j=1}^{t} \sqrt{2\mathcal{T}_j} + (88\alpha^4\beta^4 + 1)^2 t \\ \leq \; & (88\alpha^4\beta^4 + 1)\sqrt{2t\mathcal{T}} + (88\alpha^4\beta^4 + 1)^2 t \\ \leq \; & (88\alpha^4\beta^4 + 1)\sqrt{(1400\alpha^3\beta^6(\mathcal{J}\mathcal{L} + 10))\,\mathcal{T}} \\ & + (88\alpha^4\beta^4 + 1)^2 (700\alpha^3\beta^6(\mathcal{J}\mathcal{L} + 10)) \\ = \; & O\left(\alpha^{11/2}\beta^7 \sqrt{(\mathcal{J}\mathcal{L} + 1)\mathcal{T}} + \alpha^{11}\beta^{14}\mathcal{J}\mathcal{L} \right) \end{aligned}$$

The first step follows from Lemma 14 and by noting that $\sum_{j=1}^{t} \mathcal{T}_j = \mathcal{T}$. The third step follows from using Lemma 12 to upper bound t. □

The previous lemma bounds the entrance cost in a single iteration. The next lemma bounds the total algorithmic spending.

Lemma 16. *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-resource burning 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 resource-burning 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:

$$\begin{aligned}
\frac{12}{11}|S| &\leq \frac{12}{11}\left(11\left(d+b+g\right)\right) \\
&\leq 12\left(D\mathcal{L}+\sum_{j=1}^{t}\sqrt{2\mathcal{T}_j}+\mathcal{J}\mathcal{L}\right) \\
&\leq 12\left(D\mathcal{L}+\sqrt{2t\sum_{j=1}^{t}\mathcal{T}_j}+\mathcal{J}\mathcal{L}\right) \\
&\leq 12\left(D\mathcal{L}+\sqrt{1400\alpha^3\beta^6(\mathcal{J}\mathcal{L}+10)\mathcal{T}}+\mathcal{J}\mathcal{L}\right) \qquad (12)
\end{aligned}$$

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 10 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 14. The last step follows from Lemma 12 and substituting $\sum_{j=1}^{t}\mathcal{T}_j=\mathcal{T}$.

Finally, combining Equation 12 with the cost from Lemma 15, for some constant c, we have that the entrance costs plus the purge cost paid by all good IDs is:

$$\begin{aligned}
& c\left(\alpha^{11/2}\beta^7\sqrt{(\mathcal{J}\mathcal{L}+1)\mathcal{T}}+\alpha^{11}\beta^{14}\mathcal{J}\mathcal{L}\right) \\
& +12\left(D\mathcal{L}+\sqrt{1400\alpha^3\beta^6(\mathcal{J}\mathcal{L}+10)\mathcal{T}}+\mathcal{J}\mathcal{L}\right) \\
= \;& O\left(D\mathcal{L}+\alpha^{11/2}\beta^7\sqrt{(\mathcal{J}\mathcal{L}+1)\mathcal{T}}+\alpha^{11}\beta^{14}\mathcal{J}\mathcal{L}\right)
\end{aligned}$$

which proves 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 17; 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, $1 \leq x \leq y$. 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 17. *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 16, 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 14, we get:

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

which yields the result. □

We can now prove Theorem 1:

Proof. The resource cost bound follows immediately from Lemma 17 by noting that $\Delta(\mathcal{I}) = 0$ when $\mathcal{I}$ is all iterations, since the system is initially empty. Then, Lemma 9 completes the proof, by showing that the fraction of bad is always less than 3κ, which is no more than $1/6$, for $\kappa \leq 1/18$. □

9.3 Parameters and Constants

The relationships between certain model parameters and the constants in ERGO and GOOD-JEST is specified in detail within our analysis above.

Here, we provide an *informal* discussion of some of these relationships and highlight the connections between (1) the parameters, bounds and constants, and (2) our analysis. We emphasize that this discussion is only a supplement to and not a replacement for any of the formal analysis above.

9.3.1 Bounds

Bound on κ. Our current paper is "proof of concept" that it is possible to tolerate *some* constant value of κ efficiently, but we do not believe that $1/18$ is the largest possible constant. This non-optimal bound is a weakness of our result. However, it is common for the first theoretical result in an area to have non-optimal constant bounds.

Bound on n_0. In Section 2.1.2, we require $n_0 \geq \max\{6000, (720(\gamma + 1))^{4/3}, (41\beta)^2\}$. The first two terms in the max are needed to ensure a union bound to keep our probability of error polynomially small over the lifetime of the system (see Lemma 3). The third term, is

necessary to lower bound the number of good IDs joining in any interval by a constant — i.e. to ensure that $a \geq 8$ (see Lemma 4).

Bound on ϵ. Intuitively, if too many good IDs depart in the single round during which a purge is executed, then no bound on the fraction of bad IDs can be guaranteed; this motivates the role of ϵ used in the proof of Lemma 8. The bound $\epsilon < 1/12$ is required in Lemma 9 in order to ensure the fraction of bad never exceeds 3κ.

9.3.2 Constants Used in Our Algorithms

To minimize notation, we have prioritized use of constants rather than introducing new variables—or functions of variables—in our algorithms and analysis. Here, we briefly provide some explanation for the constants chosen.

GoodJEst. Why do we use the constant 5/12 for delineating intervals under GoodJEst? This constant results from the product of (i) the $(1-1/6) = 5/6$ lower bound on the fraction of good IDs guaranteed by Ergo, and (ii) the 1/2 fraction of good IDs that change in each epoch. The expected fraction of good IDs that change across an epoch is at least the product of these two quantities. The reason that this implies that an interval overlaps two epochs is described in the proof of Lemma 1. Hence, the constant 5/12 is used.

Ergo. In Ergo the fraction 1/11 is used to delineate iterations. This particular constant is a solution to a system of linear constraints. Details of these constraints, which involve the upper bound 1/12 on ϵ, the constant 5/12 used in GoodJEst, and the upper bound 1/18 on κ, are given in the proof of Lemma 9.

10 Experiments

We now report on several empirical contributions that complement our prior analysis by illustrating how the performance of our algorithms compares to that predicted by our upper bounds. First, in Section 10.1, we measure the resource burning cost for Ergo as a function of the adversarial cost, and we compare this against the cost resulting from prior results in the literature. Second, in Section 10.2, we evaluate the performance of the GoodJEst algorithm by measuring the approximation factor for the join rate of good IDs. Third, in Section 10.3, we propose and implement several heuristics for Ergo. All our experiments were written in MATLAB, and our source code can be found online [94].

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 [95].

• **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 [12].

• **Ethereum.** This dataset simulates join and departure events of IDs for the Ethereum network. Based on a study in [96], 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 [97], 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.

10.1 Evaluating ERGO

We compare the performance of ERGO against four resource burning based Sybil defenses: (1) CCOM [98], (2) SYBILCONTROL [67], (3) REMP (a name that uses the authors' initials)[99] and (4) ERGO-SF. Throughout our experiments, we measure performance in terms of the resource burning cost of each algorithm.

• **CCOM.** It is the same as ERGO, except the hardness of RB challenge assigned to joining IDs is always 1. Thus, CCOM does not need knowledge of the good join rate and, therefore, has no estimation component like GOODJEST.

• **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 [99] 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:

$$\mathcal{A}_{REMP} = (1-\kappa)N \times \frac{L}{W} = \frac{(1-\kappa)T_{max}}{\kappa} \tag{13}$$

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

• **ERGO-SF.** This simulates the combination of ERGO and the ML-based method SYBILFUSE; recall our discussion of SYBILFUSE in Section 6. ERGO is changed so that (1) each joining ID is classified as good or bad; and (2) All IDs that are classified as bad are refused entry. For (1), we assume a classification accuracy of 0.98, which is the average accuracy reported in [41] for experiments run over both synthetic and real-world data (Section IV - B, last paragraph).

We reiterate that, by itself, classification cannot solve DEFID, since classifier error allows the adversary to eventually accumulate a bad majority in the system. However, combining ERGO with SYBILFUSE allows us to explore the performance benefits a classifier may provide.

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

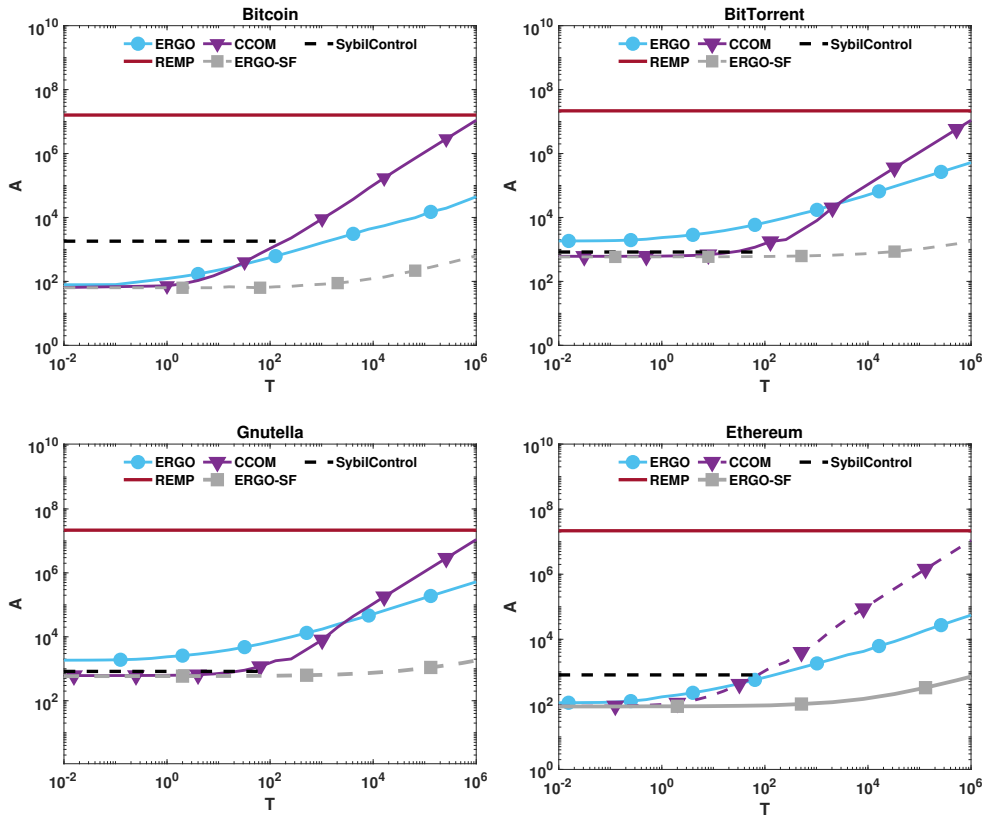

Figure 8: Illustration of the good spend rate (A) versus adversarial spend rate (T).

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 8 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 SybilControl when 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, 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.

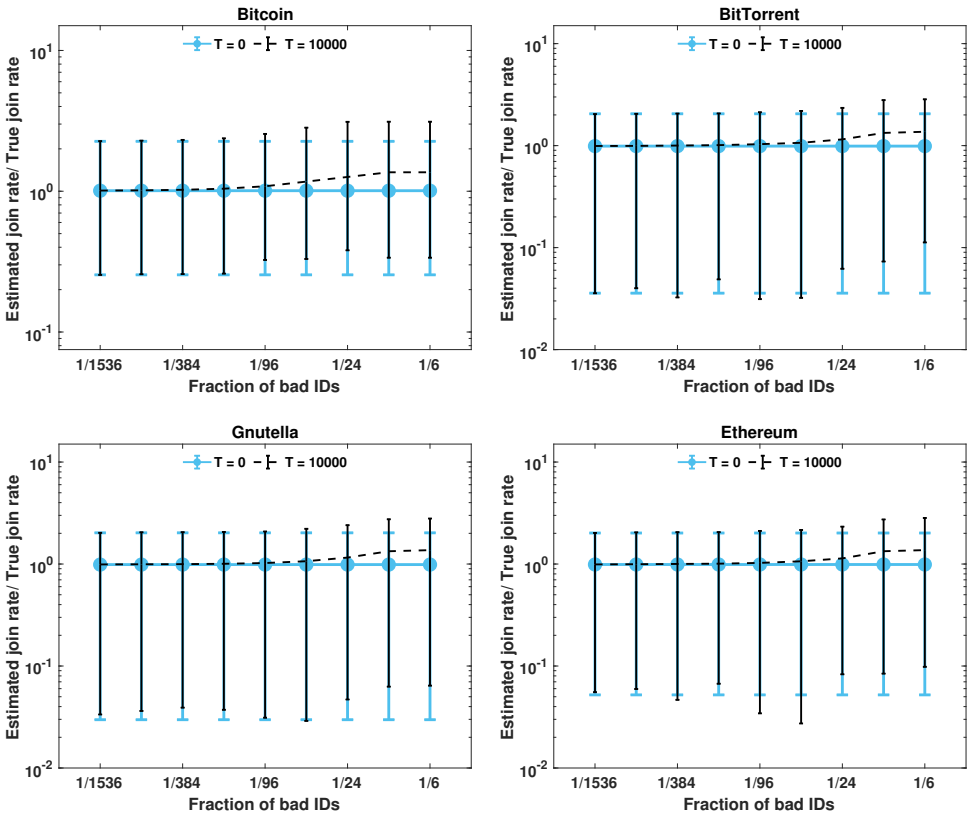

Figure 9: Illustration for the ratio of GoodJEst estimated to the true join rate for good IDs versus fraction of bad IDs.

10.2 Evaluating the performance of GoodJEst

Having witnessed the encouraging performance of Ergo (and the Ergo-based heuristic, ERGO-SF), we drill down further to examine the performance of GoodJEst. We are interested in the accuracy of estimate for the good join rate over our data sets. 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. [100]. For the remaining networks, we initialize them with 10,000 IDs each, and simulate the join and departure events over 100,000 timesteps.

In our simulations, all joins and leaves from the data sets are assumed to be good IDs. We experiment with different fractions of bad IDs that persist in the system; these fractions are $\{1/1500, 1/375, 1/94, 1/24, 1/6\}$. We note that $1/6$ actually exceeds the fraction under which our theoretical guarantees hold, but we include this value to observe the impact on performance. To test resilience, we also simulate an attack, where 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 9. These 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.

These results for GoodJEst are encouraging, since they align qualitatively with our theoretical analysis. Indeed, these results suggest that the constants in the big-O notation of our analysis are not so large that they hamper performance in practice.

10.3 Heuristics

Setup. The previous section offers encouraging evidence that GoodJEst is providing Ergo with an accurate estimate of the good join rate, thus facilitating an appropriate entrance cost. Here, we consider natural modifications to the other mechanism by which Ergo imposes resource burning—*purges*—with the aim of further reducing the resource burning performed by Ergo relative to the adversary.

We present four heuristics that aim to reduce the rate at which purges are conducted. We emphasize that none these heuristics, with the exception of Heuristic 3, sacrifice the guarantees of Theorem 1. Our modifications do not yield new theoretical results; however, our goal is to examine practical performance improvements.

Below, we describe our heuristics and provide intuition for their design.

Heuristic 1: We align the computation of $\tilde{J}$ with the end of the iteration. Specifically, at time t' marking the end of an interval, we do not necessarily calculate $\tilde{J}$ immediately using $S(t')$, but rather we wait for the end of the current iteration, if it does not also occur at time t'. After the purge occurs, we then calculate $\tilde{J}$. Thus, the fraction of bad IDs in $S(t')$ is reduced to at most κ, and this may improve the estimate from GoodJEst (recall Figure 5). Since only bad IDs are removed from $S(t')$, this heuristic does not impact our claims in Theorem 2 regarding GoodJEst.

Heuristic 2: We use the symmetric difference to determine when to do a purge. Specifically, if an iteration starts at time τ, and τ' is the current time, and if $|S(\tau) \triangle S(\tau')| \geq |S(\tau)|/11$, then a purge is executed. This still ensures that the fraction of bad IDs can increase by no more than in our original specification, but it also decreases the purge frequency. For example, consider the case where the adversary causes a single bad ID to join and depart repeatedly, and there is no other churn. Notably, this behavior does not threaten the bound of ensuring less than a 1/6-fraction of bad IDs and so there is no need to purge. However, since Ergo uses the number of joins and departures to demarcate iterations, a purge will occur nonetheless. In contrast, using the symmetric difference heuristic will avoid this.

Heuristic 3: When the threshold condition for a purge holds in Ergo, we do an additional check, which is as follows. Determine if the total join rate over the current iteration is less than some constant c times the good ID join-rate estimate from the prior iteration. If it is greater, then a purge is performed, since we have seen many more joins than expected based on the prior estimate. Else, we go to the next iteration *without* purging. In our experiments, we set $c = 1/11$. We note that this heuristic can fail to enforce correctness in the case where $c < \alpha$. However, in our experiments, we confirmed that for all data sets, the heuristic still always kept the total fraction of bad IDs less than 1/6.

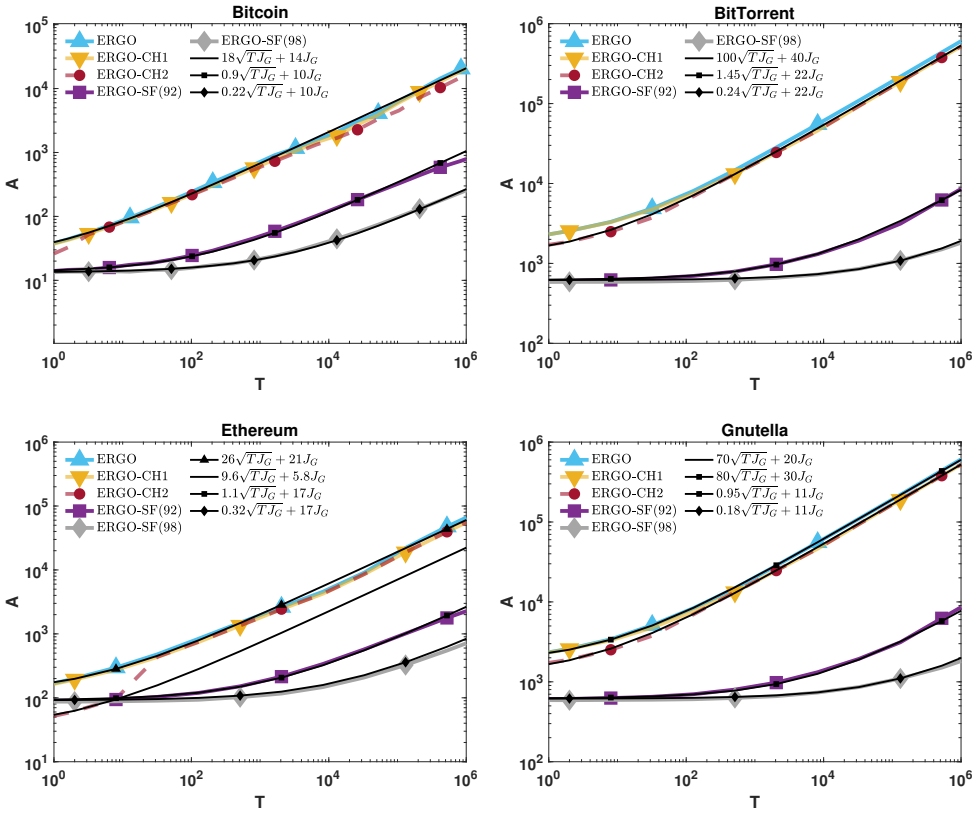

Figure 10: Algorithmic cost versus adversarial cost for ERGO and heuristics.

Heuristic 4: This is **ERGO-SF**, which we described in Section 10.1.

We evaluate the performance of the above heuristics against ERGO. The experimental setup is the same as Section 10.1. We define ***ERGO-CH1*** using both Heuristic 1 and 2, and ***ERGO-CH2*** using Heuristics 1, 2, and 3. In the name, the "CH" stands for combined heuristic. We define ***ERGO-SF(92)*** and ***ERGO-SF(98)*** as ERGO using Heuristics 1, 2, 3, and 4, with the accuracy of SYBILFUSE in Heuristic 4 set as 0.98 and 0.92, respectively. In the name, the "SF" stands for SYBILFUSE. We simulated Heuristic 1 in conjunction with ERGO, but, by itself, it did not yield improvements, and so we omit it from our plots.

Results. Figure 10 illustrates our results. ERGO-SF(92) and ERGO-SF(98) reduce costs significantly during adversarial attack, with improvements of up to three orders of magnitude during the most significant attack tested. Again, these improvements are consistent across 4 different types of data sets.

Taken together, these results indicate one of the most important heuristics is the use of a classifier. For all data sets but Ethereum, ERGO-SF(92) and ERGO-SF(98) always perform better than any other heuristic, and for all data sets, ERGO-SF(92) and ERGO-SF(98) perform best, with an increasing performance gap, for large values of T.

We note that the RB cost for Bitcoin and Ethereum when T is small is smaller than for BitTorrent and Gnutella. This behavior is likely caused by the higher churn rates inherent

to the latter networks as documented in prior literature [101, 14, 95, 102], which results in more frequent purges.

Also related to the case for small T, we observe that Heuristic 3 yields good performance for the Ethereum data set and, to a lesser degree, the Gnutella data set. We would expect Heuristic 3 to work particularly well when (1) the good join rate does not change much from iteration to iteration; and (2) there are not many bad IDs joining the system. Based on our results for this heuristic, we speculate that the Ethereum network has a churn rate with less "variability" than the other networks.

11 A Lower Bound

In the previous sections, we derived upper bounds for our algorithms and explored their empirical performance. In this section, we provide a lower bound that applies to the class of algorithms which have the attributes $B1 - B3$ described below.

- **B1.** Each new ID must pay an entrance fee in order to join the system and this is defined by a ***cost function*** $\boldsymbol{f}$, which takes as inputs the good join rate and the adversarial join rate.

- **B2.** The algorithm executes over iterations, delineated by when the condition $a + d \geq \delta\, n$ holds, for any fixed positive δ, where a and d are the number of arrivals and departures over the iteration, and n is the number of IDs in the system at the start of the iteration.

- **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 cost during an iteration depends only on the good join rate and the bad join rate. Our analysis of ERGO makes this assumption since its cost function depends on estimates of the good join rate and the total, i.e. the good-ID join rate plus the bad-ID join rate. With regard to B2 and B3, recall that we wish to ensure that the fraction of bad IDs is always less than some constant fraction. It is hard to imagine an algorithm that preserves this invariant without an RB challenge being imposed on all IDs whenever the system membership changes significantly.

11.1 Lower-Bound Analysis

Restating in terms of the conditions above, we have the following result.

Theorem 3. *Suppose an algorithm satisfies conditions B1-B3, then there exists an adversarial strategy that forces the algorithm to spend at a rate of* $\Omega(\sqrt{T\,J} + J)$*, where* J *is the good ID join rate, and* T *is the algorithmic spending rate, both taken over the iteration.*

Proof. Fix an iteration. Let n be the number of IDs in the system at the start of the iteration. The adversarial will have bad IDs join uniformly at the maximum rate possible, and then have the bad IDs drop out during the purge. In particular, let J^B be the rate at which bad IDs join, and let $f(J^B, J)$ be the algorithm's entrance cost function based on J^B and J; we pessimistically assume that the algorithm knows both J^B and J exactly. Then $J^B = T/f(J^B, J)$.

We first calculate the algorithmic spending rate due to purge puzzle costs in the iteration. Since the iteration ends after $O(n)$ join events (B2), and since each purge puzzle has a cost of $\Omega(1)$ (B3), spending on purge puzzles is asymptotically at least equal to the number of good and bad IDs join the system. Thus, the spending due to purge puzzles is $\Omega(J + J^B)$. We now have two cases:

Case 1: $f(J^B, J) \leq J^B/J$. In this case, we have:

$$J^B \;\geq\; Jf(J^B, J) = JT/J^B$$

where the above equality holds since $f(J^B, J) = T/J^B$. Solving for J^B we get:

$$J^B \;\geq\; \sqrt{T\,J}$$

Since the algorithmic spending rate due to purge costs is $\Omega(J^B + J)$, the total algorithmic spending rate is:

$$\Omega(J^B + J) \;=\; \Omega\left(\sqrt{T\,J} + J\right)$$

Case 2: $f(J^B, J) > J^B/J$. In this case, we have:

$$J^B \;<\; J\,f(J^B, J) = JT/J^B$$

The above equality follows since $f(J^B, J) = T/J^B$. Solving for J^B, we get:

$$J^B \;<\; \sqrt{T\,J}.$$

The spending rate for the algorithm due to entrance costs is $\Omega(J\,f(J^B, J))$. Adding in the spending rate for purge costs of $\Omega(J + J^B)$, we get that the total spend rate of the algorithm is:

$$\begin{aligned}\Omega(J\,f(J^B, J) + (J + J^B)) &= \Omega(J\,T/J^B + (J + J^B)) \\ &= \Omega\left(\sqrt{T\,J} + J\right).\end{aligned}$$

□

12 Decentralization

When there is no centralized server, we can run our algorithms using a ***committee***: a $O(\log n_0)$ 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 modifications needed to maintain and use such a committee.

The results in this section are straight-forward applications of tools that have already been developed in prior literature, such as state-machine replication and committee election.

Model Modifications. Our communication model inherits the assumptions made in [103] and [28]. These assumptions allow us to implement State Machine Replication and Committee Election, respectively.

Thus, we assume synchronous communication, and that there exist secure and authenticated communication channels between all pairs of IDs in the committee. Additionally, we require a secure and authenticated communication channel between each member of the committee and each ID in the system. This enables challenge solutions to be sent to the committee, and election information to be sent to all participants.

Further, in order to use the algorithm by Rabin and Ben-Or [104] for committee selection (see below in Section 12.2), the adversary is oblivious to the private, random bits generated by any good ID.

12.1 Committee and System Initialization

GenID. To initialize our system, we require a solution to GenID (recall Section 2.1.2). 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 2 [18, 37, 36, 38]. The algorithm in [38], 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.

12.2 Use and Maintenance of Committee

The committee makes use of ***State Machine Replication (SMR)*** (see [105, 39, 103, 106]) to agree on an ordering of network events so as to execute GoodJEst and Ergo in parallel. In order to scalably and correctly execute SMR, we maintain the invariant that the committee always 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 $C \log N_i$, 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 [104]. Note that subsequent results can accomplish the same

task more efficiently, but require cryptographic assumptions. For example, Awerbuch and Scheideler [29] describe an algorithm specifically for random number generation that can be used by the existing committee to select new committee members.

With the modifications above, ERGO gives the following guarantees:

Theorem 4. *For $\kappa \leq 1/18$,* ERGO *ensures that the good spend rate is:*

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

and guarantees:

1. *The fraction of bad IDs in the system is less than* $1/6$.
2. *The fraction of bad IDs in the committee is at most* $1/8$.

Much of the proof of Theorem 4 follows from earlier arguments, but we must still show that the committee does indeed always have size $\Theta(\log n_0)$ and a good majority. Thus, the following lemma is useful.

Lemma 18. *The following holds for all iterations $i > 0$, with high probability.* ERGO *maintains a committee with at least a $7/8$ fraction of good IDs. Moreover, this committee has size $\Theta(\log n_0)$.*

Proof. For iteration $i = 0$, the lemma holds true by properties of the GENID algorithm used to initialize the system (See Section 12.1, and Lemma 6 of [107]).

Fix an iteration $i > 0$. Recall that B_i and G_i are the number of bad and good IDs in the system at the end of iteration i, respectively, $N_i = B_i + G_i$. Recall that a new committee is elected by the existing committee at the end of an iteration by selecting $C \log N_i$ IDs independently and uniformly at random from the set S_i, for a sufficiently large constant $C > 0$, defined concretely later in this proof. Let X_G (X_B) be random variables for the number of good (bad) IDs, respectively elected to the new committee at the end of iteration i. Then:

$$\begin{aligned}
E[X_G] &= \frac{G_i}{N_i} C \log N_i \\
&= \left(1 - \frac{B_i}{N_i}\right) C \log N_i \\
&\geq \left(1 - \frac{1}{18(1-\epsilon)}\right) C \log N_i \\
&\geq \left(1 - \frac{1}{18(11/12)}\right) C \log N_i \\
&= \frac{31}{33} C \log N_i \qquad (14)
\end{aligned}$$

In the above, the third step follows from Lemma 8 and fourth step since $\epsilon < 1/12$. Similarly, the expected number of bad IDs elected into the committee is:

$$E[X_B] = \frac{B_i}{N_i} C \log N_i \leq \frac{1}{18(1-\epsilon)} C \log N_i \tag{15}$$

Next, by Chernoff bounds [108], for any constant $0 < \delta < 1$:

$$\begin{aligned}
&Pr\left(X_G \leq (1-\delta)\frac{31}{33} C \log N_i\right) \\
&\leq \exp\left\{-\frac{31\delta^2 C \log N_i}{66}\right\} \\
&= N_i^{-\frac{31C\delta^2}{66}} \\
&\leq n_0^{-(\gamma+1)}
\end{aligned}$$

The last step holds for all $C \geq \frac{66(\gamma+1)}{31\delta^2}$. For $\delta = 13/310$, we obtain that the number of good IDs in the committee is at least $(9/10)C \log N_i$ w.h.p.

Again:

$$\begin{aligned}
&Pr\left(X_B \geq (1+\delta)\frac{2}{33} C \log N_i\right) \\
&\leq \exp\left\{-\frac{2\delta^2 C \log N_i}{99}\right\} \\
&= N_i^{-\frac{2C\delta^2}{99}} \\
&\leq n_0^{-(\gamma+1)}
\end{aligned}$$

The last step holds for all $C \geq \frac{99(\gamma+1)}{2\delta^2}$. For $\delta = 13/20$, we obtain that the number of bad IDs in the committee is at most $(1/10)C \log N_i$ w.h.p.

Next, let Y_g be the number of good IDs that depart from the committee during iteration i. Note that the number of good departures in iteration i is at most $N_{i-1}/11$. Then, since each departing good ID is selected independently and uniformly at random (See Section 2), we obtain:

$$\begin{aligned}
E[Y_g] &\leq \frac{N_{i-1}}{11}\left(\frac{C \log N_i}{N_i}\right) \\
&\leq \left(\frac{11N_i/10}{11}\right)\frac{C \log N_i}{N_i} \\
&= \frac{C}{10}\log N_i
\end{aligned}$$

The second step holds since over an iteration at most $N_{i-1}/11$ IDs depart, so $N_i \geq 10N_{i-1}/11$.

Again, using Chernoff bounds, we have:

$$Pr\left(Y_g \geq (1+\delta')\frac{C}{10}\log N_i\right) \leq \exp\left\{-\frac{\delta'^2 C}{30}\log N_i\right\}$$
$$= N_i^{-\delta'^2 C/30}$$
$$\leq n_0^{-(\gamma+1)}$$

where the first step holds for any constant $0 < \delta' < 1$ and the last step holds for all $C \geq \frac{30(\gamma+1)}{\delta'^2}$. Thus, for $\delta' = 1/9$, the number of good IDs in the committee is always greater than $(9/10)C\log N_i - (1/9)C\log N_i = (71/90)C\log N_i$.

Next, the fraction of good IDs in the committee is minimized when only good IDs depart from the committee. Thus, the fraction of good IDs in the committee is always at least:

$$\frac{(71/90)C\log N_i}{C\log N_i - (1/9)C\log N_i} > \frac{7}{8}$$

To bound the committee size, note that the committee always has a number of IDs at least:

$$C\log N_i - \frac{C}{10}\log N_i - \frac{C}{9}\log N_i = \Theta(\log n_0)$$

Finally, we use a union bound over all n_0^γ iterations to show that the above facts hold, with high probability, for all iterations. □

The proof of Theorem 4 now follows from Lemmas 9, 17, and 18.

13 Conclusion and Future Work

ERGO is a new Sybil-defense that efficiently employs resource-burning to limit Sybil IDs, despite high churn. Specifically, ERGO guarantees (1) there is always a minority of Sybil IDs; and (2) despite a churn rate that can vary exponentially, the resource burning rate of good IDs is $O(\sqrt{TJ} + J)$, where T is the resource burning rate of the adversary, and J is the join rate of good IDs. Our experiments illustrate that ERGO significantly decreases algorithmic resource costs when compared to other Sybil defenses. We also show empirically that costs can be further reduced by combining ERGO with SYBILFUSE, thus illustrating the benefits of leveraging classification algorithms. Finally, we prove that the resource burning rate of $O(\sqrt{TJ} + J)$ is asymptotically optimal for a large class of algorithms.

There are two future research directions that we feel are important, which we discuss next.

13.1 Incentives

Under ERGO, recall that good IDs must periodically solve a 1-hard puzzle to avoid being purged from the system. In practice, as discussed in Section 3.1, the system may benefit from

providing incentives for honest users to obey this protocol. Thus, an interesting direction for future work is devising a mechanism for incentivizing good IDs to solve these puzzles. For example, as is the case for cryptocurrencies like Bitcoin, we might make use of rewards. For example, during the purge, competition for a reward could be used to ensure that IDs actually commit sufficient resources to remain in the system. If challenges are proof-or-work based, the ID that finds the smallest solution during this period could receive units of cryptocurrency, such as that received for mining a block. In this way, all good IDs would have positive incentives to commit resources. Hence, the difficulty of a 1-hard puzzle could be tuned, based on measured computational effort, to automatically adjust to new, faster hardware or heterogeneous hardware distribution. Formally analyzing the game-theoretic properties of such a scheme, using techniques similar to those used in [109, 110, 111], is left for future work.

13.2 Distributed Hash Tables

Can we apply the results in this paper to build and maintain a Sybil-resistant distributed hash table (DHT) [20]? To the best of our knowledge, there is no such result that ensures the good IDs pay a cost that is slowly growing function of both the good churn rate and the cost payed by an attacker. Third, can a similar approach be used to mitigate distributed denial-of-service (DDoS) attacks at the application layer? Here, server resources can be exhausted by bad clients whose spurious jobs cannot be *a priori* distinguished from legitimate jobs. It seems plausible that a resource-burning approach similar to ERGO might offer a defense here too. Fourth, while our lower bound applies to a large class of algorithms, it would be interesting to establish a more general result.

13.3 Tolerating $\kappa > 1/18$

In Section 9.3, we discussed the interaction between various parameters and constants, and we highlighted that there is some flexibility in selecting their values such that our formal arguments hold with only cosmetic changes. Here, we elaborate on this aspect in order to highlight the challenges involved with tolerating a larger value of κ, along with sketching ideas for how this might be accomplished.

To begin, first consider what happens if we wish to alter GOODJEST such that an interval ends whenever:

$$|S(t') \triangle S(t)| \geq \frac{1}{2}|S(t')|.$$

That is, we wish to change the constant $5/12$ to be $1/2$. It suffices to change the definition of an *epoch* so that their boundaries occur when the symmetric difference between the sets of good IDs at the start and the end of the epoch exceeds $3/5$ (rather than $1/2$) times the number of good IDs at the start. This can be seen by the proof for Lemma 1, since now the key inequality in the argument is $|S(t_2) \triangle S(t_0)| \geq (3/5)(5/6) = 1/2$, which ends an epoch under this new definition, and the proof follows as before with this minor change.

There are two reasons this concrete example is useful. First, it illustrates how the values we use in our arguments have some flexibility. Second, it is important to understanding why

the structure of our current analysis cannot accommodate a significantly larger κ, which we now discuss.

What prevents ERGO from tolerating a larger κ under our current analysis? While we have flexibility to choose values for our parameters and constants, this flexibility has limits. Specifically, a key obstacle occurs in achieving Inequality 6 within the proof of Lemma 4. Namely, we require a sufficiently large constant—currently this is 5/12—in order to achieve Inequality 6. Consider what happens if we wish to tolerate $\kappa \leq 1/9$ (rather than $\kappa \leq 1/18$), and thus we will now have a strict bound of $3\kappa = 1/3$ on the fraction of bad IDs in the system at any time (recall Lemma 9). Note that the chain of inequalities that leads to Inequality 6 no longer holds, since the constant 5/12 is too small.

But can we not increase this constant as we did in the concrete example above? Unfortunately, if we wish to tolerate a strict bound of 1/3 on the fraction of bad IDs, the cosmetic change performed in that prior example is insufficient. This can be observed by simply following through the proofs for Lemma 1 and the chain of inequalities that leads to Inequality 6. To be explicit, recall that an interval ends when:

$$|S(t_2) \triangle S(t_0)| \geq (E)(2/3) = I$$

where E is the constant used for delineating epochs, and I is the constant used for delineating intervals; recall that the latter is currently 5/12 and is the value we wish to make larger. However, note that $I \leq 2/3$ no matter what value we choose for E. Therefore, for the argument of Lemma 1 to remain correct, we must end intervals whenever:

$$|S(t') \triangle S(t)| \geq \frac{2}{3}|S(t')|.$$

What are the implications for the Inequality 6 used in the proof of Lemma 4? The portion of the argument:

$$\begin{aligned} |G(t') \triangle G_t| &\geq ... \\ &\geq \left(\frac{5}{12}\right)|S(t')| - \frac{|S(t')|}{6} - \frac{10}{7}\left(\frac{|S(t')|}{6}\right) \\ &\geq \frac{|S(t')|}{84} \end{aligned}$$

is critical. However, with our hypothetical modification, we have:

$$\begin{aligned} |G(t') \triangle G_t| &\geq ... \\ &\geq \left(\frac{2}{3}\right)|S(t')| - \frac{|S(t')|}{3} - d\left(\frac{|S(t')|}{3}\right) \\ &\not> 0 \end{aligned}$$

where d is a constant now exceeding 10/7, which can be seen from following through the proof of Lemma 2 using 2/3 instead of 5/12. Thus, we no longer achieve $|G(t') \triangle G_t| = \Omega(S(t'))$, but rather end up with a negative value.

Tolerating larger values of κ is an area of future work. It may be possible to sufficiently increase I by allowing for an interval to overlap *more* than two epochs; this would require significant changes to Lemma 1 and other arguments in Section 9.1. Note that this such changes would likely also impact the accuracy of GoodJEst, since if an interval overlaps more epochs, the exponents for α and β in Theorem 2 will increase as a result.

Acknowledgements. This work is supported by the National Science Foundation grants CNS 1816076, CNS 1816250, CNS 2210299, CNS 2210300, and CCF 2144410.

References

[1] D. Gupta, J. Saia, M. Young, Bankrupting Sybil despite churn, in: Proceedings of the 41st IEEE International Conference on Distributed Computing Systems (ICDCS), 2021, pp. 425–437.

[2] D. Gupta, J. Saia, M. Young, Peace through superior puzzling: An asymmetric Sybil defense, in: Proceedings of the $33rd$ IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2019, pp. 1083–1094.

[3] J. Douceur, The Sybil attack, in: Proceedings of the Second International Peer-to-Peer Symposium (IPTPS), 2002, pp. 251–260.

[4] S. Zhang, J.-H. Lee, Double-spending with a Sybil attack in the bitcoin decentralized network, IEEE transactions on Industrial Informatics 15 (10) (2019) 5715–5722.

[5] E. Heilman, A. Kendler, A. Zohar, S. Goldberg, Eclipse attacks on Bitcoin's peer-to-peer network, in: 24th {USENIX} Security Symposium ({USENIX} Security 15), 2015, pp. 129–144.

[6] T. Neudecker, Bitcoin cash (BCH) Sybil nodes on the Bitcoin peer-to-peer network, `http://dsn.tm.kit.edu/publications/files/332/bch_sybil.pdf` (2017).

[7] A. Mohaisen, J. Kim, The Sybil attacks and defenses: A survey, Smart Computing Review 3 (6) (2013) 480–489.

[8] D. Gupta, J. Saia, M. Young, Invited paper: Resource burning for permissionless systems, in: 27^{th} International Conference on Structural Information and Communication Complexity (SIROCCO), 2020, pp. 19–44.
URL `https://arxiv.org/abs/2006.04865`

[9] C. Dwork, M. Naor, Pricing via processing or combatting junk mail, in: Proceedings of the 12^{th} Annual International Cryptology Conference on Advances in Cryptology, 1993, pp. 139–147.

[10] T. Economist, Why Bitcoin uses so much energy, `https://www.economist.com/the-economist-explains/2018/07/09/why-bitcoin-uses-so-much-energy` (2018).

[11] A. 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).

[12] D. Stutzbach, R. Rejaie, Understanding churn in peer-to-peer networks, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC), ACM, New York, NY, USA, 2006, pp. 189–202. doi:10.1145/1177080.1177105.
URL http://doi.acm.org/10.1145/1177080.1177105

[13] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, T. Anderson, Profiling a million user DHT, in: Proceedings of the 7^{th} ACM SIGCOMM Conference on Internet Measurement, 2007, pp. 129–134.

[14] L. Wang, J. Kangasharju, Measuring large-scale distributed systems: Case of BitTorrent Mainline DHT, in: IEEE 13th International Conference on Peer-to-Peer Computing (P2P), 2013, pp. 1–10.

[15] D. Malkhi, The BFT Lens: Hot-Stuff and Casper, https://dahliamalkhi.github.io/posts/2018/03/bft-lens-casper/ (2018).

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

[17] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, http://bitcoin.org/bitcoin.pdf (2008).

[18] M. Andrychowicz, S. Dziembowski, PoW-based distributed cryptography with no trusted setup, in: Proceedings of the Annual Cryptology Conference, Springer, 2015, pp. 379–399.

[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: Scaling Byzantine agreements for cryptocurrencies, in: Proceedings of the 26th Symposium on Operating Systems Principles (SOSP), 2017, pp. 51–68.

[20] G. Urdaneta, G. Pierre, M. van Steen, A survey of DHT security techniques, ACM Computing Surveys 43 (2) (2011) 1–53.

[21] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, in: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), 2001, pp. 149–160.

[22] R. Guerraoui, F. Huc, A.-M. Kermarrec, Highly dynamic distributed computing with Byzantine failures, in: Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing (PODC), 2013, pp. 176–183.

[23] M. O. Jaiyeola, K. Patron, J. Saia, M. Young, Q. M. Zhou, Tiny groups tackle Byzantine adversaries, in: Proceedings of the IEEE International Parallel and Distributed Processing Symposium, IPDPS, 2018, pp. 1030–1039.

[24] A. Fiat, J. Saia, M. Young, Making Chord robust to Byzantine attacks, in: Proceedings of the 13^{th} European Symposium on Algorithms (ESA), 2005, pp. 803–814.

[25] B. Awerbuch, C. Scheideler, Group spreading: A protocol for provably secure distributed name service, in: Proceedings of the 31^{st} International Colloquium on Automata, Languages, and Programming (ICALP), 2004, pp. 183–195.

[26] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, D. S. Wallach, Secure routing for structured peer-to-peer overlay networks, in: Proceedings of the 5^{th} Usenix Symposium on Operating Systems Design and Implementation (OSDI), 2002, pp. 299–314.

[27] M. Young, A. Kate, I. Goldberg, M. Karsten, Towards practical communication in Byzantine-resistant DHTs, IEEE/ACM Transactions on Networking 21 (1) (2013) 190–203.

[28] B. Awerbuch, C. Scheideler, Towards scalable and robust overlay networks, in: Proceedings of the 6^{th} International Workshop on Peer-to-Peer Systems (IPTPS), 2007.

[29] B. Awerbuch, C. Scheideler, Robust random number generation for peer-to-peer systems, in: Proceedings of the $10th$ International Conference On Principles of Distributed Systems (OPODIS), 2006, pp. 275–289.

[30] B. Awerbuch, C. Scheideler, Towards a scalable and robust DHT, in: Proceedings of the 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2006, pp. 318–327.

[31] C. Scheideler, How to spread adversarial nodes? Rotate!, in: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing (STOC), 2005, pp. 704–713.

[32] T. Wang, C. Zhao, Q. Yang, S. Zhang, S. C. Liew, Ethna: Analyzing the underlying peer-to-peer network of Ethereum blockchain, IEEE Transactions on Network Science and Engineering (2021).

[33] P. Maymounkov, D. Mazieres, Kademlia: A peer-to-peer information system based on the xor metric, Lecture Notes in Computer Science (2002) 53–65.

[34] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. E. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, R. Wattenhofer, On scaling decentralized blockchains (A position paper), in: Proceedings of the International Financial Cryptography and Data Security (FC), 2016, pp. 106–125.

[35] J. Aspnes, C. Jackson, A. Krishnamurthy, Exposing computationally-challenged Byzantine impostors, Tech. rep., YALEU/DCS/TR-1332, Yale University http://www.cs.yale.edu/homes/aspnes/papers/tr1332.pdf (2005).

[36] J. Katz, A. Miller, E. Shi, Pseudonymous secure computation from time-lock puzzles, IACR Cryptol. ePrint Arch. 2014 (2014) 857.
URL http://eprint.iacr.org/2014/857

[37] R. Hou, I. Jahja, L. Luu, P. Saxena, H. Yu, Randomized view reconciliation in permissionless distributed systems, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), 2018, pp. 2528–2536.

[38] A. Aggarwal, M. Movahedi, J. Saia, M. Zamani, Bootstrapping public blockchains without a trusted setup, in: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC), ACM, 2019, pp. 366–368.

[39] N. A. Lynch, Distributed algorithms, Elsevier, San Francisco, CA, 1996.

[40] M. Ben-Or, S. Goldwasser, A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computing, in: Proceedings of the Twentieth ACM Symposium on the Theory of Computing (STOC), 1988, pp. 1–10.

[41] P. Gao, B. Wang, N. Z. Gong, S. R. Kulkarni, K. Thomas, 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, pp. 1–9, https://www.princeton.edu/~pmittal/publications/sybilfuse-cns18.pdf.

[42] J. Augustine, G. Pandurangan, P. Robinson, E. Upfal, Towards robust and efficient computation in dynamic peer-to-peer networks, in: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2012, pp. 551–569.

[43] J. Augustine, G. Pandurangan, P. Robinson, E. Upfal, Distributed agreement in dynamic peer-to-peer networks, Journal of Computer and System Sciences 81 (7) (2015) 1088–1109.

[44] J. Augustine, G. Pandurangan, P. Robinson, Fast byzantine leader election in dynamic networks, in: International Symposium on Distributed Computing, Springer, 2015, pp. 276–291.

[45] T. Jacobs, G. Pandurangan, Stochastic analysis of a churn-tolerant structured peer-to-peer scheme, Peer-to-peer Networking and Applications 6 (1) (2013) 1–14.

[46] J. Augustine, G. Pandurangan, P. Robinson, S. Roche, E. Upfal, Enabling robust and efficient distributed computation in dynamic peer-to-peer networks, in: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science, IEEE, 2015, pp. 350–369.

[47] J. Augustine, A. R. Molla, E. Morsy, G. Pandurangan, P. Robinson, E. Upfal, Storage and search in dynamic peer-to-peer networks, in: Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures, 2013, pp. 53–62.

[48] J. Augustine, G. Pandurangan, P. Robinson, Distributed algorithmic foundations of dynamic networks, ACM SIGACT News 47 (1) (2016) 69–98.

[49] S. Y. Ko, I. Hoque, I. Gupta, Using tractable and realistic churn models to analyze quiescence behavior of distributed protocols, in: 2008 Symposium on Reliable Distributed Systems, IEEE, 2008, pp. 259–268.

[50] M. K. Aguilera, A pleasant stroll through the land of infinitely many creatures, ACM Sigact News 35 (2) (2004) 36–59.

[51] D. Liben-Nowell, H. Balakrishnan, D. Karger, Analysis of the evolution of peer-to-peer systems, in: Proceedings of the twenty-first annual symposium on Principles of distributed computing, 2002, pp. 233–242.

[52] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, J. Zahorjan, Measurement, modeling, and analysis of a peer-to-peer file-sharing workload, in: Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003, pp. 314–329.

[53] M. A. Imtiaz, D. Starobinski, A. Trachtenberg, N. Younis, Churn in the bitcoin network: Characterization and impact, in: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, 2019, pp. 431–439.

[54] R. John, J. P. Cherian, J. J. Kizhakkethottam, A survey of techniques to prevent Sybil attacks, in: Proceedings of the International Conference on Soft-Computing and Networks Security (ICSNS), 2015, pp. 1–6.

[55] J. Newsome, E. Shi, D. Song, A. Perrig, The Sybil attack in sensor networks: Analysis & defenses, in: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN), 2004, pp. 259–268.

[56] L. Wang, J. Kangasharju, Real-world Sybil attacks in BitTorrent Mainline DHT, in: Proceedings of the IEEE Global Communications Conference (GLOBECOM), 2012, pp. 826–832.

[57] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, Y. Dai, Uncovering social network Sybils in the wild, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC), 2011, pp. 259–268.

[58] H. Yu, M. Kaminsky, P. B. Gibbons, A. Flaxman, Sybilguard: Defending against Sybil attacks via social networks, in: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), 2006, pp. 267–278.

[59] A. Mohaisen, S. Hollenbeck, Improving social network-based Sybil defenses by rewiring and augmenting social graphs, in: Proceedings of the 14^{th} International Workshop on Information Security Applications (WISA), 2014, pp. 65–80.

[60] W. Wei, F. Xu, C. C. Tan, Q. Li, SybilDefender: A defense mechanism for Sybil attacks in large social networks, IEEE Transactions on Parallel & Distributed Systems 24 (12) (2013) 2492–2502.

[61] S. Misra, A. S. M. Tayeen, W. Xu, Sybilexposer: An effective scheme to detect Sybil communities in online social networks, in: 2016 IEEE International Conference on Communications (ICC), IEEE, 2016, pp. 1–6.

[62] M. Sherr, M. Blaze, B. T. Loo, Veracity: Practical secure network coordinates via vote-based agreements, in: Proceedings of the USENIX Annual Technical Conference, 2009, pp. 13 –13.

[63] Y. Liu, D. R. Bild, R. P. Dick, Z. M. Mao, 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) 2376–2391.

[64] S. Gil, S. Kumar, M. Mazumder, D. Katabi, D. Rus, Guaranteeing spoof-resilient multi-robot networks, in: Proceedings of Robotics: Science and Systems, Rome, Italy, 2015, pp. 1383–1400.

[65] G. Danezis, C. Lesniewski-laas, M. F. Kaashoek, R. Anderson, Sybil-resistant DHT routing, in: Proceedings of the 10^{th} European Symposium On Research In Computer Security (ESORICS), 2005, pp. 305–318.

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

[67] F. Li, P. Mittal, M. Caesar, N. Borisov, SybilControl: Practical Sybil defense with computational puzzles, in: Proceedings of the Seventh ACM Workshop on Scalable Trusted Computing, 2012, pp. 67–78.

[68] T. Moran, I. Orlov, Simple proofs of space-time and rational proofs of storage, in: Annual International Cryptology Conference, Springer, 2019, pp. 381–409.

[69] A. Shoker, Sustainable blockchain through proof of exercise, in: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), IEEE, 2017, pp. 1–9.

[70] M. Ball, A. Rosen, M. Sabin, P. N. Vasudevan, Proofs of work from worst-case assumptions, in: Proceedings of the Annual International Cryptology Conference (CRYPTO), Springer, 2018, pp. 789–819.

[71] L. Von Ahn, M. Blum, N. J. Hopper, J. Langford, CAPTCHA: Using hard AI problems for security, in: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2003, pp. 294–311.

[72] M. Moradi, M. Keyvanpour, CAPTCHA and its alternatives: A review, Security and Communication Networks 8 (12) (2015) 2135–2156.

[73] H. S. Baird, M. A. Moll, S.-Y. Wang, Scattertype: A legible but hard-to-segment CAPTCHA, in: Proceedings of the Eighth International Conference on Document Analysis and Recognition (ICDAR), 2005, pp. 935–939.

[74] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, M. Blum, Recaptcha: Human-based character recognition via web security measures, Science 321 (5895) (2008) 1465–1468.

[75] D. Mónica, L. Leitao, L. Rodrigues, 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, 2009, pp. 21–26.

[76] S. Gilbert, C. Zheng, Sybilcast: Broadcast on the open airwaves, in: Proceedings of the 25^{th} Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2013, pp. 130–139.

[77] S. Gilbert, C. Newport, C. Zheng, Who are you? Secure identities in ad hoc networks, in: Proceedings of the 28^{th} International Symposium on Distributed Computing (DISC), 2014, pp. 227–242.

[78] A. Kiayias, A. Russell, B. David, R. Oliynykov, Ouroboros: A provably secure proof-of-stake blockchain protocol, in: Proceedings of the $37th$ Annual International Cryptology Conference, Vol. 10401 of Lecture Notes in Computer Science, Springer, 2017, pp. 357–388.

[79] O. Kharif, Bye-bye, miners! How Ethereum's big change will work, `https://www.bloombergquint.com/quicktakes/bye-bye-miners-how-ethereum-s-big-change-will-work-quicktake` (2021).

[80] CoinDesk, Vulnerable? Ethereum's Casper tech takes criticism at Curacao event, `https://www.coindesk.com/fundamentally-vulnerable-ethereums-casper-tech-takes-criticism-curacao` (2018).

[81] D. Gupta, J. Saia, M. Young, Proof of work without all the work, arXiv preprint arXiv:1708.01285 (2017).

[82] S. Gilbert, M. Young, Making Evildoers Pay: Resource-Competitive Broadcast in Sensor Networks, in: Proceedings of the 31^{th} Symposium on Principles of Distributed Computing (PODC), 2012, pp. 145–154.

[83] S. Gilbert, V. King, S. Pettie, E. Porat, J. Saia, M. Young, (Near) optimal resource-competitive broadcast with jamming, in: Proceedings of the 26^{th} ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2014, pp. 257–266.

[84] V. King, J. Saia, M. Young, Conflict on a communication channel, in: Proceedings of the 30^{th} Symposium on Principles of Distributed Computing (PODC), 2011, pp. 277–286.

[85] M. A. Bender, J. T. Fineman, S. Gilbert, M. Young, How to scale exponential backoff: Constant throughput, polylog access attempts, and robustness, in: Proceedings of the 27^{th} Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2016, pp. 636–654.

[86] V. Dani, M. Movahedi, J. Saia, M. Young, Interactive communication with unknown noise rate, in: Proceedings of the Colloquium on Automata, Languages, and Programming (ICALP), 2015, pp. 464–486.

[87] V. Dani, T. Hayes, M. Movahedi, J. Saia, M. Young, Interactive communication with unknown noise rate, Information and Computation 261 (2017) 464–486.

[88] A. Aggarwal, V. Dani, T. Hayes, J. Saia, Secure one-way interactive communication, in: Proceedings of the 15th International Conference on Distributed Computing and Networking (ICDCN), 2017.

[89] S. Gilbert, V. King, J. Saia, M. Young, Resource-competitive analysis: A new perspective on attack-resistant distributed computing, in: Proceedings of the 8^{th} ACM International Workshop on Foundations of Mobile Computing, 2012, pp. 1–6.

[90] M. Zamani, J. Saia, J. Crandall, Torbricks: Blocking-resistant tor bridge distribution, in: International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Springer, 2017, pp. 426–440.

[91] M. A. Bender, J. T. Fineman, M. Movahedi, J. Saia, V. Dani, S. Gilbert, S. Pettie, M. Young, Resource-competitive algorithms, SIGACT News 46 (3) (2015) 57–71.

[92] D. Dubhashi, A. Panconesi, Concentration of Measure for the Analysis of Randomized Algorithms, 1st Edition, Cambridge University Press, 2009.

[93] R. A. Horn, C. R. Johnson, Matrix analysis, Cambridge University Press, Cambridge, UK, 2012.

[94] D. Gupta, Source code link, `https://www.dropbox.com/sh/ao19eqnresc530k/AAAUF5oPqBJo5ql_CT7T9WbNa?dl=0`, last updated on August 25, 2022 (2022).

[95] T. Neudecker, P. Andelfinger, 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), 2015, pp. 1327–1332.

[96] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, M. Bailey, Measuring Ethereum network peers, in: Proceedings of the Internet Measurement Conference, 2018, pp. 91–104.

[97] H. Rowaihy, W. Enck, P. McDaniel, T. La Porta, Limiting Sybil attacks in structured P2P networks, in: Proceedings of the IEEE Intl. Conference on Computer Communications (INFOCOM), 2007, pp. 2596–2600.

[98] D. Gupta, J. Saia, M. Young, Proof of work without all the work, in: Proceedings of the 19^{th} International Conference on Distributed Computing and Networking (ICDCN), 2018, pp. 1–10.

[99] H. Rowaihy, W. Enck, P. Mcdaniel, T. La Porta, Limiting Sybil attacks in structured peer-to-peer networks, in: IEEE INFOCOM Mini-Symposium, 2005.

[100] T. Neudecker, P. Andelfinger, H. Hartenstein, Timing analysis for inferring the topology of the Bitcoin peer-to-peer network, in: Proceedings of the 13th IEEE International Conference on Advanced and Trusted Computing (ATC), 2016, pp. 358–367.

[101] S. Saroiu, P. K. Gummadi, S. D. Gribble, Measurement study of peer-to-peer file sharing systems, in: Multimedia computing and networking 2002, Vol. 4673, SPIE, 2001, pp. 156–170.

[102] L. Kiffer, A. Salman, D. Levin, A. Mislove, C. Nita-Rotaru, Under the hood of the ethereum gossip protocol, in: International Conference on Financial Cryptography and Data Security, Springer, 2021, pp. 437–456.

[103] I. Abraham, D. Malkhi, K. Nayak, L. Ren, M. Yin, Sync HotStuff: Simple and practical synchronous state machine replication, IACR Cryptology ePrint Archive (2019).

[104] T. Rabin, M. Ben-Or, Verifiable secret sharing and multiparty protocols with honest majority, in: Proceedings of the 21^{st} Annual ACM symposium on Theory of Computing, 1989, pp. 73–85.

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

[106] A. Bessani, J. Sousa, E. E. Alchieri, State machine replication for the masses with BFT-SMART, in: Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, IEEE, 2014, pp. 355–362.

[107] M. Andrychowicz, S. Dziembowski, D. Malinowski, Ł. Mazurek, Modeling Bitcoin contracts by timed automata, in: Proceedings of the International Conference on Formal Modeling and Analysis of Timed Systems, Springer, 2014, pp. 7–22.

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

[109] I. Abraham, D. Malkhi, K. Nayak, L. Ren, A. Spiegelman, Solidus: An Incentive-compatible Cryptocurrency Based on Permissionless Byzantine Consensus, CoRR, abs/1612.02916 (2016).

[110] N. B. Margolin, B. N. Levine, Informant: Detecting Sybils using incentives, in: International Conference on Financial Cryptography and Data Security, Springer, 2007, pp. 192–207.

[111] Z. Liu, N. C. Luong, W. Wang, D. Niyato, P. Wang, Y.-C. Liang, D. I. Kim, A survey on blockchain: A game theoretical perspective, IEEE Access 7 (2019) 47615–47643.